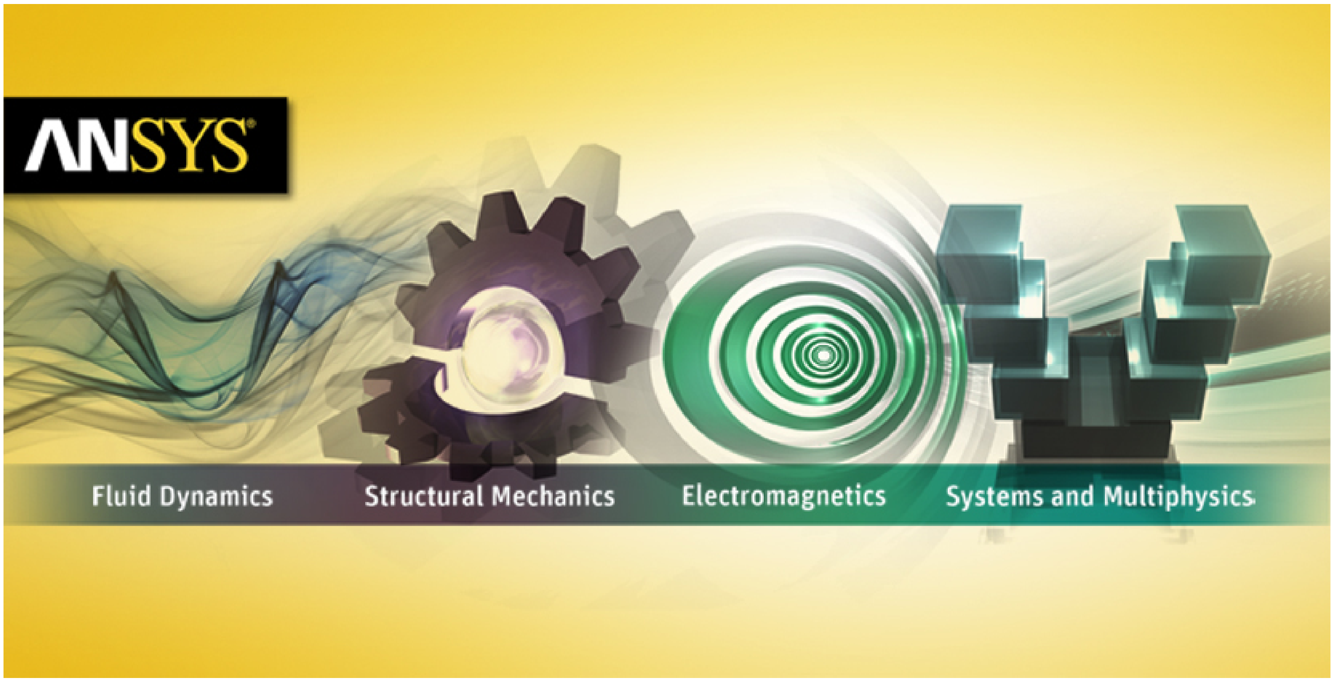


ANSYS POLYFLOW User's Guide

ANSYS, Inc.
Southpointe
275 Technology Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 14.0
November 2011

Copyright and Trademark Information

Disclaimer Notice

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. is certified to ISO 9001:2008.

U.S. Government Rights

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

Third-Party Software

See the legal information in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, please contact ANSYS, Inc.

Published in the U.S.A.

Table of Contents

Using This Manual

1. What's in This Manual

The ANSYS POLYFLOW User's Guide tells you how to use ANSYS POLYFLOW. *Material Properties* (p. 201) through *Computing Derived Quantities* (p. 551) describe the physical models and solution techniques available in ANSYS POLYFLOW and how to use them.

An index allows you to look up material relating to a particular subject in order to help you find answers to your questions quickly and directly, whether you are a first-time user or an experienced user.

Important

Under U.S. and international copyright law, ANSYS, Inc. is unable to distribute copies of the papers listed in the bibliography, other than those published internally by ANSYS, Inc. Use your library or a document delivery service to obtain copies of copyrighted papers.

A brief description of each chapter follows:

- *Getting Started* (p. 1) gives you an overview about the capabilities of ANSYS POLYFLOW and the way in which it interacts with GAMBIT and other preprocessors, modules, and third-party programs. It also gives you information about starting ANSYS POLYFLOW.
- *ANSYS POLYDATA Graphical User Interface* (p. 17) describes the ANSYS POLYDATA user interface. Since ANSYS POLYFLOW runs based on input from the data file created by ANSYS POLYDATA, there is no user interface for ANSYS POLYFLOW itself.
- *Managing ANSYS POLYFLOW Projects Using ANSYS POLYMAN* (p. 29) describes how you can start GAMBIT, ANSYS POLYDATA, ANSYS POLYFLOW, FieldView, CFD-Post, and other modules (ANSYS POLYMAT, ANSYS POLYDIAG, ANSYS POLYFUSE, ANSYS POLYSTAT) from a single starting point.
- *Sample Session* (p. 61) presents a sample session designed to give you a general idea of how ANSYS POLYMAN, ANSYS POLYDATA, ANSYS POLYFLOW, and CFD-Post work. Detailed information for the modeling options that are used is provided in other chapters.
- *Reading and Writing Files* (p. 103) contains information about the files that ANSYS POLYFLOW and its modules can read and write.
- *Unit Systems* (p. 129) describes the unit systems available in ANSYS POLYFLOW.
- *Meshes* (p. 133) describes the various sources of computational meshes, mesh-to-mesh interpolation, and combination of multiple meshes.
- *Boundary Conditions* (p. 173) explains the different types of flow boundary conditions available in ANSYS POLYFLOW, when to use them, and how to define them.
- *Material Properties* (p. 201) presents general information about defining material properties.
- *Generalized Newtonian Flow* (p. 207) presents the equations that ANSYS POLYFLOW uses to calculate generalized Newtonian flow, and describes the associated setup and solution procedures.
- *Viscoelastic Flows* (p. 225) presents the equations solved by ANSYS POLYFLOW for viscoelastic flows, and describes the related setup and solution procedures.
- *Flow Induced Crystallization (FIC)* (p. 271) introduces the basic equations for a (single-mode) viscoelastic model that incorporates a flow induced crystallization mechanism.

- *Heat Transfer* (p. 283) describes how ANSYS POLYFLOW models nonisothermal flow and heat conduction, and explains how to set up a nonisothermal flow or heat conduction problem.
- *Porous Media* (p. 305) describes the Darcy model used by ANSYS POLYFLOW to solve porous media problems, and explains how to set up and solve a porous media model.
- *Free Surfaces and Extrusion* (p. 311) describes ANSYS POLYFLOW's approach to modeling free surfaces and moving interfaces, and explains how to set up and solve problems involving them.
- *Remeshing* (p. 343) describes the different remeshing methods available in ANSYS POLYFLOW, when to use them, and how to define them.
- *Blow Molding and Thermoforming* (p. 379) describes how ANSYS POLYFLOW models blow molding and thermoforming, and explains how to set up and solve these types of problems.
- *Film Casting* (p. 417) describes ANSYS POLYFLOW's approach to modeling film casting, and explains how to set up and solve this type of problem.
- *Chemically Reacting Flows* (p. 429) describes ANSYS POLYFLOW's approach to modeling chemical reactions, and explains how to set up and solve this type of problem.
- *Volume of Fluid (VOF) Model* (p. 443) describes how ANSYS POLYFLOW can model a liquid with free surfaces using an Eulerian approach that does not require remeshing.
- *Flows with Internal Moving Parts* (p. 457) describes the superposition technique used by ANSYS POLYFLOW to model flows with internal moving parts, such as extruders and mixing tanks, and explains how to set up and solve this type of model.
- *Sliding Mesh Technique* (p. 477) describes the use of sliding mesh technique for solving transient flows with internal moving parts.
- *Glass Furnaces and Electrical Heating* (p. 485) describes how ANSYS POLYFLOW models phenomena related to glass furnaces, including electrical heating and bubbling. Explanations of how to set up and solve problems of this type are also provided in this chapter.
- *Residual Stresses and Deformations* (p. 493) describes how to calculate residual stresses and deformations in glass using ANSYS POLYFLOW's Narayanaswamy model.
- *Fluid Structure Interaction (FSI) Model* (p. 503) provides information about elasticity in a solid and the coupled interaction with the flow.
- *Evolution* (p. 517) describes the evolution technique available in ANSYS POLYFLOW for solving nonlinear problems, and explains how to use it.
- *Time-Dependent Flows* (p. 539) describes how ANSYS POLYFLOW solves transient flows, and explains how to set up and solve them.
- *Computing Derived Quantities* (p. 551) describes the “postprocessor" quantities that ANSYS POLYFLOW can solve, and explains how to specify them during the problem setup.
- *Using the Solver* (p. 575) provides information about how to control the ANSYS POLYFLOW solver.
- *User-Defined Functions (UDFs)* (p. 597) explains how you can customize ANSYS POLYFLOW using user-defined functions for boundary conditions, material properties, sources, etc.
- *User-Defined Templates (UDTs)* (p. 615) describes how you can use templates to create a baseline case that can be easily modified via a simplified ANSYS POLYDATA interface, which provides access only to preselected parameters. It also explains how to run a series of similar simulations that vary by a few parameters.
- *Die Shape Parameterization* (p. 619) describes how to alter the mesh geometry strictly via user input, as part of a preprocessor task.

- *Optimization* (p. 651) describes ANSYS POLYFLOW's internal optimizer, and how to use it to adjust the geometry, material data parameters, flow rates, boundary condition parameters, and/or sub-model parameters such that an objective function (e.g., the flow imbalance at a die exit) is maximized or minimized.
- *Appendix A* (p. 697) provides information about the compatibility of sub-tasks.
- *Appendix B* (p. 701) provides additional information about how to couple ANSYS POLYFLOW with the external optimizer Visual DOC, in order to manage optimization that is based on geometric or other parameters (e.g., material data, operating conditions).

2. What's in the Other Manuals

In addition to this User's Guide, there are a number of other manuals available to help you use ANSYS POLYFLOW and its associated modules and programs:

- The ANSYS POLYFLOW Tutorial Guide contains a number of example problems with complete detailed instructions, commentary, and postprocessing of results.
- The ANSYS POLYSTAT User's Guide explains how to set up a **MIXING task** in ANSYS POLYDATA and how to use the ANSYS POLYSTAT module for statistical postprocessing of results.
- The ANSYS POLYMAT User's Guide explains how to use the ANSYS POLYMAT module for material property evaluation.
- The GAMBIT manuals teach you how to use the GAMBIT preprocessor for geometry creation and mesh generation.
- The CFD-Post User's Guide explains how to use CFD-Post to examine your results.
- The ANSYS POLYFLOW Examples Manual provides overviews of solutions to a variety of problem types.
- The ANSYS POLYFLOW in Workbench User's Guide explains how to use the ANSYS POLYFLOW application within ANSYS Workbench.
- The ANSYS POLYFLOW in ANSYS Workbench Tutorial contains an example problem that illustrates the use of the ANSYS POLYFLOW application within ANSYS Workbench, with complete detailed instructions, commentary, and postprocessing of results.

3. Typographical Conventions Used in This Manual

Several typographical conventions are used in the text of this manual to facilitate your learning process.

- Different typographical fonts are used to indicate graphical user interface menu items and text inputs that you enter (e.g., **Flow boundary conditions** panel, enter the name `fluids`).
- In ANSYS POLYDATA and ANSYS POLYMAN, a mini flow chart is used to indicate the menu selections that lead you to a specific panel. For example, **Display** → **Contours...** indicates that the **Contours...** menu item can be selected from the **Display** pull-down menu at the top of the ANSYS POLYMAN console window.

  The words before the arrow invoke menus (or submenus) and the arrows point from a specific menu toward the item you should select from that menu.

4. Contacting Technical Support

Technical Support for ANSYS, Inc. products is provided either by ANSYS, Inc. directly or by one of our certified ANSYS Support Providers. Please check with the ANSYS Support Coordinator (ASC) at your company to determine who provides support for your company, or go to www.ansys.com and select

About ANSYS> Contacts and Locations. The direct URL is: http://www1.ansys.com/customer/public/supportlist.asp. Follow the on-screen instructions to obtain your support provider contact information. You will need your customer number. If you don't know your customer number, contact the ASC at your company.

If your support is provided by ANSYS, Inc. directly, Technical Support can be accessed quickly and efficiently from the ANSYS Customer Portal, which is available from the ANSYS Website (www.ansys.com) under **Support> Technical Support** where the Customer Portal is located. The direct URL is: http://www.ansys.com/customerportal.

One of the many useful features of the Customer Portal is the Knowledge Resources Search, which can be found on the Home page of the Customer Portal.

Systems and installation Knowledge Resources are easily accessible via the Customer Portal by using the following keywords in the search box: `Systems/Installation`. These Knowledge Resources provide solutions and guidance on how to resolve installation and licensing issues quickly.

NORTH AMERICA

All ANSYS, Inc. Products

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

Toll-Free Telephone: 1.800.711.7199

Fax: 1.724.514.5096

Support for University customers is provided only through the ANSYS Customer Portal.

GERMANY

ANSYS Mechanical Products

Telephone: +49 (0) 8092 7005-55

Email: support@cadfem.de

All ANSYS Products

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

National Toll-Free Telephone:

German language: 0800 181 8499

English language: 0800 181 1565

International Telephone:

German language: +49 6151 3644 300

English language: +49 6151 3644 400

Email: support-germany@ansys.com

UNITED KINGDOM

All ANSYS, Inc. Products

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

Telephone: +44 (0) 870 142 0300

Fax: +44 (0) 870 142 0302

Email: support-uk@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

JAPAN

CFX , ICEM CFD and Mechanical Products

Telephone: +81-3-5324-8333

Fax: +81-3-5324-7308

Email: *CFX:* japan-cfx-support@ansys.com; *Mechanical:* japan-ansys-support@ansys.com

FLUENT Products

Telephone: +81-3-5324-7305

Email: *FLUENT:* japan-fluent-support@ansys.com; *POLYFLOW:* japan-polyflow-support@ansys.com; *FfC:* japan-ffc-support@ansys.com; *FloWizard:* japan-flowizard-support@ansys.com

Icepak

Telephone: +81-3-5324-7444

Email: japan-icepak-support@ansys.com

Licensing and Installation

Email: japan-license-support@ansys.com

INDIA

ANSYS Products (including FLUENT, CFX, ICEM-CFD)

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

Telephone: +91 1 800 233 3475 (toll free) or +91 1 800 209 3475 (toll free)

Fax: +91 80 2529 1271

Email: *FEA products:* feasup-india@ansys.com; *CFD products:* cfdsup-india@ansys.com; *Installation:* installation-india@ansys.com

FRANCE

All ANSYS, Inc. Products

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

Toll-Free Telephone: +33 (0) 800 919 225

Email: support-france@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

BELGIUM

All ANSYS Products

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

Telephone: +32 (0) 10 45 28 61

Email: support-belgium@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

SWEDEN

All ANSYS Products

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

Telephone: +44 (0) 870 142 0300

Email: support-sweden@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

SPAIN and PORTUGAL

All ANSYS Products

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

Telephone: +33 1 30 60 15 63

Email: support-spain@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

ITALY

All ANSYS Products

Web: Go to the ANSYS Customer Portal (http://www.ansys.com/customerportal) and select the appropriate option.

Telephone: +39 02 89013378

Email: support-italy@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

Chapter 1: Getting Started

This chapter provides an introduction to ANSYS POLYFLOW, an overview of how to use it, and instructions for starting it. In addition, a sample session is presented. A description of the `.p3rc` configuration file is also provided here.

1.1. Introduction

ANSYS POLYFLOW is a finite-element computational fluid dynamics (CFD) program designed primarily for simulating applications where viscous and viscoelastic flows play an important role. The flows can be isothermal or nonisothermal, two- or three-dimensional, steady-state or time-dependent. ANSYS POLYFLOW is used primarily to solve flow problems in polymer and rubber processing, food rheology, glasswork furnaces, and many other rheological applications. The calculation of such flows is based on non-Newtonian fluid mechanics, characterized by a wide variety of fluid models and strong nonlinearities. The development of ANSYS POLYFLOW is intimately linked to progresses in numerical simulation of non-Newtonian fluid mechanics; the most recent and best-performing algorithms are incorporated in ANSYS POLYFLOW on a regular basis. The selection of constitutive models available in ANSYS POLYFLOW is also based on current research in the area.

ANSYS POLYFLOW can also be used to solve chemically reacting flows. Transport of species as well as chemical reactions that act as sources or sinks of materials can be included.

It is possible to detect contact during ANSYS POLYFLOW simulations. This capability makes ANSYS POLYFLOW useful for blow molding, thermoforming, and compression molding simulations. A major advantage is that access to a library of non-Newtonian materials is maintained for all contact problems. ANSYS POLYFLOW also provides additional capabilities for glass furnaces, such as bubbling, radiative correction, and electrical heating.

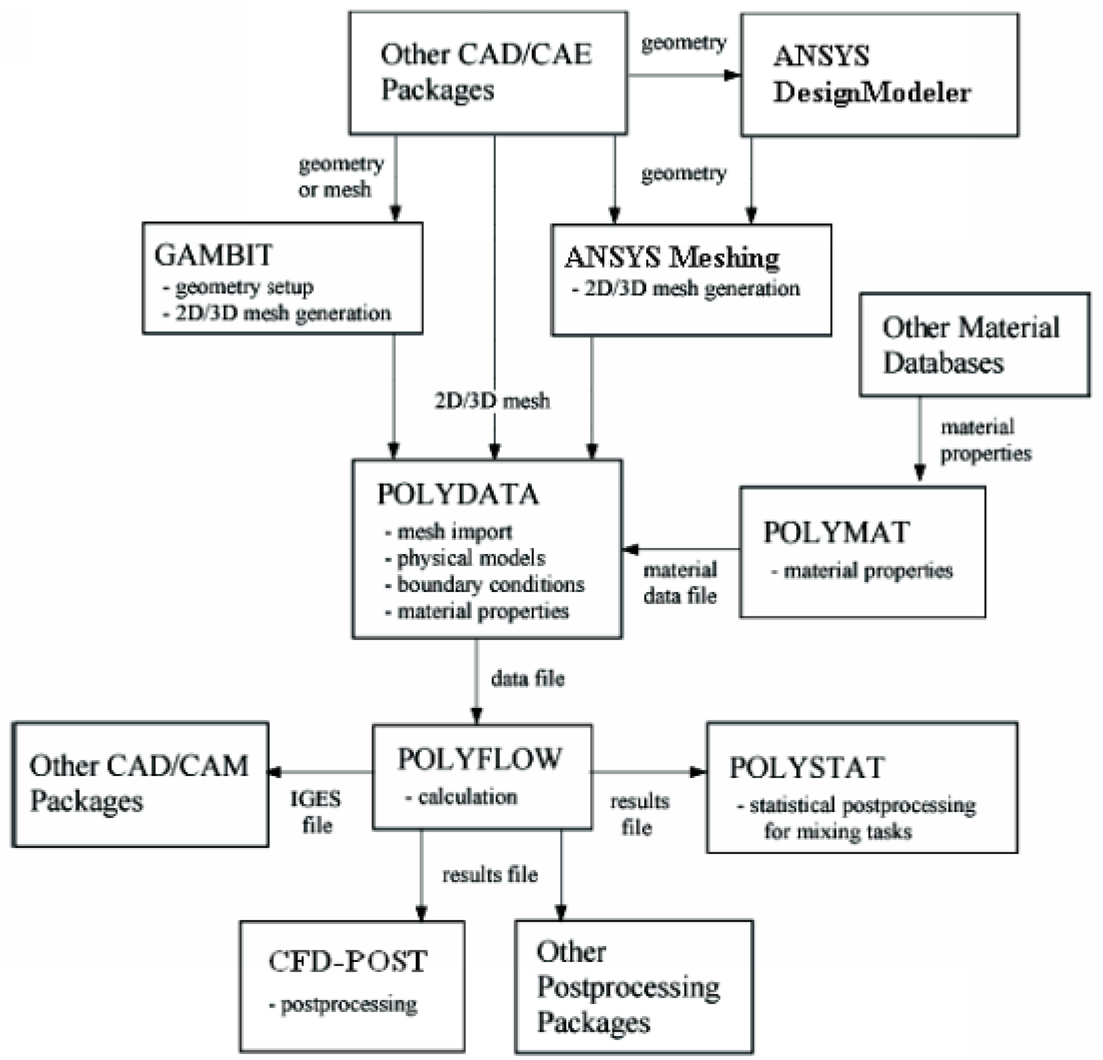
ANSYS POLYFLOW can perform a number of complex calculations such as multidomain simulations, coextrusion of several fluids, three-dimensional extrusion, and implicit and time-dependent calculation of free surfaces.

Note

When running ANSYS POLYFLOW on Linux, Itanium platforms are not supported.

1.2. Program Structure

Figure 1.1 Basic Program Structure



1.2.1. The ANSYS POLYFLOW Package

The ANSYS POLYFLOW package includes the following products and modules:

- ANSYS POLYFLOW, the solver
- ANSYS POLYDATA, the preprocessor for problem definition
- GAMBIT, the preprocessor for geometry modeling and mesh generation
- ANSYS POLYMAT, the preprocessor for material data specification
- ANSYS POLYSTAT, the statistical postprocessor for quantitative comparison of flows
- CFD-Post, the graphical postprocessor for examining results
- filters (translators) for import of meshes from CAD/CAE packages such as PATRAN and I-deas, and export of meshes and results to these and other programs

Figure 1.1 (p. 2) shows the organizational structure of these components.

ANSYS POLYMAN is an environment that allows you to manage your ANSYS POLYFLOW projects and start the modules and products listed above from a single entry point. See *Managing ANSYS POLYFLOW Projects Using ANSYS POLYMAN* (p. 29) for details.

You can create your geometry and mesh using GAMBIT. See the GAMBIT documentation for details. ANSYS ICEM CFD (POWERMESH) and POLYMESH (preprocessors that were used before the introduction of GAMBIT) can also be used to create meshes for ANSYS POLYFLOW. It is also possible to create meshes for ANSYS POLYFLOW using the PATRAN and I-deas third-party CAD/CAE packages and other software packages that support these file formats.

Once your mesh is created, you can read it into ANSYS POLYDATA and set up the simulation. In ANSYS POLYDATA, you will define the physical models, material properties, boundary and process conditions, numerical parameters, etc. When you have completed the problem definition, you will save it to a data file, which can be used to run ANSYS POLYFLOW.

In conjunction with the problem specification in ANSYS POLYDATA, you may want to use the ANSYS POLYMAT preprocessor for some preliminary material property analysis. ANSYS POLYMAT allows you compute material properties based on experimental or other data. The result of the ANSYS POLYMAT calculation is material property data that is passed to ANSYS POLYDATA through a material data file. The use of ANSYS POLYMAT is optional; it is generally used when you need to determine complex material property data for your model.

ANSYS POLYFLOW is the central solver. It computes a solution based on the problem definition specified in the data file that you created in ANSYS POLYDATA, and saves the solution to a results file. You can also start an ANSYS POLYFLOW calculation from a previous results file. Such a restarting procedure is useful, for example, in nonlinear problems where you wish to save CPU time.

When you have completed your calculation in ANSYS POLYFLOW, you can use the graphical postprocessor CFD-Post to examine your results. Other postprocessing packages that can be used to examine ANSYS POLYFLOW results are FieldView, CFView-PF, ANSYS POLYPLOT, V3DMSH, DataVisualizer, PATRAN, and I-deas.

In addition to the graphical postprocessors, a statistical postprocessor called ANSYS POLYSTAT is also available. ANSYS POLYSTAT allows you to interactively analyze properties calculated along particle trajectories and perform statistical calculations that can be used to predict mixing efficiency and other macroscopic flow properties. ANSYS POLYSTAT operates on a set of trajectories created by ANSYS POLYFLOW for a mixing task.

1.2.1.1. Application-Specific Versions of ANSYS POLYFLOW

Besides the standard package described previously, ANSYS POLYFLOW offers application-specific versions of ANSYS POLYDATA and the solver. These application-specific versions limit the capabilities to only those needed to set up and perform simulations of specific industrial processes, including the following:

- blow molding
- extrusion

In order to use an application-specific version of ANSYS POLYFLOW, you must have a license for that version or the standard ANSYS POLYFLOW license. In terms of how you set up and utilize application-specific versions, the information provided in the sections that follow about ANSYS POLYFLOW sessions applies equally to the application-specific versions. See *Starting ANSYS POLYDATA and ANSYS POLYFLOW* (p. 10) for details on how to launch the application-specific versions.

1.3. Overview of Using ANSYS POLYFLOW

1.3.1. Planning Your ANSYS POLYFLOW Analysis

When you are planning to solve a problem using ANSYS POLYFLOW, you should first give consideration to the following issues:

- Definition of the modeling goals: What specific results are required from the CFD model and how will they be used? What degree of accuracy is required from the model?
- Choice of the computational model: How will you isolate a piece of the complete physical system to be modeled? Where will the computational domain begin and end? What boundary conditions will be used at the boundaries of the model? Can the problem be modeled in two dimensions, or is a 3-dimensional or 2.5-dimensional model required?
- Choice of physical models: Is the flow steady or unsteady? Is heat transfer important? Are there other physical models that should be applied?
- Determination of the solution procedure: Can the problem be solved simply, using the default solution parameters? Can convergence be accelerated with a more aggressive solution procedure? Can the problem benefit from a staged (evolution) solution technique? Will the problem fit within the memory constraints of your computer? How long will the problem take to converge on your computer?

Careful consideration of these issues before beginning your CFD analysis will contribute significantly to the success of your modeling effort. When you are planning a CFD project, take advantage of the customer support provided to all ANSYS POLYFLOW users.

1.3.2. Problem Solving Steps

After determining the important features of the problem you want to solve, perform the following basic procedural steps:

1. Create the model geometry and mesh, using GAMBIT or another preprocessor.
2. Start ANSYS POLYDATA.
3. If necessary, convert the mesh file.
4. Read the mesh file into ANSYS POLYDATA.
5. Define the physical models and, if appropriate, time dependence or evolution by specifying a task and a sub-task (described in *Tasks and Sub-Tasks* (p. 5)).
6. Specify the material properties.
7. Specify the boundary and process conditions.
8. Define the remeshing (for moving boundary problems only).
9. Create another sub-task, if necessary, and define the appropriate parameters as described above for the first sub-task.
10. Assign the field parameters (stream function and/or pressure).
11. Create another task, if necessary, and set the appropriate parameters as described above for the first task.
12. Specify the output format for the results, if different from the default postprocessor, CFD-Post.
13. Save the data file and exit from ANSYS POLYDATA.
14. Calculate a solution with ANSYS POLYFLOW, using the data file as input.

15. Examine the results, using CFD-Post or another graphical postprocessor.
16. If necessary, consider revisions to the numerical or physical model.

1.4. Basic Concepts

1.4.1. Tasks and Sub-Tasks

The concepts used in problem definition with ANSYS POLYDATA require some explanation. First, the problem that you are solving (e.g., analysis of flow through a die) is referred to as a task. Within a given task, the computations that ANSYS POLYFLOW will perform are divided into sub-tasks. There is always at least one task and one sub-task, but there may be more than one sub-task if different physics apply and need to be solved simultaneously.

1.4.1.1. Tasks

A task consists of the following parameters:

- Geometrical dimension of the problem (2D, 2D 1/2, 2D axisymmetric, 3D, etc.).
- Numerical parameters for the calculation (maximum number of solver iterations, initial solution, convergence criteria, etc.).
- For evolution calculations, the parameters controlling the continuation scheme (but not the dependence of each computed variable upon the evolution variable, which is defined for each sub-task).
- Specification of a steady-state, evolution, or time-dependent calculation.
- For time-dependent calculations, the parameters controlling the time-marching scheme (but not the time dependence of each variable or boundary condition, which is defined for each sub-task); for example, the initial and final time values are task attributes, but the time dependence of the flow rate along a particular boundary is a sub-task attribute.
- Coupling of sub-tasks. Complex problems can involve several sub-tasks, coupled or uncoupled. By default, all sub-tasks of a given task are assembled into the same system of algebraic equations and solved simultaneously. If coupling is not required, you can request that the sub-tasks be solved sequentially.

For most cases, you will define only one task, but there are cases in which more than one task is required. For example, in an evolution calculation where the initial solution is different from the default initial solution generated by ANSYS POLYDATA, you will need to generate two tasks: a steady-state task for generating the initial solution, and an evolution task that starts from the initial solution computed by the steady-state task.

1.4.1.2. Sub-Tasks

A sub-task consists of the following parameters:

- models or equations being solved (fluid flow, heat transfer, etc.)
- domain of definition (i.e., the region where the sub-task applies)
- model parameters (relaxation times, etc.)
- boundary conditions (inlet flow rate, wall temperature, etc.)
- material properties (viscosity, thermal conductivity, etc.)
- finite element interpolation
- for evolution calculations, the dependence of each computed variable on the evolution variable

- for time-dependent calculations, the time dependence of each variable or boundary condition

For simple cases (e.g., flow of a homogeneous fluid through a contraction), define only a single sub-task. For others, you may need to define multiple sub-tasks. For example, if your problem involves more than one material, you will have to define a sub-task for each material, since only one set of material properties can be defined for a single sub-task.

As an example of a problem with multiple sub-tasks, consider the nonisothermal flow of a viscous fluid through an axisymmetric steel die (shown in *Figure 1.2* (p. 6)).

Figure 1.2 Flow through an Axisymmetric Die

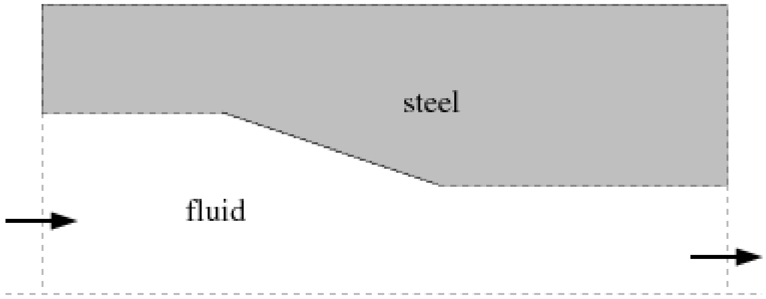
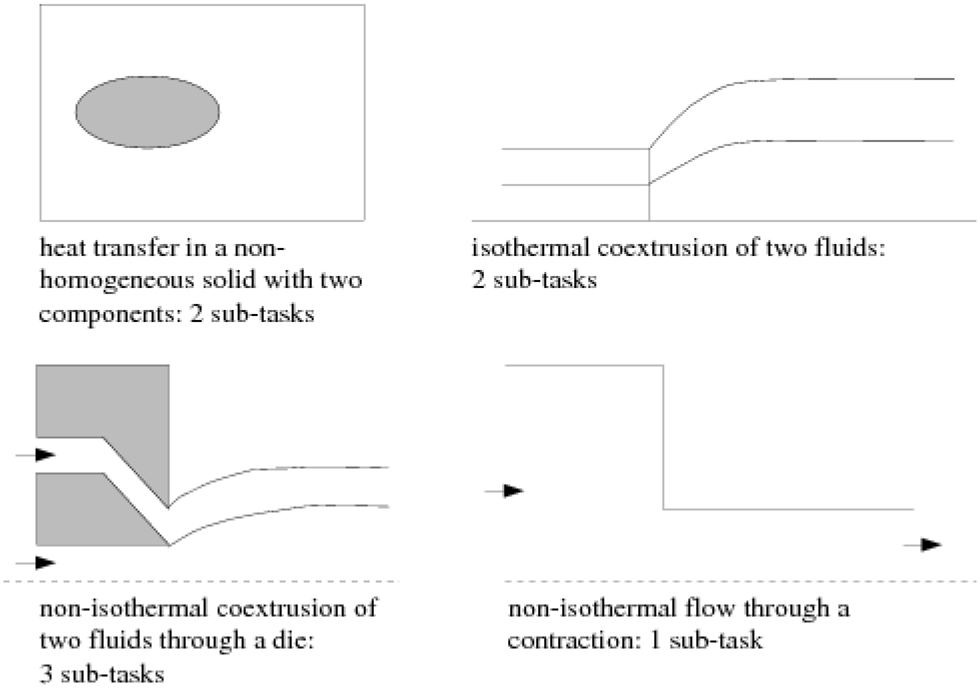


Figure 1.3 Sub-Tasks


heat transfer in a non-homogeneous solid with two components: 2 sub-tasks

isothermal coextrusion of two fluids: 2 sub-tasks

non-isothermal coextrusion of two fluids through a die: 3 sub-tasks

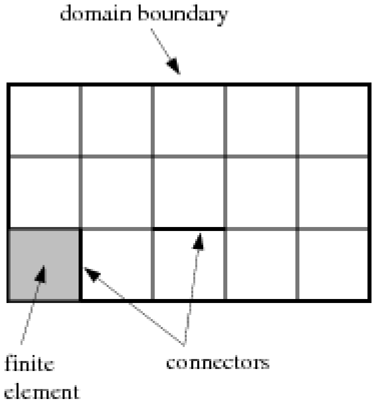
non-isothermal flow through a contraction: 1 sub-task

For the fluid region, you want to compute heat transfer and motion of the fluid. For the solid (steel) region, you want to compute heat transfer only. Each of these requires a different sub-task, because each will solve different equations and use different material properties. The two sub-tasks must be solved simultaneously, because the global solution depends on the values of the variables (e.g., temperature) at the interface between the two regions. *Figure 1.3* (p. 6) shows some other examples requiring different numbers of sub-tasks.

1.4.2. Subdomains and Boundary Sets

ANSYS POLYFLOW uses a finite element method (FEM) to solve partial differential equations that are defined for the flow domain, with boundary conditions specified on the domain boundary or a part of it (see *Figure 1.4* (p. 7)).

Figure 1.4 Finite-Element Definitions



Since it is often desirable to solve different sets of equations in different parts of the domain, and since the conditions on different parts of the boundary are usually not the same, the concepts of subdomains and boundary sets are introduced.

Subdomains and boundary sets can be defined in terms of the finite element technique itself. The flow domain is subdivided into finite elements, as shown in *Figure 1.4* (p. 7). The elements must not overlap, and the union of the elements is the domain itself. In 2D, the finite elements are quadrilaterals or triangles; in 3D, they are bricks, wedges, or tetrahedra. The finite elements define a network of element connectors, which are segments in 2D and faces in 3D. Connectors are shared either by two finite elements or by one element and the domain boundary.

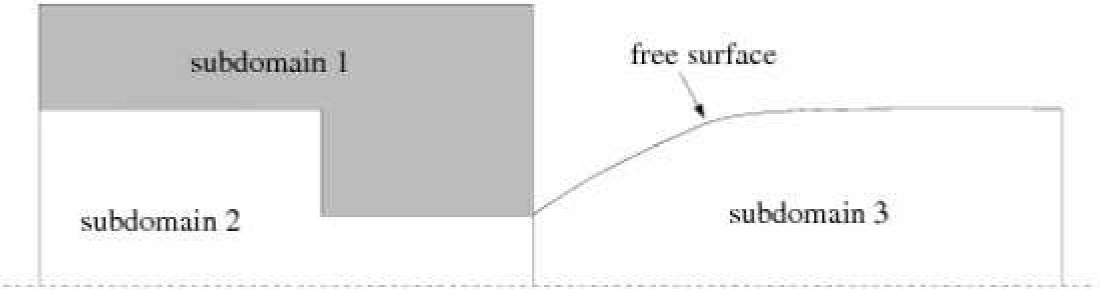
A *boundary set* is a group of connectors located on the domain boundary. Connectors that lie on the intersection between two subdomains are referred to as internal boundaries, and are not part of any boundary set.

A *subdomain* is a group of finite elements. The union of all subdomains is the original domain. Note that subdomains do not need to be contiguous. That is, a single subdomain can consist of several unconnected parts. The specification of subdomains occurs during mesh generation, and will therefore be done using GAMBIT or another preprocessor. *Meshes* (p. 133) contains some guidelines for defining subdomains in different preprocessors.

In most cases, you will need to define more than one subdomain for your model. A different subdomain is required for each sub-task, so, for example, you will need two subdomains if you have a fluid in one part of your domain and a solid material in another. Some of ANSYS POLYFLOW's features (such as the calculation of free surfaces) will also require multiple subdomains.

As an example of a problem requiring multiple subdomains, consider the 2D axisymmetric jet-like free-surface problem shown in *Figure 1.5* (p. 8).

Figure 1.5 Subdomains for a Simple Free-Surface Problem



A heat conduction problem is solved in the solid region (subdomain 1), and a fluid flow problem is solved in the fluid region (subdomains 2 and 3). Two sub-tasks are required because of the two different materials and sets of equations being solved, but the domain is divided into three subdomains.

The division of the fluid region into subdomains 2 and 3 is required because of the free surface of the jet. When ANSYS POLYFLOW computes a free surface, it automatically remeshes the subdomain bordering the free surface. Since only one portion of the fluid region is adjacent to the free surface, the fluid region must be divided into two subdomains: one for the fixed section (subdomain 2) and one for the jet section (subdomain 3), which borders the free surface. The remeshing will be defined only for subdomain 3. The domain of definition for the heat conduction sub-task will be subdomain 1, and the domain of definition for the fluid flow sub-task will be the union of subdomains 2 and 3.

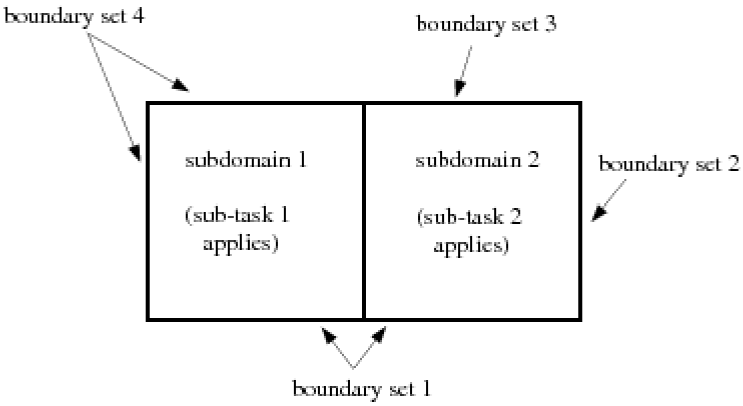
There is no limit on how many subdomains you can have in your model, but you should only define as many as necessary, to make the selection of subdomains in ANSYS POLYDATA easier. There are several operations that will require you to specify certain subdomains or unions of subdomains, and these operations can be very tedious if you have an unnecessarily large number of subdomains to choose from.

1.4.3. Boundary Conditions

In order to calculate a solution, ANSYS POLYFLOW requires data at the boundaries of each sub-task's domain of definition. This means that boundary conditions will be required both along external boundaries (boundary sets) and along internal boundaries between subdomains that belong to different sub-tasks. For the example shown in *Figure 1.5* (p. 8), which has three subdomains and two sub-tasks, boundary conditions will be required on the intersection between subdomains 1 and 2, but not on the intersection between subdomains 2 and 3, since they comprise the domain of definition for the same sub-task.

When you are defining your sub-task, ANSYS POLYDATA will request input only for the appropriate boundary conditions. Consider the problem shown in *Figure 1.6* (p. 9), where the boundary sets have been defined in the preprocessor.

Figure 1.6 Boundary Sets

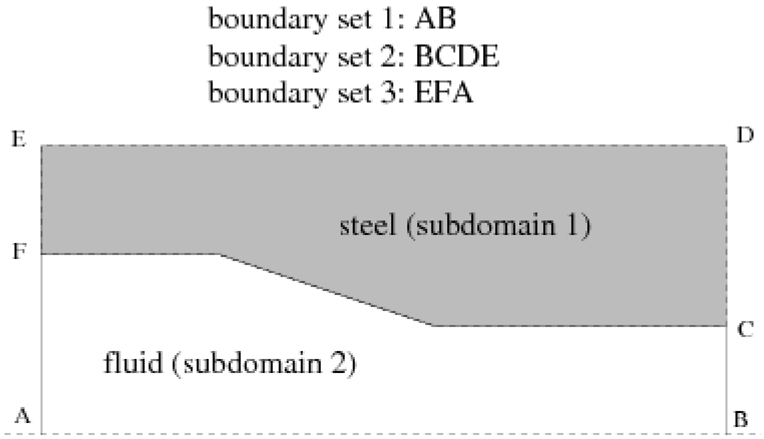


When you define boundary conditions for sub-task 1, ANSYS POLYDATA will request input for boundary sets 1 and 4, as well as for the intersection with subdomain 2. Your specification for boundary set 1 will apply to the intersection of boundary set 1 with subdomain 1. You need not divide boundary set 1 into two different boundary sets in your mesh generator because ANSYS POLYDATA interprets your inputs properly without this division.

When you define boundary conditions for sub-task 2, ANSYS POLYDATA will request input for boundary sets 1, 2, and 3, and for the intersection with subdomain 1. This time, the specification for boundary set 1 will apply to the intersection of boundary set 1 with subdomain 2.

When you are generating your mesh, you will need to identify boundary sets for the external boundaries where you intend to apply different boundary conditions. The union of all boundary sets must cover the entire outer boundary of the domain. For the die problem discussed in *Tasks and Sub-Tasks* (p. 5), the boundary sets are shown in *Figure 1.7* (p. 9).

Figure 1.7 Boundary Sets for the Axisymmetric Die of*Figure 1.2* (p. 6)



Note that boundary set 2 includes both CD and DE. This means that the boundary conditions on both segments are the same (e.g., heat flux=0). If you wanted to assign different boundary conditions on CD and DE, you would need to put each segment in a different boundary set.

As discussed earlier, ANSYS POLYDATA will determine for you which boundary sets require boundary condition specification. For the example shown in *Figure 1.7* (p. 9), ANSYS POLYDATA will request input of boundary conditions for sub-task 1 as follows:

- boundary set 2 (CDE only)
- boundary set 3 (EF only)
- interface between subdomains 1 and 2 (FC)

For sub-task 2, the requested inputs will be as follows:

- boundary set 1 (AB)
- boundary set 2 (BC only)
- interface between subdomains 2 and 1 (CF)
- boundary set 3 (FA only)

As mentioned earlier, it is not necessary to associate the limits of boundary sets with subdomain boundaries. Boundary conditions for both sub-tasks are assigned on boundary sets 2 and 3. For sub-task 1, ANSYS POLYDATA applies the specified conditions on CDE and EF, and for sub-task 2, it applies the specified conditions on BC and FA.

For each boundary set, ANSYS POLYDATA will offer default boundary conditions. When you define boundary conditions for each sub-task, you will be able to change the type of boundary condition and the parameters associated with the specified type (e.g., select an inflow type and specify the flow rate). Information about boundary conditions for different models is provided in the chapters about those models.

1.5. Starting ANSYS POLYDATA and ANSYS POLYFLOW

The installation process (described in the separate installation instructions for your computer type) is designed to ensure that the requested program is launched when you follow the instructions below. If it is not, consult your computer systems manager or your technical support engineer.

On Linux systems, all programs can be started by typing the appropriate command from the command line of an xterm window. On Windows systems, you can type the appropriate command in an MS-DOS Command Prompt window.

1.5.1. Starting ANSYS POLYDATA

To start ANSYS POLYDATA, type

```
polydata
```

It is also possible to start ANSYS POLYDATA without the graphical user interface described in *ANSYS POLYDATA Graphical User Interface* (p. 17), for cases when you are running ANSYS POLYDATA over a slow computer line, for example. To start a text-menu version of ANSYS POLYDATA, type

```
polydata -MENUS
```

You can also start ANSYS POLYDATA from ANSYS POLYMAN, as described in *Starting the Programs* (p. 40).

1.5.1.1. Starting Application-Specific Versions of ANSYS POLYDATA

To start the blow molding application-specific version of ANSYS POLYDATA, type

```
polydata -BLOWMOLDING
```

To start the extrusion application-specific version of ANSYS POLYDATA, type

```
  polydata -EXTRUSION
```

1.5.2. Starting ANSYS POLYFLOW

To start ANSYS POLYFLOW, type

```
 polyflow < my.dat
```

where `my.dat` is the name of the data file you saved at the end of your ANSYS POLYDATA session. (The mesh file and the data file must both be in the directory where you start ANSYS POLYFLOW.)

If you want to keep the listing that records the ANSYS POLYFLOW session so that you can review it at a later time (rather than viewing it as it scrolls past on your screen during the ANSYS POLYFLOW session), you can type

```
 polyflow < my.dat > my.lst
```

where `my.lst` is the name of the listing file. See *Writing an ANSYS POLYFLOW Listing File* (p. 110) for information about listing files.

You can use the `-opt msh` command line option to use an already-optimized mesh in the ANSYS POLYFLOW calculation, or the `-opt` *number_of_sub-parts* command line option to specify the number of sub-parts for mesh decomposition. See *Using Optimized or Decomposed Mesh* (p. 149) or *Specifying the Number of Sub-Parts for Decomposition* (p. 149) for details.

You can also start ANSYS POLYFLOW from ANSYS POLYMAN, as described in *Starting the Programs* (p. 40).

1.5.2.1. Starting Application-Specific Versions of ANSYS POLYFLOW

To start the blow molding application-specific version of ANSYS POLYFLOW, type

```
 polyflow -BLOWMOLDING  < my.dat
```

where `my.dat` is the name of the data file you saved at the end of your ANSYS POLYDATA session. (The mesh file and the data file must both be in the directory where you start the application-specific version of ANSYS POLYFLOW.)

To start the extrusion application-specific version of ANSYS POLYFLOW, type

```
 polyflow -EXTRUSION <  my.dat
```

where `my.dat` is the name of the data file you saved at the end of your ANSYS POLYDATA session. (The mesh file and the data file must both be in the directory where you start the application-specific version of ANSYS POLYFLOW.)

1.6. The .p3rc Configuration File

Information about your software password, the user interface, and memory management for ANSYS POLYFLOW is provided in a configuration file called the `.p3rc` file. This file contains a list of keywords and their corresponding values. If necessary, you can customize the running of ANSYS POLYFLOW by modifying the `.p3rc` file.

Setting Up Your .p3rc Files (p. 12) explains how to set up your `.p3rc` files, and *Keywords in the .p3rc File* (p. 12) describes the keywords. See also *Setting Options for ANSYS POLYDATA, ANSYS POLYFLOW, FieldView & CFD-Post* (p. 43) for information about modifying the `.p3rc` file through ANSYS POLYMAN.

1.6.1. Setting Up Your .p3rc Files

When you start ANSYS POLYFLOW (or one of its modules), it will look in different files on the system in order to determine if a keyword has been defined. On Linux systems, it first looks in your personal `.p3rc` file located in your home directory, if such a file exists. On Windows systems, it looks for a file named `.p3rc` in the directory that you have defined with the `%HOME%` variable.

If it does not find all the keywords in your personal `.p3rc` file and you have not redefined the `$POLYFLOW` variable, it will look in the following file:

- For Linux:

  path/ansys_inc/v140/polyflow/polyflow14.0.*x*/.p3rc

- For Windows:

  path\ANSYS Inc\v140\polyflow\polyflow14.0.*x*\ARCH\\.p3rc

where *path* is the directory in which ANSYS POLYFLOW has been installed, *x* represents the appropriate number for the release (e.g., `0` for `polyflow14.0.0`), and *ARCH* is the architecture (e.g., `ntx86`). However, if you have redefined the `$POLYFLOW` variable, it will instead look in `$POLYFLOW/.p3rc`.

Note that it is possible to have ANSYS POLYFLOW or ANSYS POLYDATA access a personal `.p3rc` file located somewhere other than your home directory. To look for keywords in a file called `mystartupfile`, use the following command line option:

```
polyflow -s mystartupfile
```

or

```
polydata -s mystartupfile
```

It is recommended that you keep information that is shared by all users (license-related keywords) in the standard (common) `.p3rc` file in the ANSYS POLYFLOW installation area that is accessed by all users. User-specific information (interface and memory management keywords) can be placed in a personal `.p3rc` file located in each user's home directory.

Important

Note that, if a keyword is defined in more than one `.p3rc` file, the one that ANSYS POLYFLOW finds first (based on the search procedure described above) is the one that will be used.

1.6.2. Keywords in the .p3rc File

1.6.2.1. License-Related Keywords

The `.p3rc` file contains the following keyword:

```
FLEXLM  <optional filename>
```

If a filename is not given, it is mapped to the standard license file `license.dat` in the following directory:

- For Linux:

`path/ansys_inc/v140/polyflow/polyflow14.0.x/license`

- For Windows:

  `path\ANSYS Inc\v140\polyflow\polyflow14.0.x\license`

where `path` is the directory in which ANSYS POLYFLOW has been installed and *x* represents the appropriate number for the release (e.g., `0` for `polyflow14.0.0`).

1.6.2.2. Interface-Related Keywords

The following keyword related to the appearance of the graphical user interface is also provided in the `.p3rc` file:

```
  INTERFACE  <type>
```

where the interface `<type>` can be `MOTIF` (the default) or `MENUS`.

1.6.2.3. Solver and Memory-Management Keywords

The following keywords related to the solver and memory are also included in the `.p3rc` file:

```
BLAS  <level>
BLOCS  <nn>
MAXMEM  <mm>
```

where `<level>` can be `LEVEL1`, `LEVEL2`, or `LEVEL3`, and `<nn>` is the size of the blocks in the BLAS3 solver. See *Using the Solver* (p. 575) for details. The entry `<mm>` represents the maximum size allowed for in-core factor matrices, in megabytes. Factor matrices can be large, but are efficiently stored on disk. By default, ANSYS POLYFLOW will take all the virtual memory available to store the matrix. In the event that a shortage of memory occurs, ANSYS POLYFLOW will automatically write the LU matrix to the disk, in a directory specified by the `$TMPDIR` environment variable (on Linux systems) or the `%TMP%` variable (on Windows systems).

If you specify a value (in megabytes) for the `MAXMEM` keyword, ANSYS POLYFLOW will never use more memory than specified for the factor matrix. This allows you to save some memory in a multiple-user environment, for example.

1.6.2.4. Example

This example reverts to the settings of the POLYFLOW 3.7 release:

```
# An alternate location of the license file.
FLEXLM /nfs/myfiles/pfl/polyflow/polyflow14.0.0/license/license.dat
# The (much) slower Blas1, sequential elimination solver.
BLOCS 1
BLAS LEVEL1
# No buffers in memory if larger than 10 MB!
MAXMEM 10
# for polydata C++ and the MENUS interface, not the MOTIF!
INTERFACE MENUS
```

1.7. ANSYS POLYFLOW Documentation

Documentation concerning ANSYS POLYFLOW is available in the ANSYS help viewer, as PDF files in the installation area, on the ANSYS Customer Portal, in the **Help** tab in ANSYS POLYDATA, and in the help topic panel in ANSYS POLYMAT.

You can access most of the POLYFLOW documentation through the ANSYS help viewer. All of the POLYFLOW manuals available via the help viewer are also available as PDF files in your ANSYS, Inc. installation area. Note that the POLYSTAT User's Guide and the Examples Manual are only available as PDF files in the installation area, and the Tutorial Guide is only available on the ANSYS Customer Portal (`www.ansys.com/customerportal`).

1.7.1. Accessing the Documentation Files Using the ANSYS Help Viewer

The ANSYS help viewer provides access to documentation for most ANSYS products. The help viewer can be opened from the POLYFLOW user interface or from the Windows **Start** menu. To open it from the POLYFLOW user interface, use the **Help** pull-down menu in the various POLYFLOW applications or the **Help** chart button in POLYMAT and POLYCURVE. To open it from the Windows **Start** menu, select the following:

Start > All Programs> ANSYS 14.0 > Help > ANSYS Help 14.0

After the help viewer is open you can navigate to POLYFLOW documentation by performing the following steps:

1. Scroll down to **POLYFLOW** in the **Contents** tab of the panel on the left.
2. Expand the POLYFLOW documentation set by clicking the icon next to **POLYFLOW**.
3. Click on a document title to display the table of contents for the selected document.
4. To find specific information, you can do any of the following:
   a. In the **Contents** tab, click an icon next to a title to expand the tree hierarchy, or click an item in the tree hierarchy to display the corresponding information.
   b. In the **Search** tab, you can search for keywords or section titles in all or selected ANSYS manuals.

      Note

      Search results can be sorted by clicking on the column title.

   c. In the **Index** tab, type an index string.

      Note

      Index entries that partially match your typing will be displayed. You can continue typing, or scroll through the list to find specific entries.

1.7.2. Accessing the PDF Documentation Files in the Installation Area

PDF files of most of the POLYFLOW manuals can be accessed by using the **Help** pull-down menu of POLYDATA, POLYMAN, and POLYSTAT. You can also access these PDF files by opening the following folder/directory. Note that the file names of all of the POLYFLOW manuals begin with **poly_**.

For Windows:

path`\ANSYS Inc\v140\commonfiles\help\en-us\pdf\`

where *path* is the folder in which you have installed POLYFLOW (by default, the path is `C:\Program Files`).

For Linux:

path`/ansys_inc/v140/commonfiles/help/en-us/pdf/`

where *path* is the directory in which you have installed POLYFLOW.

The Examples Manual can be accessed by pointing your web browser to

- For Windows:

  path`\ANSYS Inc\v140\polyflow\polyflow14.0.`*x*`\help\index.htm`
- For Linux:

  path`/ansys_inc/v140/polyflow/polyflow14.0.`*x*`/help/index.htm`

where *path* is the directory in which ANSYS POLYFLOW has been installed and x represents the appropriate number for the release (e.g., `0` for `polyflow14.0.0`).

If, for example, you are using Internet Explorer as your browser, select the **File** > **Open...** menu item and click the **Browse** button to browse through your directories to find the file. When opened, the file displays the ANSYS POLYFLOW documentation “home" page; this page provides a link to the Examples Manual page, which allows you to view individual examples or download a zipped file of all of the examples.

1.7.3. Viewing and Printing the PDF Documentation

The Adobe Acrobat PDF files that are provided can be used for viewing and printing all or part of the manuals using Adobe Reader, which is available for most Linux and Windows systems. These files are distinguished by a `.pdf` suffix in their file names. While you can also print individual topics from the ANSYS help viewer, the PDF files are recommended when printing long sections because the printout will have a higher quality.

If you do not have Adobe Reader, you can download it (at no cost) from http://www.adobe.com. If you are not able to download this software, contact your support engineer and ask for the Adobe Reader CD-ROM.

1.7.3.1. Navigating the PDF Files

For the purpose of easier online document navigation, the PDF files contain hyperlinks in the table of contents and index. In addition, there are hyperlinks for all cross references to chapters, sections, figures, tables, bibliography, and index entries.

1.7.3.2. Printing the PDF Files

Note that you can select the paper size to which you are printing in Adobe Reader by selecting the **File/Print Setup...** menu item and choosing the desired **Paper** size. If the page is too large to fit on your paper size, you can reduce it by selecting the **File/Print...** menu item and then selecting **Reduce to Printer Margins** from the **Page Scaling** drop-down list.

Chapter 2: ANSYS POLYDATA Graphical User Interface

This chapter contains information about the ANSYS POLYDATA user interface. Since ANSYS POLYFLOW runs based on input from the data file created by ANSYS POLYDATA, there is no user interface for ANSYS POLYFLOW itself.

This chapter contains the following sections:

2.1. ANSYS POLYDATA GUI

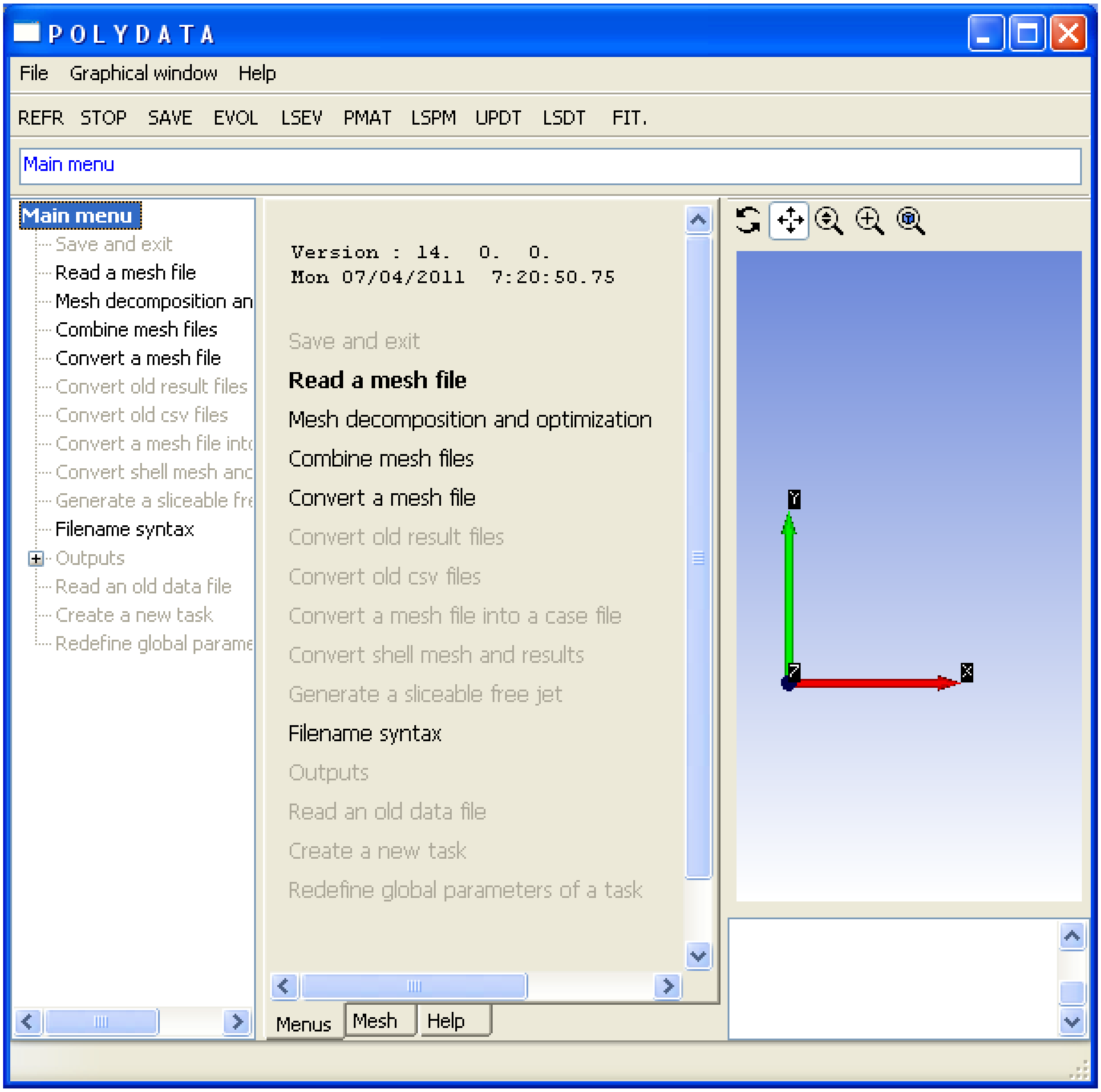
When you start ANSYS POLYDATA, the main window will appear (*Figure 2.1* (p. 18)). The ANSYS POLYDATA main window consists of several components, each of which serves a separate purpose.

These components are:

- Menu bar
- Text window
- Keywords
- The tree window
- ANSYS POLYDATA main menu
- **Graphics Display** window
- Output text window

After you read in a mesh file, the geometry of the mesh is displayed in the **Graphics Display** window (*Figure 2.10* (p. 24)).

Figure 2.1 ANSYS POLYDATA Graphical User Interface (GUI)



```
Version : 14.  0.  0.
Mon 07/04/2011  7:20:50.75
```

You can modify the size of the tabs, the **Graphics Display** window, and the lower text window. *Figure 2.1* (p. 18) shows these components of the ANSYS POLYDATA interface. The six different components are described in detail in the following sections.

2.2. Menu Bar

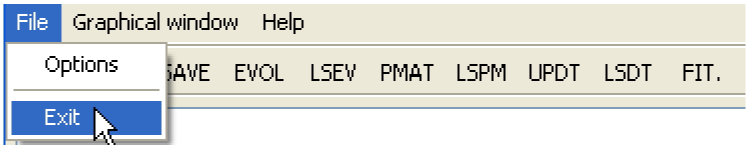
The menu bar organizes the GUI menu hierarchy using a set of pull-down menus. A pull-down menu contains items that perform commonly executed actions and allows you to modify the parameters of the graphical area.

Figure 2.2 Menu Bar

2.2.1. File

The **File** pull-down menu allows you to change the font for certain text in the GUI and to exit ANSYS POLYDATA.

Figure 2.3 File Pull-down Menu



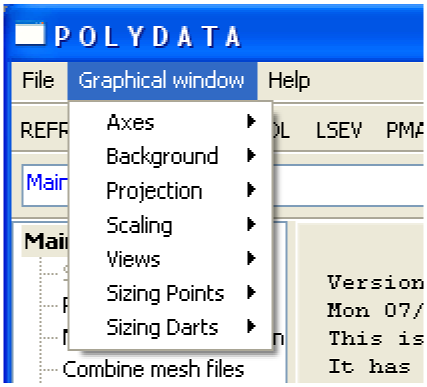
Options
opens the **Polydata options...** dialog box, which you can use to modify the font used for the comments (i.e., the notes at the top of the menu), the options (i.e., the menu items), the tree (i.e., the tree view window), and the log area (i.e., the output text window).

Exit
closes the POLYDATA application.

2.2.2. Graphical Window

The **Graphical Window** menu allows you to modify the parameters of the **Graphics Display** window.

Figure 2.4 Graphical Window Pull-down Menu



Axes
allows you to display/erase the axes of the coordinates system.

Background
modifies the background color.

Projection
offers the choice between Perspective and Orthographic projections.

Scaling
allows non-isotropic zooming.

Views
aligns the display with one of the coordinates system axis.

Sizing Points
allows you to increase/decrease the size of points appearing in the **Graphics Display** window. Such points are displayed when defining a Probe in the Outputs menu (see *Saving Data at a Specified*

Point (p. 124) for more information) or a transformation method in a mesh deformation preprocessor (see *Die Shape Parameterization* (p. 619) for more information). As the default size of those points is automatically set according to the dimensions of the geometry, this feature is sometimes useful to change their size in the **Graphics Display** window.

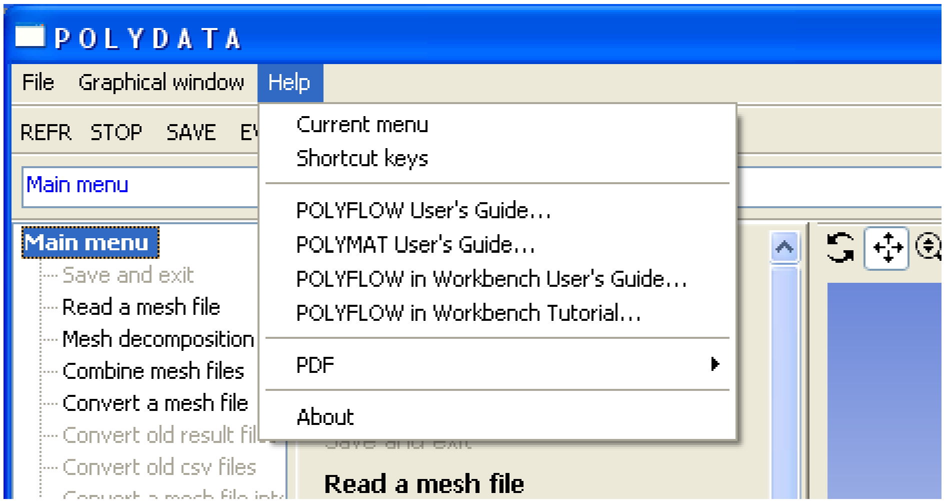
Sizing Darts

allows you to increase or decrease the size of darts in the **Graphics Display** window. The darts are displayed automatically when required by the setup, such as when you click **Specify mold side / cavity side** in the **Modification of a contact problem** menu when setting up a contact detection problem for a shell task, or when you define the settings in the **Diffuse gray wall parameters** menu for an internal radiation problem. The size of the dart is initially based on the dimensions of the geometry.

2.2.3. Help

The **Help** pull-down menu gives you access to information that can help you use POLYFLOW.

Figure 2.5 Help Pull-down Menu



Current menu

opens the **Help** tab to display information about the current ANSYS POLYDATA menu. See *Help Tab* (p. 24) for further details.

Shortcut keys

shows the shortcuts of the **Graphics Display** window.

POLYFLOW User's Guide...

opens the ANSYS help viewer to the table of contents of the POLYFLOW User's Guide.

POLYMAT User's Guide...

opens the ANSYS help viewer to the table of contents of the POLYMAT User's Guide.

POLYFLOW in Workbench User's Guide...

opens the ANSYS help viewer to the table of contents of the POLYFLOW in Workbench User's Guide.

POLYFLOW in Workbench Tutorial...

opens the ANSYS help viewer to the table of contents of the POLYFLOW in Workbench Tutorial.

PDF

opens a submenu, that allows you to access PDF files of the POLYFLOW User's Guide, POLYMAT User's Guide, POLYSTAT User's Guide, POLYFLOW in Workbench User's Guide, and POLYFLOW in Workbench Tutorial. The PDF files will be displayed in Adobe Reader, which you can download it (at no cost) from http://www.adobe.com.

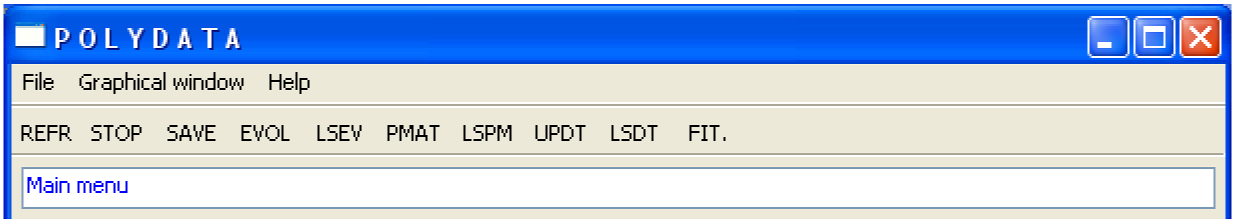
About

gives you information about the version of ANSYS POLYDATA and some details about its development.

2.3. Text Window

Under the list of keywords, there is a text window that contains the path to the current ANSYS POLYDATA menu.

Figure 2.6 Text Window

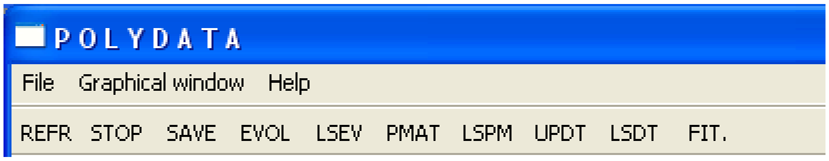


Main menu

2.4. Keywords

Under the Menu bar, there is a series of ten keywords.

Figure 2.7 Keywords



REFR

refreshes the display of the ANSYS POLYDATA menu.

STOP

stops ANSYS POLYDATA *without* saving the current session to a file.

SAVE

saves the current session to a session file without generating a complete data file. You can restart ANSYS POLYDATA and read in this file to continue your session (see *Reading and Writing ANSYS POLYDATA Session Files* (p. 108)), but you *cannot* read the session file into ANSYS POLYFLOW. The data file (see *Reading and Writing ANSYS POLYDATA Data Files* (p. 107) and *Reading Mesh, Data, and Results Files into ANSYS POLYFLOW* (p. 108)) is the way to transfer your problem definition to ANSYS POLYFLOW to calculate a solution.

EVOL

turns on or off the menu for specifying dependencies of solution variables on the evolution variable or on time. This menu will appear for each material property or boundary condition that you specify. See *Evolution* (p. 517) for details.

LSEV

lists all evolution or time dependencies that are currently defined, and allows you to edit or delete them.

PMAT

turns on or off the menu for specifying dependencies of material properties on a variable, such as pressure, temperature, or species. This menu will appear for each material property that you specify. See *Specifying Material Properties as Algebraic Functions* (p. 201) for details.

LSPM
lists all property dependencies that are currently defined, and allows you to edit or delete them.

UPDT
turns on the menu for flagging a parameter as modifiable in a User Defined Template (UDT). See *User-Defined Templates (UDTs)* (p. 615) for details.

LSDT
lists all templates entries that are currently defined, and allows you to edit or delete them.

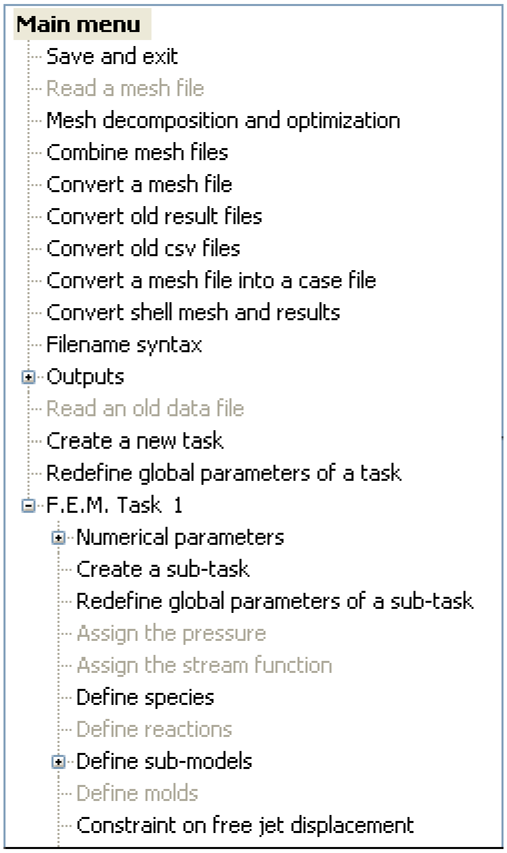
FIT.
starts ANSYS POLYMAT, a module that allows you to fit the parameters of selected fluid properties using experimental data.

You have to save the results of your work in ANSYS POLYMAT yourself and read them into ANSYS POLYDATA through a material data file, because ANSYS POLYDATA and ANSYS POLYMAT will not exchange material data directly. See the ANSYS POLYMAT User's Guide for details.

2.5. The Tree View Window

The Tree View Window displays the tree structure of the ANSYS POLYDATA menus.

Figure 2.8 The Tree View Window



It helps you to understand where the current menu is in the tree structure and how you can access other menus. The current menu is highlighted.

A 'plus' (+) sign in front of an entry in the tree indicates that it is actually the root of a subtree. Clicking on the 'plus' (+) sign will expand this subtree and the submenus will be displayed.

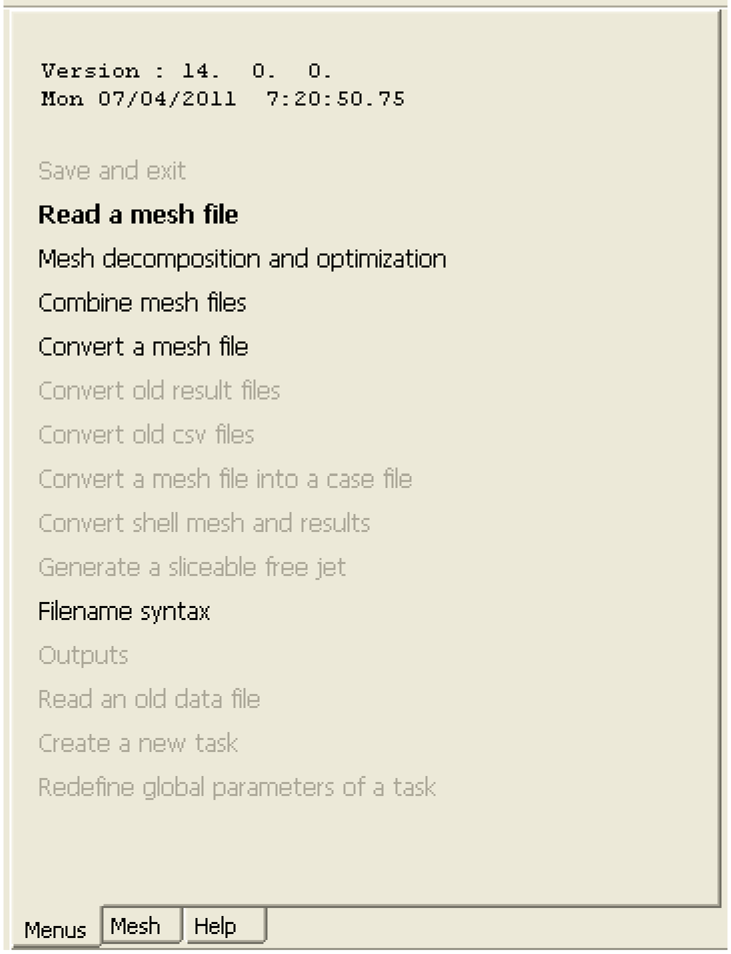
This tree is dynamic, that is, a double-click on an entry will open the corresponding menu. Such a jump in the tree of menus is not always possible. Some menus require the completion of the initiated procedure before you can move to another menu. In such a case, a double-click in the tree will be ineffective.

2.6. ANSYS POLYDATA Tabs

2.6.1. Menus Tab

The ANSYS POLYDATA **Menus** tab allows you to select the menu items used to set up your model. For each menu, ANSYS POLYDATA will highlight one menu item, suggesting that you select it next. In *Figure 2.9* (p. 23), since no mesh file has been read in yet, the **Read a mesh** file item is highlighted. You can choose a menu item other than the one that is highlighted. For example, if you are going to read a mesh file created by GAMBIT you will select **Convert a mesh** file instead. To select a menu item, click it once with your left mouse button.

Figure 2.9 ANSYS POLYDATA Menus Option

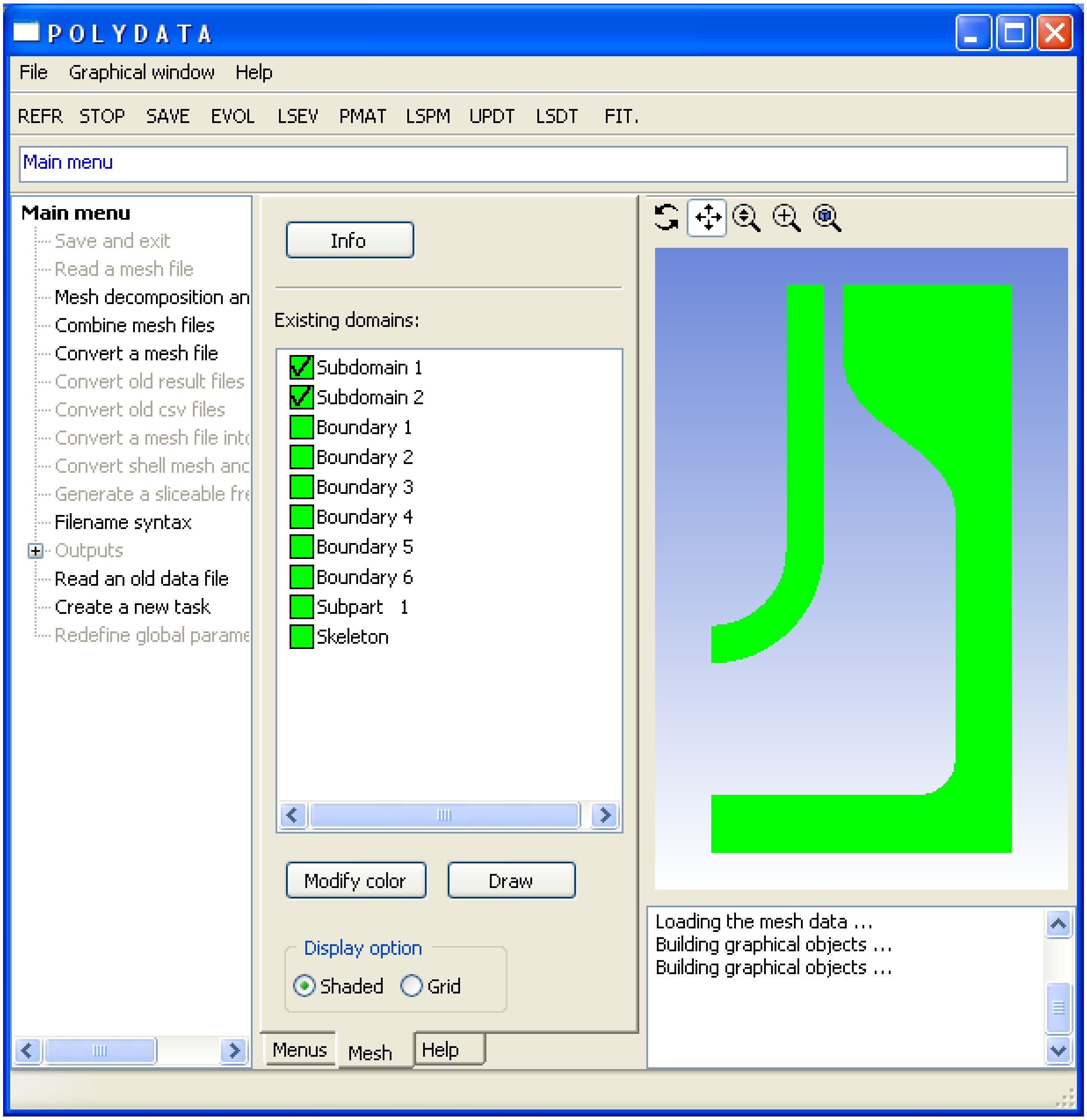


```
Version : 14.   0.   0.
Mon 07/04/2011   7:20:50.75
```

2.6.2. Mesh Tab

Click the **Mesh** tab of the main window to open the **Mesh Topology** panel (see *Figure 2.10* (p. 24)), which you can use to examine the mesh. This is useful for checking the mesh before you start defining your problem.

Figure 2.10 Mesh Topology Panel



Existing domains:

☑ Subdomain 1
☑ Subdomain 2
□ Boundary 1
□ Boundary 2
□ Boundary 3
□ Boundary 4
□ Boundary 5
□ Boundary 6
□ Subpart 1
□ Skeleton

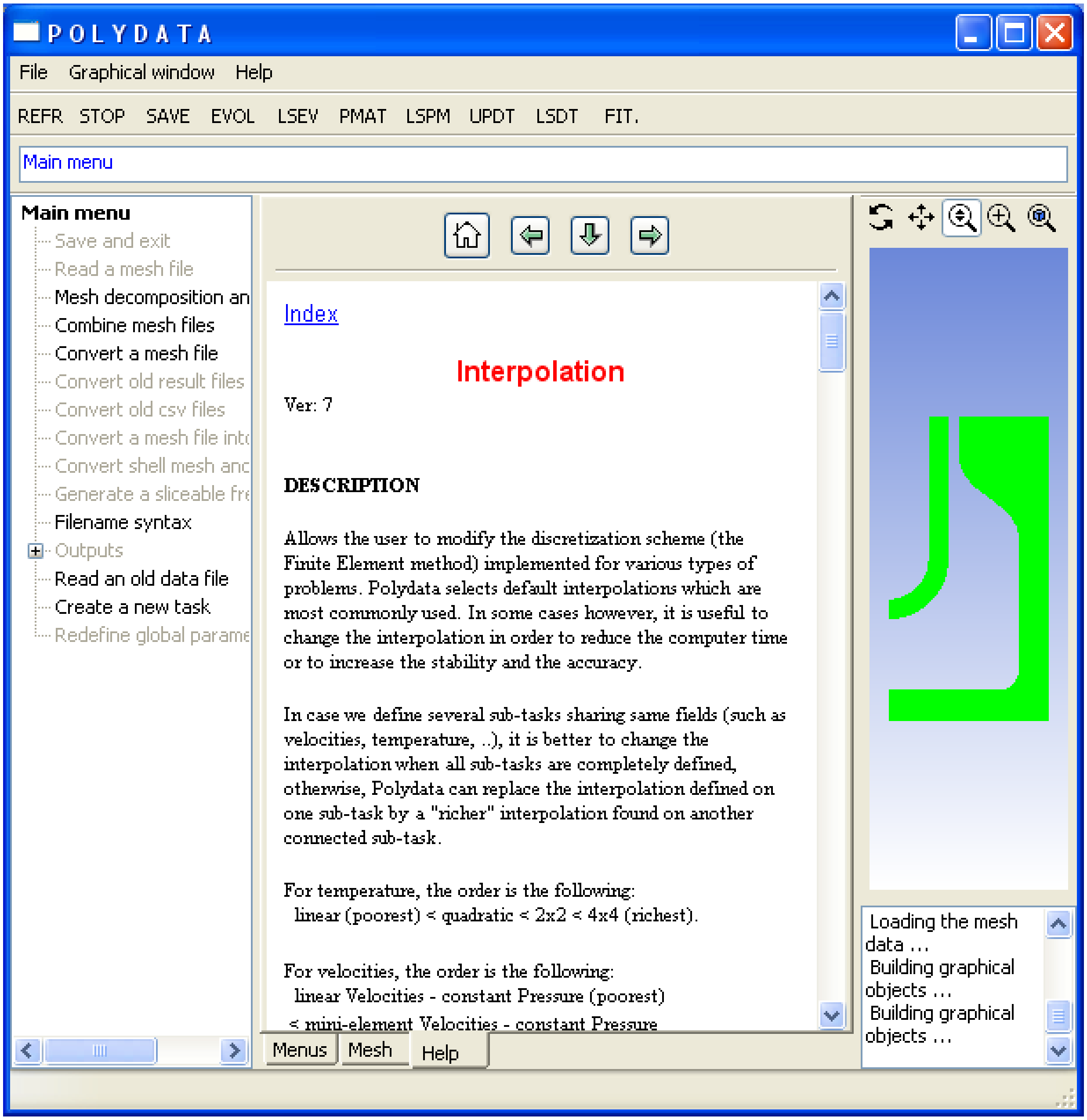
Display option: ● Shaded ○ Grid

```
Loading the mesh data ...
Building graphical objects ...
Building graphical objects ...
```

2.6.3. Help Tab

The **Help** tab (*Figure 2.11* (p. 25)) provides information about the POLYDATA menus. This tab is automatically opened to information about the current menu when you select the **Current menu** option from the **Help** pull-down menu. Buttons at the top of the tab allow you to browse the available information: the **Index** button (⌂) displays the index; the **Back** button (⇦) displays the previous page of your browsing history; the **Current** button (⇩) displays the page for the current menu; and the **Forward** button (⇨) displays the next page of your browsing history.

Figure 2.11 Help Tab



Index

Interpolation

Ver: 7

DESCRIPTION

Allows the user to modify the discretization scheme (the Finite Element method) implemented for various types of problems. Polydata selects default interpolations which are most commonly used. In some cases however, it is useful to change the interpolation in order to reduce the computer time or to increase the stability and the accuracy.

In case we define several sub-tasks sharing same fields (such as velocities, temperature, ..), it is better to change the interpolation when all sub-tasks are completely defined, otherwise, Polydata can replace the interpolation defined on one sub-task by a "richer" interpolation found on another connected sub-task.

For temperature, the order is the following:
linear (poorest) < quadratic < 2x2 < 4x4 (richest).

For velocities, the order is the following:
linear Velocities - constant Pressure (poorest)
< mini-element Velocities - constant Pressure

Loading the mesh data ...
Building graphical objects ...
Building graphical objects ...

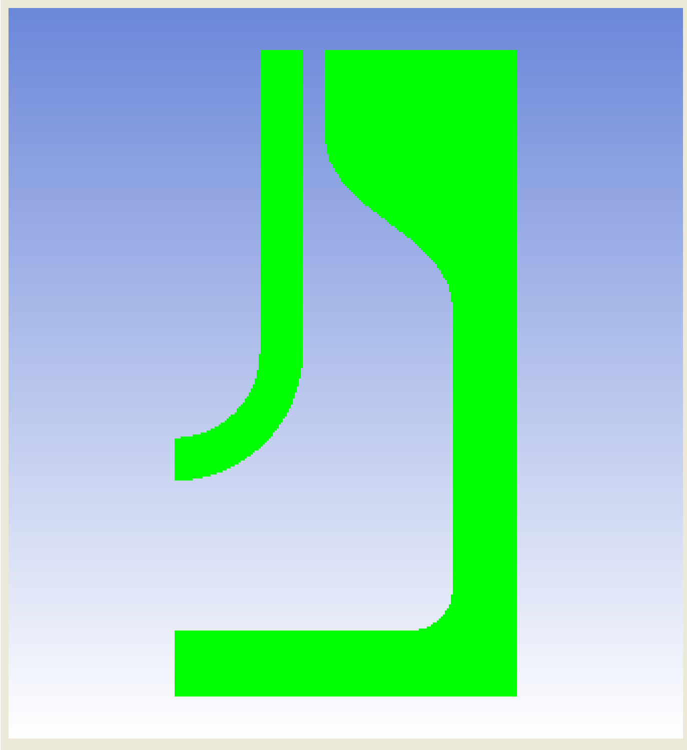
2.7. Graphics Display Window

The **Graphics Display** window (*Figure 2.12* (p. 26)) shows the mesh that you have read into ANSYS POLYDATA.

Depending on the menu item being used, the **Graphics Display** window will be updated to show relevant information. For example, when you are setting flow boundary conditions, the mesh outline will be shown and the selected boundary set will change color.

You can manipulate the colors and view in the **Graphics Display** window using your mouse or the controls in the **Graphics Display Options** menu. See *Examining the Mesh in ANSYS POLYDATA* (p. 160) for details.

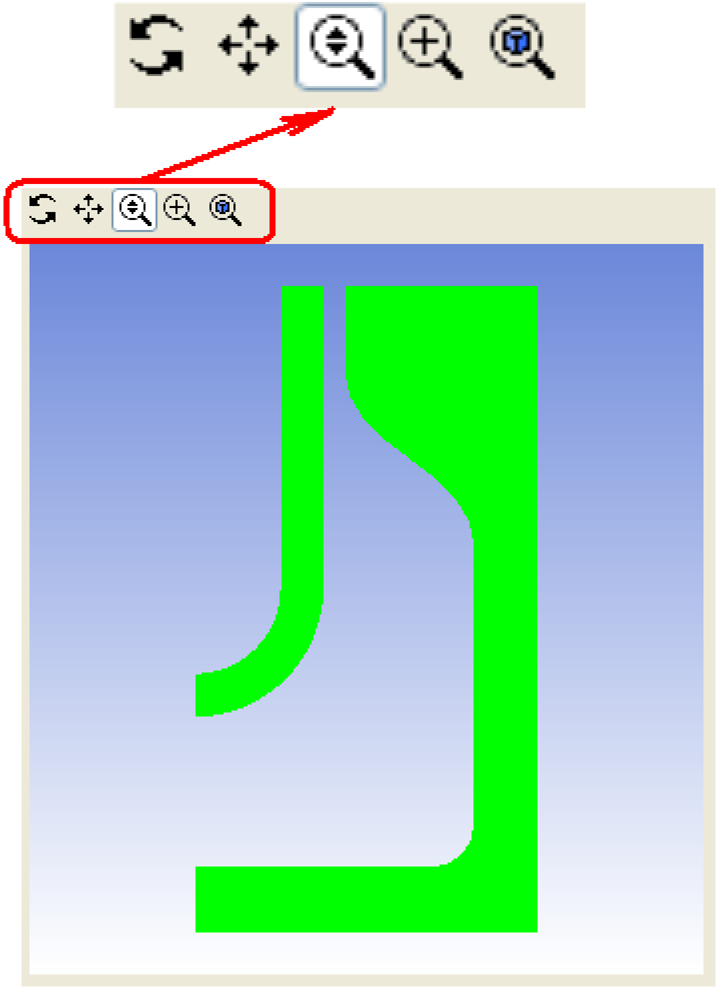
Figure 2.12 Graphics Display Window



2.8. Graphics Toolbar

The graphics toolbar contains a set of buttons (see *Figure 2.13* (p. 27)) that allows you to manipulate the view displayed in the **Graphics Display** window.

Figure 2.13 Graphics Display Window with Graphics Toolbar



You can perform any of the actions described in *Table 2.1: View Manipulation Instructions* (p. 27) (note that these descriptions are based on the default mouse button settings).

Table 2.1 View Manipulation Instructions

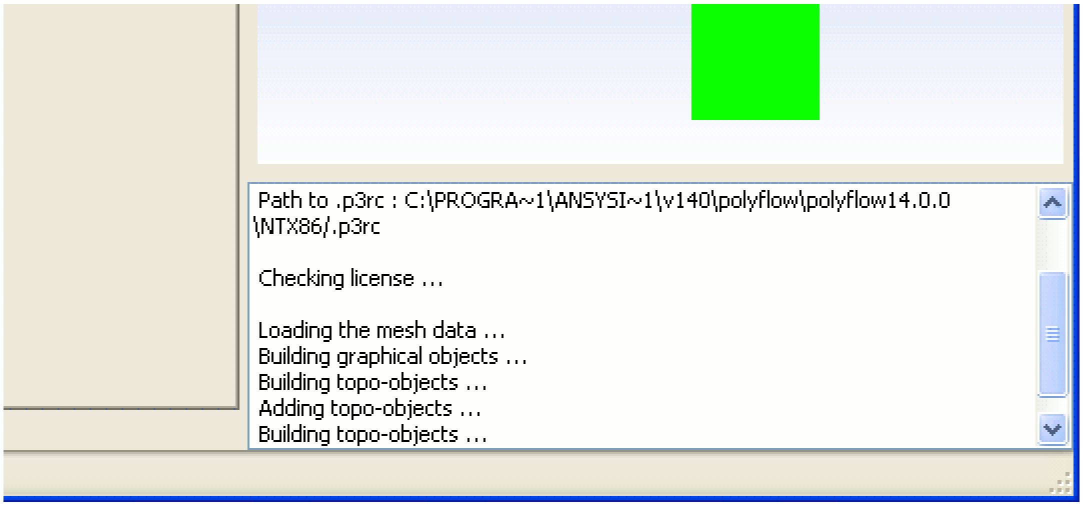
Action	Using Graphics Toolbar Buttons
Rotate view	Perform either of the following actions: - After clicking ⟳, press the left mouse button and drag the mouse. - After clicking any button in the graphics toolbar except for 🔍, press the middle mouse button and drag the mouse. Dragging the mouse side to side rotates the view about the vertical axis, and up and down rotates the view about the horizontal axis. If you hold the **Ctrl** key before you press the mouse button and drag, the rotation will be constrained to the plane of the view (i.e., rotation will be about the axis perpendicular to the view plane).
Translate view	After clicking ✥, press the left mouse button and drag the mouse.

Action	Using Graphics Toolbar Buttons
Zoom in on view	Perform one of the following actions: • After clicking 🔍, press the left mouse button and drag the mouse up. • After clicking 🔍, press the left mouse button and drag the mouse. This action will cause a rectangle to appear in the **Graphics Display** window. When you release the mouse button, a new view will be displayed which consists entirely of the contents of the rectangle. • After clicking any button in the graphics toolbar except for 🔍, press the right mouse button and drag the mouse. This action will cause a rectangle to appear in the **Graphics Display** window. When you release the mouse button, a new view will be displayed which consists entirely of the contents of the rectangle.
Zoom out from view	After clicking 🔍, press the left mouse button and drag the mouse down.
Fit to window	Click 🔍 to change the view so that all objects are visible and centered in the **Graphics Display** window.

2.9. Output Text Window

The output text window (*Figure 2.14* (p. 28)) contains information on the progress of ANSYS POLYDATA and possible warnings and errors.

Figure 2.14 Output Text Window



```
Path to .p3rc : C:\PROGRA~1\ANSYSI~1\v140\polyflow\polyflow14.0.0
\NTX86\.p3rc

Checking license ...

Loading the mesh data ...
Building graphical objects ...
Building topo-objects ...
Adding topo-objects ...
Building topo-objects ...
```

Chapter 3: Managing ANSYS POLYFLOW Projects Using ANSYS POLYMAN

ANSYS POLYMAN is an environment layer built on top of the programs used in the ANSYS POLYFLOW package. It allows you to start GAMBIT, ANSYS POLYDATA, ANSYS POLYFLOW, FieldView, CFD-Post, and other modules (ANSYS POLYMAT, ANSYS POLYDIAG, ANSYS POLYFUSE, ANSYS POLYSTAT) from a single starting point. Furthermore, this POLYFLOW manager helps you organize all of the cases within a given analysis by automatically creating new directories for new simulations. Moreover, ANSYS POLYMAN keeps track of the creation and modification dates of any files, and warns you if a file becomes obsolete (due to changes in the related files) in order to maintain the coherence of the files.

Information about ANSYS POLYMAN is provided in the following sections:

An objective of ANSYS POLYMAN is to hide the numerous files involved in any simulation. Also, the concepts of a project (analysis of a given case that could involve different meshes and/or geometries) and a simulation (a given set of material properties and operating conditions) are created.

3.1. Starting ANSYS POLYMAN

To start ANSYS POLYMAN, type:

```
polyman
```

at the command line prompt (in Windows, an MS-DOS Command Prompt Window). On Windows systems, you can also start ANSYS POLYMAN by clicking the **Start** button, clicking **All Programs**, selecting **ANSYS 14.0**, selecting **Fluid Dynamics**, and clicking **POLYFLOW 14.0**; alternatively, you can double-click the ANSYS POLYMAN icon in the following folder:

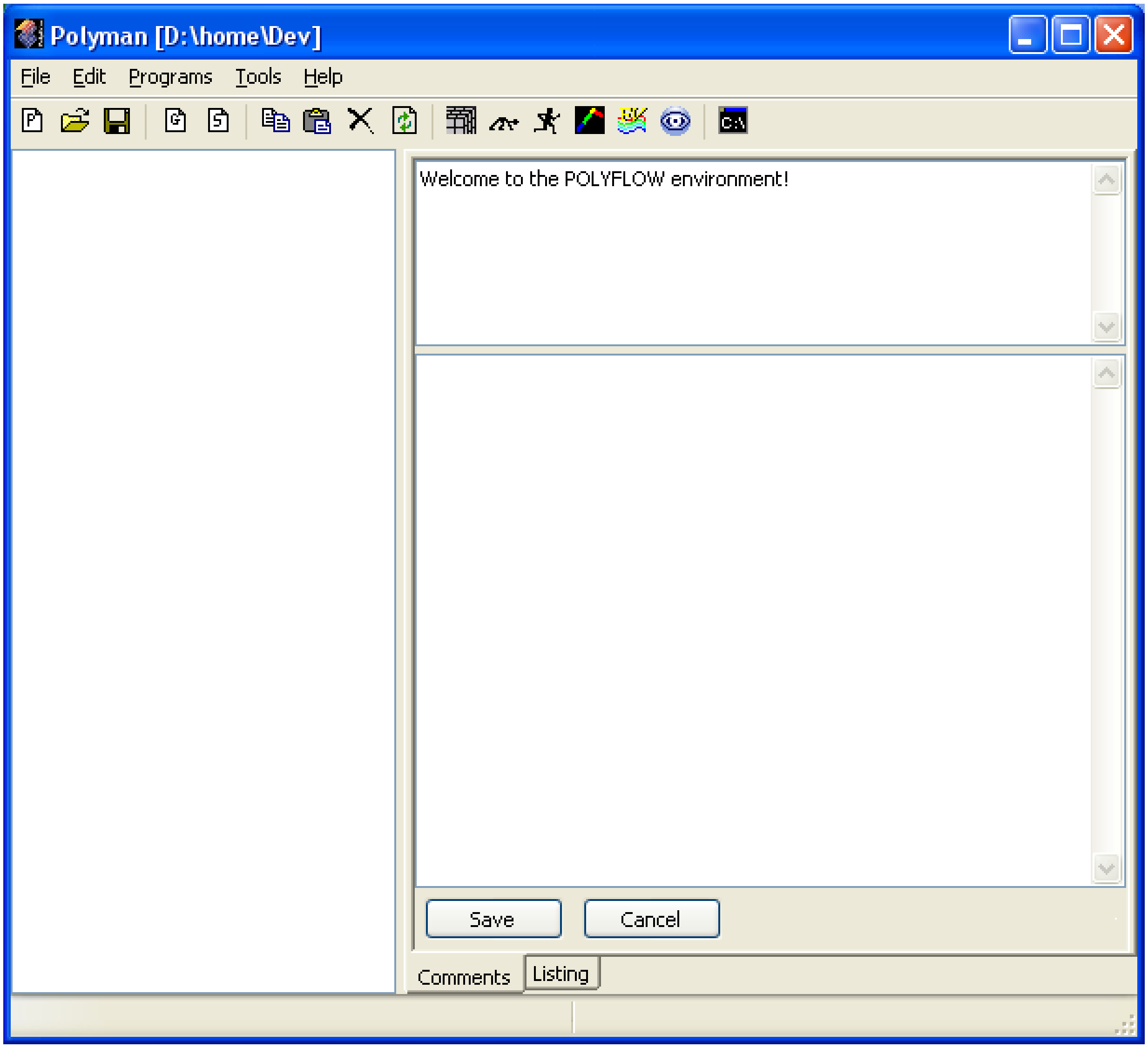
```
path\ANSYS Inc\v140\polyflow\ntbin\ntx86
```

where *path* is the folder in which ANSYS POLYFLOW has been installed.

3.2. Overview of the ANSYS POLYMAN Interface

When you first start ANSYS POLYMAN, the interface will appear as shown in *Figure 3.1* (p. 30).

Figure 3.1 ANSYS POLYMAN Interface at Startup



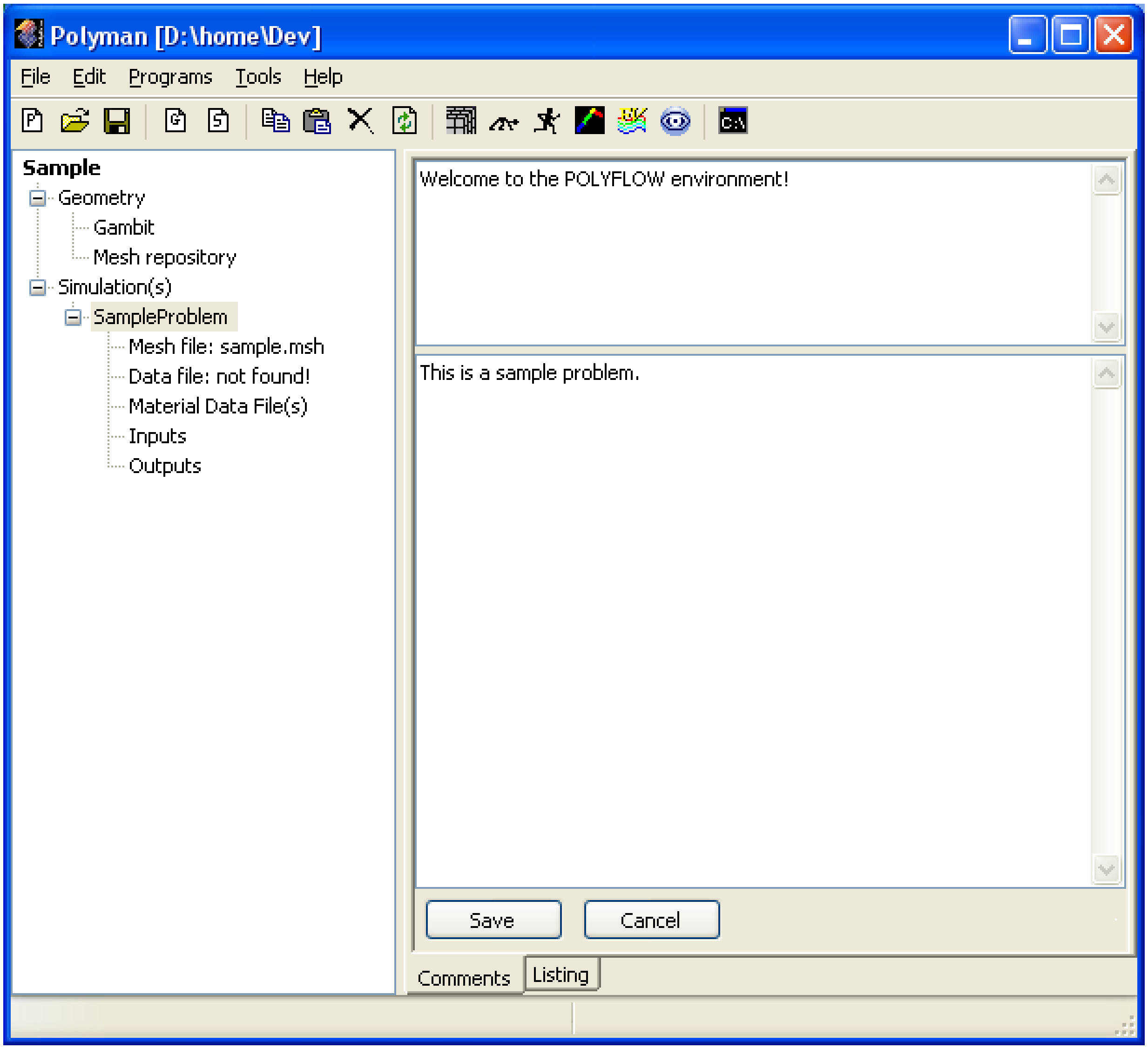
Welcome to the POLYFLOW environment!

After a project and simulation have been created, and files have been imported or created, the interface will appear as shown in *Figure 3.2* (p. 31).

A right-click on most items allows you to copy, paste, and delete. A right-click on mesh and data files allows you to start ANSYS POLYDATA. The main components of the ANSYS POLYMAN interface are:

- Menu bar
- Toolbar
- Tree region
- Information region
- Comment region
- Tab Bar

Figure 3.2 ANSYS POLYMAN Interface After Creation of a Project and Simulation



Welcome to the POLYFLOW environment!

This is a sample problem.

3.2.1. Menu Bar

Figure 3.3 ANSYS POLYMAN Menu Bar

The menu bar contains five pull-down menus:

File

contains menu items for creating new projects and simulations, opening existing projects, importing and exporting files, and exiting ANSYS POLYMAN. See *Creating a New Project, Simulation, or GAMBIT Branch* (p. 34), *Opening an Existing Project* (p. 36), *Importing Files into a Project* (p. 36), *Exporting Files* (p. 39), and *Exiting ANSYS POLYMAN* (p. 59) for details.

Edit

contains menu items for copying and deleting files, and refreshing the project tree. **Edit** also allows you to rename a simulation. See *Copying and Deleting Files* (p. 40) for details.

Programs

contains menu items for starting the programs that are included in the ANSYS POLYFLOW package. See *Starting the Programs* (p. 40) for details.

Tools

contains menu items for assorted tasks, including converting a GAMBIT neutral file, testing a UDF file, viewing a listing file, obtaining information about a project or simulation, and modifying startup options for ANSYS POLYDATA and ANSYS POLYFLOW. **Tools** also allows you to schedule, kill, stop, restart running calculations, and to gzip FLUENT/Post result files. See *Setting Options for ANSYS POLYDATA, ANSYS POLYFLOW, FieldView & CFD-Post* (p. 43), *Converting a GAMBIT Neutral File* (p. 47), *Viewing a Listing File* (p. 48), *Scheduling a Simulation* (p. 48), and *Obtaining Information about a Project or Simulation* (p. 48) for details.

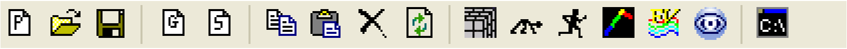
You can open a Command Prompt window (on a Windows system) or an xterm window (on a Linux system) by selecting the **Tools/Shell** menu item.

Help

contains menu items for getting help on using ANSYS POLYMAN, as well as direct access to ANSYS, Inc. web sites. See *Getting Help* (p. 58) for details.

3.2.2. Toolbar

Figure 3.4 ANSYS POLYMAN Toolbar



The toolbar contains the command buttons (see *Table 3.1: Toolbar Buttons* (p. 32)

Table 3.1 Toolbar Buttons

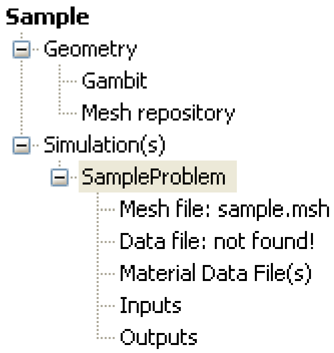
Icon	Action	Equivalent to	Shortcut
P	creates a new project	**File** > **New** > **Open**	
📂	opens an existing project	**File** > **Open**	**Ctrl-O**
💾	saves the project	**File** > **Save**	
G	creates a new GAMBIT file	**File** > **New** > **Gambit file**	
S	opens a new simulation	**File** > **New** > **Simulation**	
⎘	copies the selected item	**File** > **Copy**	**Ctrl-C**
📋	pastes the copied item	**File** > **Paste**	**Ctrl-V**
✕	deletes the selected item	**File** > **Delete**	**Del**
🔄	creates a new project	**File** > **Refresh**	
▦	launches GAMBIT	**Programs** > **Gambit**	**Ctrl-G**

Icon	Action	Equivalent to	Shortcut
	launches ANSYS POLYDATA	**Programs** > **Polydata**	**Ctrl-D**
	launches ANSYS POLYFLOW	**Programs** > **Polyflow**	**Ctrl-F**
	launches FLUENT/Post	**Programs** > **Flpost**	**Ctrl-P**
	launches FieldView	**Programs** > **FieldView**	
	launches CFD-Post	**Programs** > **CFD-Post**	
C:\	opens a terminal (xterm)	**Programs** > **Others**	

3.2.3. Tree Region

The tree region is located on the left side of the ANSYS POLYMAN window (see *Figure 3.2* (p. 31)). It shows the current structure of the project you are working on, including a section for GAMBIT files and a section for the simulation (see *Figure 3.5* (p. 33)).

Figure 3.5 ANSYS POLYMAN Tree Region



```
Sample
⊟ Geometry
    Gambit
    Mesh repository
⊟ Simulation(s)
    ⊟ SampleProblem
        Mesh file: sample.msh
        Data file: not found!
        Material Data File(s)
        Inputs
        Outputs
```

The simulation section may contain several branches, corresponding to the different simulations you are running for this project. ANSYS POLYMAN organizes the files related to a given simulation or a given project in a way that makes it easier for you to keep track of them. Rather than searching through your files and directories manually, you can view the files for your current analysis using a project tree. Similarly, the GAMBIT section may contain several branches corresponding to different geometries and/or meshes.

3.2.4. Information Region

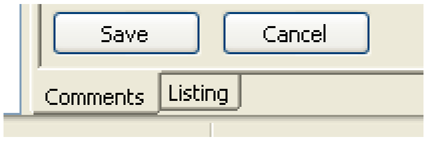
The information region is located on the upper right side of the window. It lists all the messages issued by the ANSYS POLYFLOW codes during the current session.

3.2.5. Comment Region

The comment region is located below the information region. If you select an item in the project tree (as shown in *Figure 3.2* (p. 31)), any comment associated with this item will appear in the comment region. You can click your mouse in the comment region and edit or add text, and then click **Save** to save the new or modified comment. To undo an edit before clicking **Save**, click **Cancel** or simply select another item in the project tree.

3.2.6. Tab Bar

Figure 3.6 ANSYS POLYMAN Tab Bar



There are two tabs below the comment region:

Comments
Contains the Information region

Listing
Allows you to display the listing file corresponding to a specific simulation.

3.3. Creating a New Project, Simulation, or GAMBIT Branch

After you start ANSYS POLYMAN (as described in *Starting ANSYS POLYMAN* (p. 29)), you will need to either create a new project, as described in the following section, or open an existing project (as described in *Opening an Existing Project* (p. 36)).

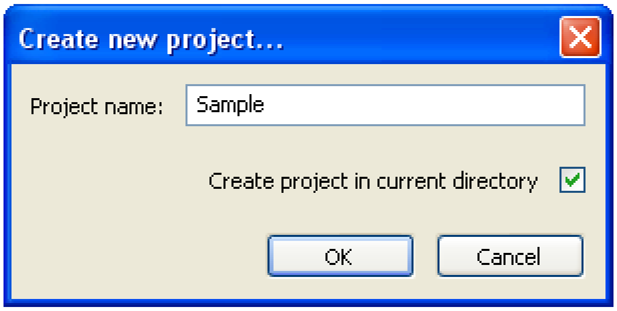
3.3.1. Creating a New Project

A project contains all the files related to a given analysis. It can include several different geometries and meshes, as well as multiple simulations.

To create a new project, select **Project**:

File → **New** → **Project**

Figure 3.7 The Create new project Panel



Project name: Sample

Create project in current directory ☑

You can also open a new project by clicking on the corresponding icon in the toolbar. In the resulting **Create new project** panel (*Figure 3.7* (p. 34)), specify the **Project name** (only letters and numbers are allowed; no spaces or other characters are allowed), indicate whether or not you want to **Create project in current directory**, and click **OK**. If you turn off the **Create project in current directory** option, you will be asked (after you click **OK**) to specify the desired directory.

After the project is created, its name will appear at the top of the tree, as shown in *Figure 3.2* (p. 31), along with two branches: **Geometry** and **Simulation(s)**. ANSYS POLYMAN will create a directory with the same name as the project, as well as a file with the same name as the project and a `.prj` extension (e.g., `Sample.prj`). The `Geometry` and `Simulation` directories are created within the project directory. These two directories contain all the information related to the project.

Important

Note that ANSYS POLYMAN also creates a `.prj_sav` file. This file is a copy of the `.prj` file and can be useful for recovering a project if the project file becomes corrupted.

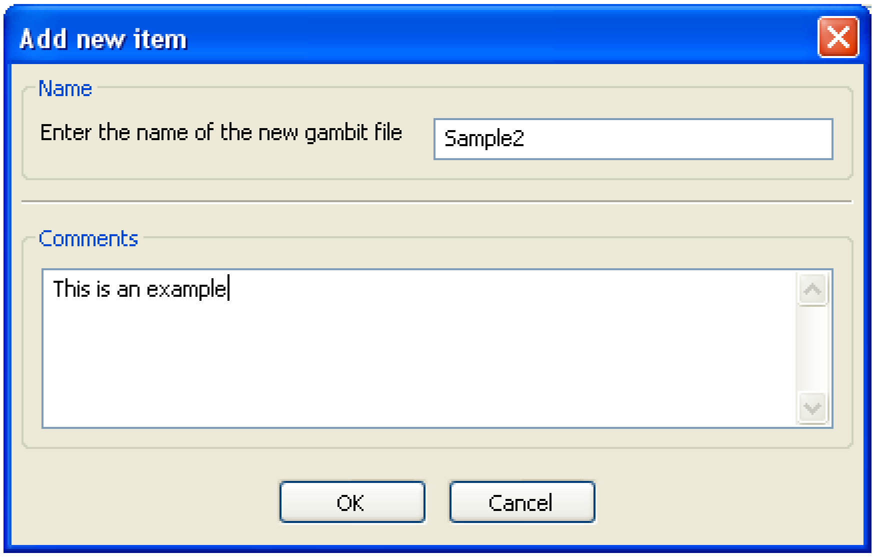
3.3.2. Creating a New Branch for GAMBIT Files

Once you have created or opened a project, you can add a new branch under **Gambit** in the project tree.

File → **New** → **Gambit file**

You can also open a new GAMBIT file by clicking on the corresponding icon in the toolbar. In the resulting **Add new item** panel (*Figure 3.8* (p. 35)), specify the name of the new file and any comments, if desired, and click **OK**.

Figure 3.8 The Add new item Panel



Add new item

Name

Enter the name of the new gambit file: Sample2

Comments

This is an example

ANSYS POLYMAN will add a new item under **Gambit** in the project tree. This branch will contain all the files related to GAMBIT for this particular geometry. It is recommended that you create a new branch each time you modify the geometry or create a new geometry.

3.3.3. Creating a New Simulation

Before you can solve your problem, you need to first create a new simulation for the project.

File → **New** → **Simulation**

You can also open a new simulation by clicking on the corresponding icon in the toolbar. In the resulting **Add new item** panel (similar to *Figure 3.8* (p. 35)), specify the name of the new simulation and any comments, if desired, and click **OK**. ANSYS POLYMAN will add a new item under **Simulation(s)** in the project tree. This branch will contain all the files related to this simulation, including the mesh file, data file, material data file(s), input file(s), and results file(s).

A simulation cannot contain more than one mesh file or data file, so you cannot store multiple mesh or data files in a single simulation. If you make any changes to the material properties, operating conditions, or numerical parameters in the data file, for example, you can either overwrite the original data file or create a new simulation.

In the project directory under `Simulation`, ANSYS POLYMAN will create a directory with the name you specified for the new simulation. This directory contains the `Inputs`, `MaterialData`, and `Outputs` directory.

3.4. Opening an Existing Project

As noted earlier, ANSYS POLYMAN creates a `.prj` file for each new project. To reopen an existing project, you will select the corresponding `.prj` file.

You can open an existing project from the menu bar:

File → **Open...**

or by typing **Ctrl-O**

or by clicking on the corresponding icon in the toolbar.

In the **Choose a file** panel, select the `.prj` file for the desired project, and click **OK**.

The four projects that you accessed most recently will appear at the bottom of the **File** menu. You can select one of these as a shortcut. Also, on a Windows system, you can double-click on a `.prj` file to start ANSYS POLYMAN and open the selected project.

3.5. Importing Files into a Project

Many types of files can be imported into ANSYS POLYMAN. Details about each are provided in the following sections.

3.5.1. Importing a GAMBIT File

To import a GAMBIT database file, use the **File/Import/Gambit file** menu item.

File → **Import** → **Gambit file**

In the **Choose a file** panel, select the appropriate `.dbs` file and click **Open** (or double-click the file name). ANSYS POLYMAN will store the file in the `Geometry/Gambit` subdirectory and create a leaf named **filename.dbs** under the **Gambit** section of the project tree.

3.5.2. Importing a Simulation

To import a whole simulation, use the **File/Import/Simulation** menu item.

File → **Import** → **Simulation**

In the resulting **Choose a file** panel, select the appropriate `.zip` file and click **Open**, or double-click the file name (ANSYS POLYMAN exports a simulation to a file using a `.zip` file extension, see *Exporting Files* (p. 39)). ANSYS POLYMAN will create a new simulation in the project with all associated files as stored during the exportation of the simulation. The name of the simulation will be the same in the current project as in the project from which it was exported.

3.5.3. Importing an ANSYS POLYFLOW Mesh File

To import an ANSYS POLYFLOW mesh file, use the **File/Import/Polyflow mesh** menu item.

File → **Import** → **Polyflow mesh**

In the resulting **Choose a file** panel, select the appropriate `.msh` file and click **Open** (or double-click the file name). You will then be prompted to specify the simulation where you want to store this mesh file.

ANSYS POLYMAN will store the file in the appropriate simulation subdirectory and create a leaf named **filename.msh** under **Mesh file** in the relevant branch of the **Simulation(s)** section of the project tree.

3.5.4. Importing a Material Data File

To import a material data file, use the **File/Import/Material data file** menu item.

File → **Import** → **Material data file**

In the resulting **Choose a file** panel, select the appropriate `.mat` file and click **Open** (or double-click the file name). You will then be prompted to specify the simulation where you want to store this material data file. ANSYS POLYMAN will store the file in the `MaterialData` subdirectory of the appropriate simulation subdirectory and create a leaf named **filename.mat** under **Material Data File(s)** in the relevant branch of the **Simulation(s)** section of the project tree.

3.5.5. Importing a Data File

To import an ANSYS POLYFLOW data file, use the **File/Import/Data file** menu item.

File → **Import** → **Data file**

In the **Choose a file** panel, select the appropriate `.dat` file and click **Open** (or double-click the file name). You will then be prompted to specify the simulation where you want to store this data file. ANSYS POLYMAN will store the file in the appropriate simulation subdirectory and create a leaf named **filename.dat** under **Data file** in the relevant branch of the **Simulation(s)** section of the project tree.

3.5.6. Importing a Result File

To import an ANSYS POLYFLOW result file, use the **File/Import/Result file** menu item.

File → **Import** → **Result file**

In the resulting **Choose a file** panel, select the appropriate `.res` file and click **Open** (or double-click the file name). You will then be prompted to specify the simulation where you want to store this result file. ANSYS POLYMAN will store the file in the `Inputs` subdirectory of the appropriate simulation subdirectory and create a leaf named **filename.res** under **Inputs** in the relevant branch of the **Simulation(s)** section of the project tree.

3.5.7. Importing a UDF File

To import a UDF file, use the **File/Import/UDF file** menu item.

File → Import → UDF file

In the **Choose a file** panel, select the appropriate `.`UDF file and click **OK**. You will be prompted to specify the simulation where you want to store this UDF file.

ANSYS POLYMAN will store the file in the `Inputs` subdirectory of the appropriate simulation subdirectory and create a leaf named **filename.udf** under **Inputs** in the relevant branch of the **Simulation(s)** section of the project tree.

The filename of imported UDF files must contain the character string `udf`. Examples of valid names include `udf.clp`, `flow_udf.clp`, and `flow.udf`.

3.5.8. Importing a CSV File

To import a CSV (comma separated variable) file, use the **File/Import/CSV file** menu item.

File → Import → CSV file

In the **Choose a file** panel, select the appropriate `.csv` file and click **Open**. You will be prompted to specify the simulation where you want to store this CSV file. ANSYS POLYMAN will store the file in the `Inputs` subdirectory of the appropriate simulation subdirectory and create a leaf named **filename.csv** under **Inputs** in the relevant branch of the **Simulation(s)** section of the project tree.

3.5.9. Importing a `.poly` File

To import a `.poly` file created by ANSYS Meshing or ANSYS ICEM CFD, use the **File/Import/ANSYS Meshing file (*.poly)** or **File/Import/Icem file (*.poly)** menu item.

File → Import → ANSYS Meshing file (*.poly)

or

File → Import → Icem file (*.poly)

In the **Choose a file** panel, select the appropriate `.poly` file and click **Open**. You will be prompted to specify the simulation where you want to store the mesh from this file. ANSYS POLYMAN will convert the `.poly` file to an ANSYS POLYFLOW mesh file, store it in the appropriate simulation subdirectory, and create a leaf named `filename.msh` under **Mesh file** in the relevant branch of the **Simulation(s)** section of the project tree.

For more information about `.poly` files, see *`.poly` Meshes Created with ANSYS ICEM CFD or ANSYS Meshing* (p. 139).

3.5.10. Importing an I-deas File

To import an I-deas universal file, use the **File/Import/Ideas file** menu item.

File → Import → Ideas file

In the **Choose a file** panel, select the appropriate `.unv` file and click **Open**. You will be prompted to specify the simulation where you want to store the mesh from this file.

ANSYS POLYMAN will convert the I-deas file to an ANSYS POLYFLOW mesh file, store it in the appropriate simulation subdirectory and create a leaf named **filename.msh** under **Mesh file** in the relevant branch of the **Simulation(s)** section of the project tree.

3.5.11. Importing a PATRAN File

To import a PATRAN neutral file, use the **File/Import/Patran file** menu item.

File → **Import** → **Patran file**

In the **Choose a file** panel, select the appropriate `.pat` file and click **Open** (or double-click the file name). You will then be prompted to specify the simulation where you want to store the mesh from this file.

ANSYS POLYMAN will convert the PATRAN file to an ANSYS POLYFLOW mesh file, store it in the appropriate simulation subdirectory and create a leaf named **filename.msh** under **Mesh file** in the relevant branch of the **Simulation(s)** section of the project tree.

3.5.12. Importing a (user) File

It might be useful to store notes, images, or presentations in an ANSYS POLYMAN project in order to centralize the source of information. These files can be imported into a user folder.

> **Note**
>
> In order to import a user file, a user folder must already exist in the project structure. To create a user folder, right-click on a simulation and select **New folder**. A new **MyFiles** leaf will be created in the simulation. This default name can be modified by clicking on the leaf twice (a slow double-click).

To import a file with an undefined type, first select the user folder in the project, then use the **File/Import/File** menu item.

File → **Import** → **File**

In the **Choose a file** panel, browse to the directory containing the user file, select the file, and click **Open** (or double-click the file name).

3.6. Exporting Files

To export a file to a different directory (e.g., so that you can more easily send it to your technical support engineer if you have a question about it), first select the file to be exported, and then select the **File**/**Export...** menu item.

File → **Export...**

In the **Choose a file** panel, go to the directory on your network where you would like to place a copy of the file, and specify the file name for the copy.

A whole simulation can be exported as a `.zip` file containing the complete structure of the simulation with all its files.

3.7. Copying and Deleting Files

You can copy and delete files that are visible in the project tree.

- To copy a file, select it and then select the **Edit/Copy** menu item

  Edit → **Copy**

  or type **Ctrl**-**C**.
- To paste a copied file, select the simulation where you want to paste it, and then select the **Edit/Paste** menu item

  Edit → **Paste**

  or type **Ctrl**-**V**.
- To rename a file or the simulations, select it and then select the **Edit/Rename** menu item

  Edit → **Rename**

  or right-click the name of the simulation in the tree.

  ANSYS POLYMAN will check that the desired name is actually different from existing names.
- To update the project tree with your latest changes, select the **Edit/Refresh** menu item.

  Edit → **Refresh**

  The project tree is automatically refreshed on a regular basis, so you will generally not need to refresh it manually.
- To delete a file, select it and then select the **Edit/Delete** menu item

  Edit → **Delete**

  or press the `Delete` key on your keyboard.
- To delete all of the graphic output files (i.e., `.flum` and `.flur` files) from a simulation, select the simulation and then select the **Edit/Delete Outputs** menu item.

  Edit → **Delete Outputs**

  The **Delete output files ?** panel will open, which allows you to select the files that you want to delete. Multiple files can be selected by holding the **Shift** key or the **Ctrl** key while clicking the left mouse button.

3.8. Starting the Programs

In addition to starting ANSYS POLYDATA, ANSYS POLYFLOW, GAMBIT, and other related programs in the usual way, you can also start them from within ANSYS POLYMAN, as described in this section.

3.8.1. Starting GAMBIT

There are four ways to start GAMBIT from ANSYS POLYMAN:

- Select a file (not a neutral file) in the relevant **Gambit** branch of the project tree, and then select the **Programs/Gambit** menu item.

  Programs → Gambit
- Select a file (not a neutral file) in the relevant **Gambit** branch of the project tree, and then type **Ctrl-G**.
- Double-click a file (not a neutral file) in the relevant **Gambit** branch of the project tree.
- Select a file (not a neutral file) in the relevant **Gambit** branch of the project tree, and then click the icon.

You can also use the methods above without first selecting a `.dbs` file. In this case, a new GAMBIT session will start.

3.8.2. Starting ANSYS POLYDATA

There are four ways to start ANSYS POLYDATA from ANSYS POLYMAN:

- Select a mesh or data file in the relevant simulation branch of the project tree, and then select the **Programs/Polydata** menu item.

  Programs → Polydata
- Select a mesh or data file in the relevant simulation branch of the project tree, and then type **Ctrl-D**.
- Double-click a mesh or data file in the relevant simulation branch of the project tree.
- Select a mesh file or data file in the relevant simulation branch of the project tree, and then click the icon.

If you select a mesh file, ANSYS POLYDATA will read the selected mesh file. If you select a data file, ANSYS POLYDATA will read the selected data file and the corresponding mesh file.

3.8.3. Starting ANSYS POLYFLOW

There are four ways to start ANSYS POLYFLOW from ANSYS POLYMAN:

- Select a data file in the relevant simulation branch of the project tree, and then select the **Programs/Polyflow** menu item.

  Programs → Polyflow
- Select a data file in the relevant simulation branch of the project tree, and then type **Ctrl-F**.
- Select a data file in the relevant simulation branch of the project tree, and then click the icon.
- Select the **Tools/Schedule simulations/Add menu** item. See *Scheduling a Simulation* (p. 48) for details.

ANSYS POLYFLOW will compute a solution for the selected data file and save the appropriate output files.

3.8.4. Starting FLUENT/Post

There are four ways to start FLUENT/Post from ANSYS POLYMAN:

- Select a `.flum` and `.flur` file pair under **Outputs** in the relevant simulation branch of the project tree, and then select the **Programs/Flpost** menu item.

Programs → **Flpost**

- Select a `.flum` and `.flur` file pair under **Outputs** in the relevant simulation branch of the project tree, and then type **Ctrl**-**P**.
- Double-click a `.flum` and `.flur` file pair under **Outputs** in the relevant simulation branch of the project tree.
- Select a `.flum` and `.flur` pair of files under **Outputs** in the relevant simulation branch of the project tree, and then click the icon.

FLUENT/Post will read the selected `.flum` and `.flur` files, and display a perspective view of the mesh.

Note

If you click just once on a `.flum` and `.flur` file pair in the project tree, comments about the files, including the value of the time step or evolution variable, will be displayed.

3.8.5. Starting FieldView

There are three ways to start FieldView from ANSYS POLYMAN:

- Select a `.uns` file under **Outputs** in the relevant simulation branch of the project tree, and then select the **Programs/FieldView** menu item.
- Double-click a `.uns` file under **Outputs** in the relevant simulation branch of the project tree.
- Select a `.uns` file under **Outputs** in the relevant simulation branch of the project tree, and then click the icon.

FieldView will start and read the `.uns` file(s) and display a perspective view of a box including the mesh. The mesh must be displayed using FieldView commands.

3.8.6. Starting CFD-Post

There are three ways to start CFD-Post from ANSYS POLYMAN:

- Select a `.cfx.res` file under **Outputs** in the relevant simulation branch of the project tree, and then select:

  Programs → **CFD-Post**
- Double-click a `.cfx.res` file under **Outputs** in the relevant simulation branch of the project tree.
- Select a `.cfx.res` file under **Outputs** in the relevant simulation branch of the project tree, and then click the icon.

CFD-Post will start and read the `.cfx.res` file and display a perspective view of a box including the mesh. The mesh must be displayed using CFD-Post commands.

3.8.7. Starting ANSYS POLYFUSE

There are two ways to start ANSYS POLYFUSE from ANSYS POLYMAN:

- Select the **Programs/Others/Polyfuse** menu item.

Programs → **Others** → **Polyfuse**

- Type **Ctrl**-**U**.

ANSYS POLYFUSE will start up in the `MeshRepository` directory. When you view or list files inside ANSYS POLYFUSE, you will see all of the mesh files stored in the `MeshRepository` directory.

3.8.8. Starting ANSYS POLYSTAT

There are two ways to start ANSYS POLYSTAT from ANSYS POLYMAN:

- Select a simulation under **Simulation(s)** in the project tree, and then select the **Programs/Others/Polystat** menu item.

  Programs → **Others** → **Polystat**

- Select a simulation under **Simulation(s)** in the project tree, and then type **Ctrl**-**S**.

3.8.9. Starting ANSYS POLYMAT

There are two ways to start ANSYS POLYMAT from ANSYS POLYMAN:

- Select a material data file in the relevant simulation branch of the project tree, and then select the **Programs** > **Others** > **Polymat** menu item.

  Programs → **Others** → **Polymat**

- Select a material data file in the relevant simulation branch of the project tree, and then type **Ctrl**-**M**.

ANSYS POLYMAT will start and open the selected material data file.

3.8.10. Starting ANSYS POLYDIAG

There are two ways to start ANSYS POLYDIAG from ANSYS POLYMAN:

- Select a simulation under **Simulation(s)** in the project tree, and then select the **Programs** > **Others** > **Polydiag** menu item.

  Programs → **Others** → **Polydiag**

- Select a simulation under **Simulation(s)** in the project tree, and then type **Ctrl**-**A**.

ANSYS POLYDIAG will start and open the `instances.clp` file for the selected simulation.

3.9. Setting Options for ANSYS POLYDATA, ANSYS POLYFLOW, FieldView & CFD-Post

There are several options for ANSYS POLYDATA, ANSYS POLYFLOW, FieldView, and CFD-Post that you can control. The following sections describe these options for each product.

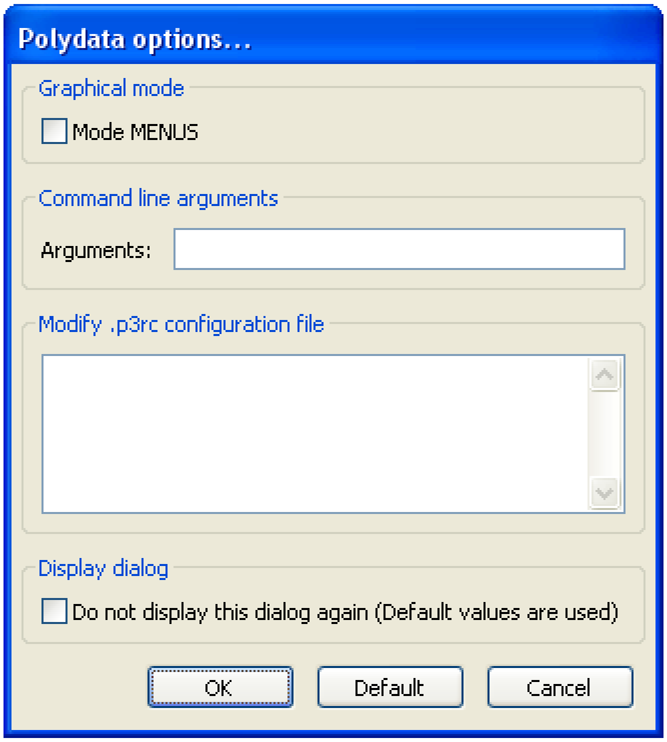
3.9.1. Options for ANSYS POLYDATA

To modify the setup options for ANSYS POLYDATA, select the **Tools** > **Options** > **Polydata...** menu item.

Tools → **Options** → **Polydata...**

The **Polydata options** panel (*Figure 3.9* (p. 44)) will open.

Figure 3.9 The Polydata options Panel



If you would like to run ANSYS POLYDATA without the graphical user interface (equivalent to the `-MENUS` option described in *Starting ANSYS POLYDATA* (p. 10)), turn on the **MENUS** option under **Graphical mode**.

If you want to use any other command-line arguments for ANSYS POLYDATA, enter them in the **Arguments** field under **Command line arguments**.

If you want to modify the .p3rc configuration file, add the relevant keywords under **Modify.p3rc configuration file**. See *The .p3rc Configuration File* (p. 11) for details about this configuration file and its keywords.

If you have modified the settings in this panel, then every time you start ANSYS POLYDATA, this panel will appear so you can confirm that you want to use your latest settings. If you do not want to see this panel every time you start ANSYS POLYDATA, turn on the **Do not display this dialog again** option under **Display dialog**. If this option is on, ANSYS POLYDATA will start up using the settings in the panel. If you want to change the settings again, you will need to open the panel from the **Tools** menu.

You can return to the default settings by clicking the **Default** button at the bottom of the panel.

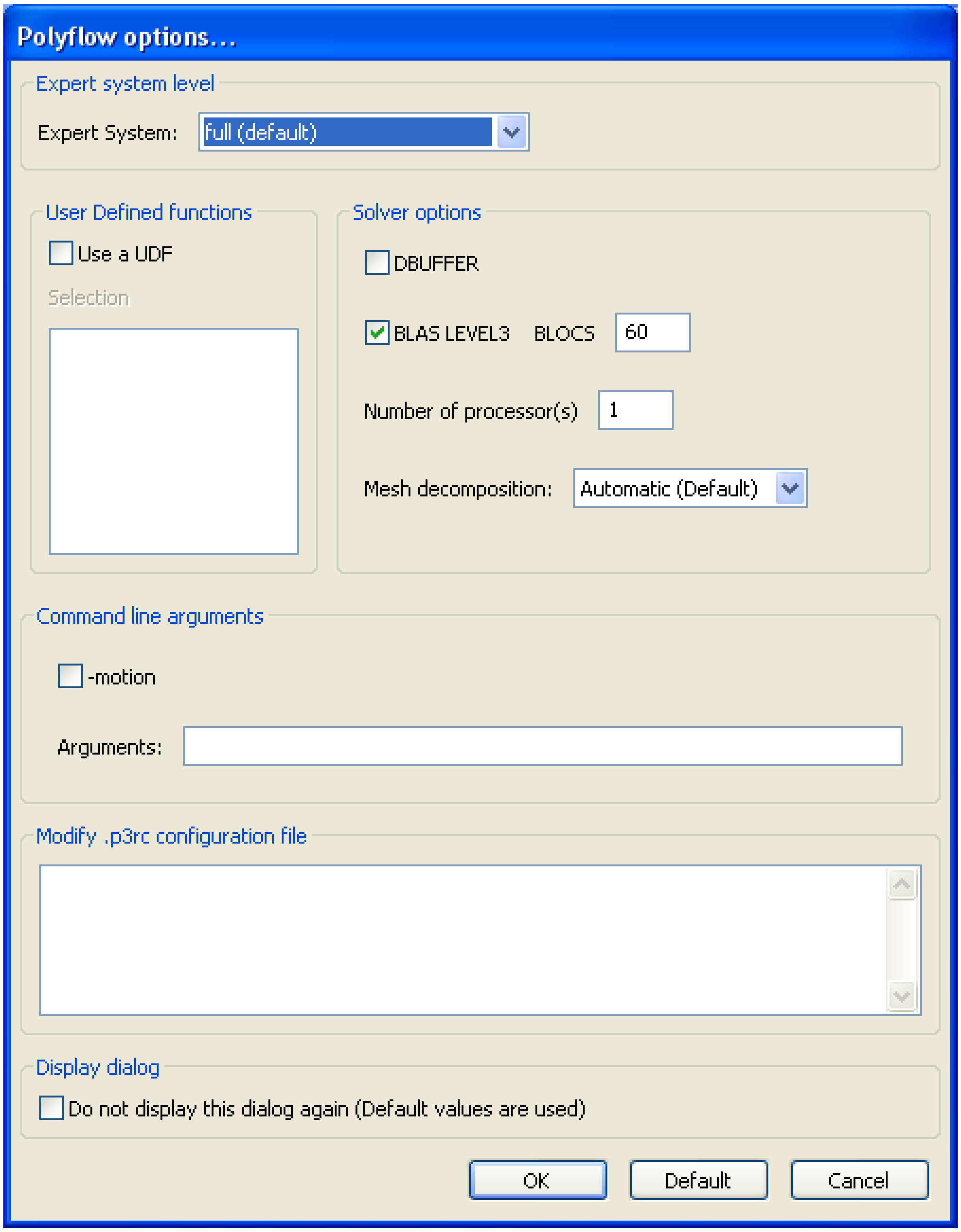
3.9.2. Options for ANSYS POLYFLOW

To modify the setup options for ANSYS POLYFLOW, select the **Tools** > **Options** > **Polyflow...** menu item.

Tools → **Options** → **Polyflow...**

The **Polyflow options** panel (*Figure 3.10* (p. 45)) will open.

Figure 3.10 The POLYFLOW options Panel



Expert System: full (default)

☐ Use a UDF

☐ DBUFFER

☑ BLAS LEVEL3 BLOCS 60

Number of processor(s) 1

Mesh decomposition: Automatic (Default)

☐ -motion

Arguments:

☐ Do not display this dialog again (Default values are used)

You have the following options:

- If you want to modify the level of assistance you would like from the expert system, select the appropriate option in the **Expert System** drop-down list under **Expert system level**. Choose **full (default)** to use the standard expert system, **no** to disable the expert system, or **nofile** to prevent the expert system from writing the `instances.clp` file (see *Loading, Viewing, and Updating the Simulation Status* (p. 583)).
- If you want to run the simulation using double-precision numbers for the factorized matrix, turn on the **DBUFFER** option under **Solver options**.
- If you want to switch from BLAS3 to another level, deselect **BLAS LEVEL 3** under **Solver options**. If you keep the default BLAS3 solver (which is recommended) you can modify the size of the blocks in the **BLOCS** field. See *ANSYS POLYFLOW's Implementation* (p. 580) for details.

- If you are running the parallel version of ANSYS POLYFLOW, you can specify the number of processors on which to run a simulation in the **Number of processor(s)** field under **Solver options**.
- If you want to explicitly specify the number of subdomains for decomposition, select the relevant item in the **Mesh decomposition** drop-down list under **Solver options**. See *Mesh Decomposition and Optimization* (p. 146) for details.
- If you want to use any special command-line arguments for ANSYS POLYFLOW, enter them in the **Arguments** field under **Command line arguments**.
- If you want to run ANSYS POLYFLOW with a UDF file, click the **Use a UDF** button and select the required UDF file. Filenames that contain the characters string `udf` will be visible, and therefore available.
- If you want to modify the `.p3rc` configuration file, add the relevant keywords under **Modify.p3rc configuration file**. See *The .p3rc Configuration File* (p. 11) for details about this configuration file and its keywords.
- If you have modified the settings in this panel, then every time you start ANSYS POLYFLOW, this panel will appear so you can confirm that you want to use your latest settings. If you do not want to see this panel every time you start ANSYS POLYFLOW, turn on the **Do not display this dialog again** option under **Display dialog**. If this option is on, ANSYS POLYFLOW will start up using the settings in the panel. If you want to change the settings again, you will need to open the panel from the **Tools** menu.
- You can return to the default settings by clicking the **Default** button at the bottom of the panel.

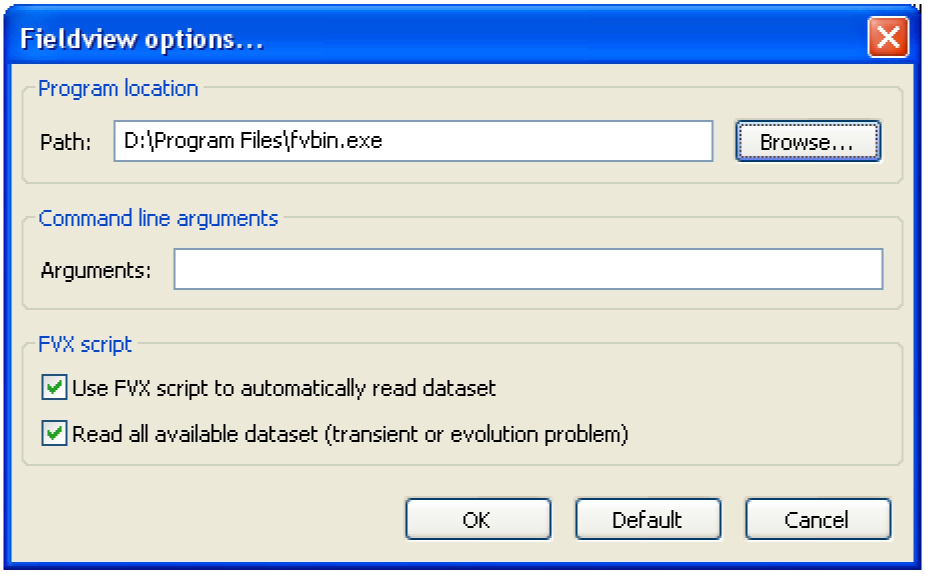
3.9.3. Options for FieldView

To modify the setup options for FieldView, select the **Tools** > **Options** > **FieldView...** menu item.

Tools → **Options** → **FieldView...**

The **FieldView options** panel (*Figure 3.11* (p. 46)) will open.

Figure 3.11 The FieldView options Panel



Fieldview options...

Program location

Path: D:\Program Files\fvbin.exe

Command line arguments

Arguments:

FVX script

☑ Use FVX script to automatically read dataset

☑ Read all available dataset (transient or evolution problem)

Note the following:

- As FieldView is not installed via the ANSYS FLUENT installation procedure, the path to the right executable must be provided before launching FieldView. This path is stored in the `polyman.ini` file in your `HOME` directory. So you do not have to enter it for each ANSYS POLYMAN session.
- If you want to use any special command-line arguments for FieldView, enter them in the **Arguments** field under **Command** line arguments.
- Deselecting **Use FVX script to automatically read dataset** makes FieldView start without reading any file. Files can be read using standard FieldView commands.
- Deselecting **Read all available dataset (transient or evolution problem)** makes FieldView read only the selected `.uns` file.

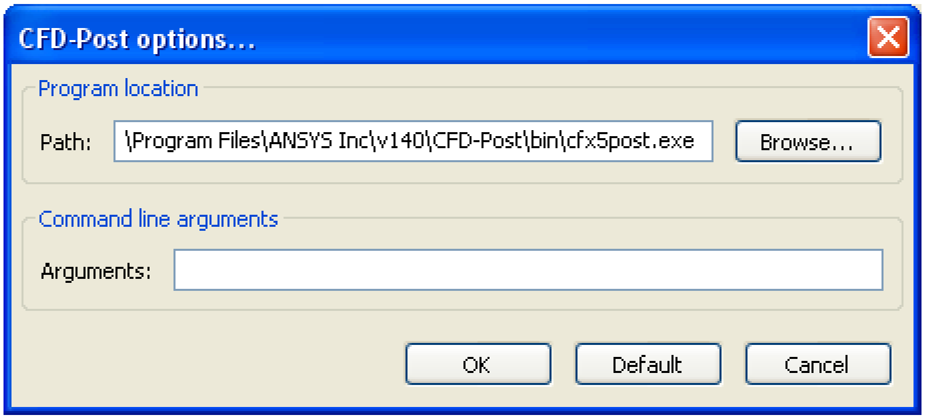
3.9.4. Options for CFD-Post

To modify the setup options for CFD-Post, select the **Tools** > **Options** > **CFD-Post...** menu item.

Tools → **Options** → **CFD-Post...**

The CFD-Post options panel (*Figure 3.12* (p. 47)) will open.

Figure 3.12 The CFD-Post options Panel



CFD-Post options...

Program location

Path: \Program Files\ANSYS Inc\v140\CFD-Post\bin\cfx5post.exe

Command line arguments

Arguments:

Note the following:

1. Since ANSYS POLYFLOW and CFD-Post are installed independently, the path to the CFD-Post executable must be provided before launching CFD-Post. This path is stored in the `polyman.ini` file in your `HOME` directory. So you do not have to enter it for each ANSYS POLYMAN session.
2. If you want to use any special command-line arguments for CFD-Post, enter them in the **Arguments** field under **Command line arguments**.

3.10. Converting a GAMBIT Neutral File

To convert a GAMBIT neutral file to a mesh file that ANSYS POLYDATA can read, do the following:

1. Select the `.neu` file under the relevant **Gambit** branch in the project tree.
2. Select the **Tools** > **Convert a gambit neutral file** menu item.

   Tools → **Convert a gambit neutral file**
3. Select (or create) the simulation where you want to import the mesh.

ANSYS POLYMAN will convert the selected neutral file to a mesh file, and store it under **Mesh file** in the selected simulation branch of the project tree. For more information about GAMBIT meshes, see *Meshes Created with GAMBIT* (p. 138).

3.11. Viewing a Listing File

There are two ways to view the listing file for a simulation:

- Select a simulation under **Simulation(s)** in the project tree, and then select the **Tools/View listing** menu item.

  Tools → **View listing**
- Select a simulation under **Simulation(s)** in the project tree, and then type **Ctrl-L**.

See *Writing an ANSYS POLYFLOW Listing File* (p. 110) for details about listing files.

3.12. Obtaining Information about a Project or Simulation

3.12.1. Obtaining Project Information

To obtain information about a project, select the **Tools/Project information** menu item

Tools → **Project information**

or type **Ctrl-I**.

ANSYS POLYMAN will open an **Information** panel containing details about the current project. Each file is listed, along with its complete path and the date of its creation or last modification. If the mesh file is newer than the data file, a warning message is added to let you know that the data and mesh files may not be compatible.

If the output files are newer than either the data file or the mesh file, this means you have not run the simulation after modifying the data or mesh file. Thus, the output files do not correspond to the data and/or mesh files. A warning message will appear in this case as well.

3.12.2. Obtaining Simulation Information

To obtain information about a particular simulation, first select the simulation in the project tree. Then select the **Tools/Simulation information** menu item.

Tools → **Simulation information**

ANSYS POLYMAN will open an **Information** panel containing details similar to those described above for the project information report, but only for the selected simulation.

3.13. Scheduling a Simulation

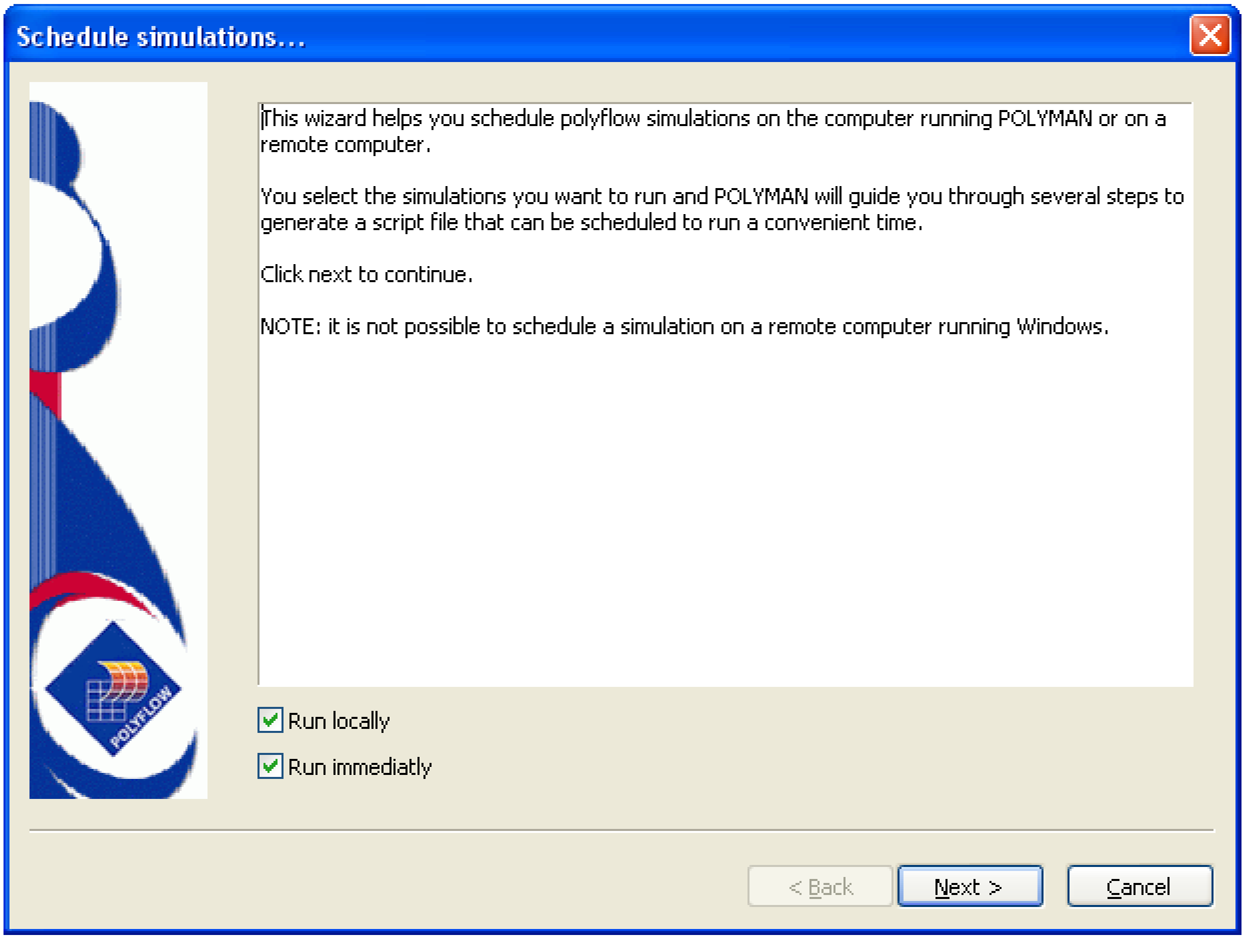
To take advantage of the reduced loading of computers during nights or weekends, you can schedule when the simulation will start. To schedule a simulation, select the **Tools/Schedule simulations/Add** menu item.

Tools → **Schedule simulations** → **Add**

This will open a wizard that guides you through the procedure for scheduling a simulation on the computer running ANSYS POLYMAN or on a remote machine. This wizard also allows you to define a list of simulations that will be run successively on a given machine at a given time.

In the first panel of the wizard (shown in *Figure 3.13* (p. 49), **Run locally** should be unchecked if you want to use a remote computer and **Run immediately** should be unchecked if you want to delay the start of the calculation. Click the **Next** button to go to the second panel.

Figure 3.13 The First Schedule simulations... Panel


Schedule simulations...

This wizard helps you schedule polyflow simulations on the computer running POLYMAN or on a remote computer.

You select the simulations you want to run and POLYMAN will guide you through several steps to generate a script file that can be scheduled to run a convenient time.

Click next to continue.

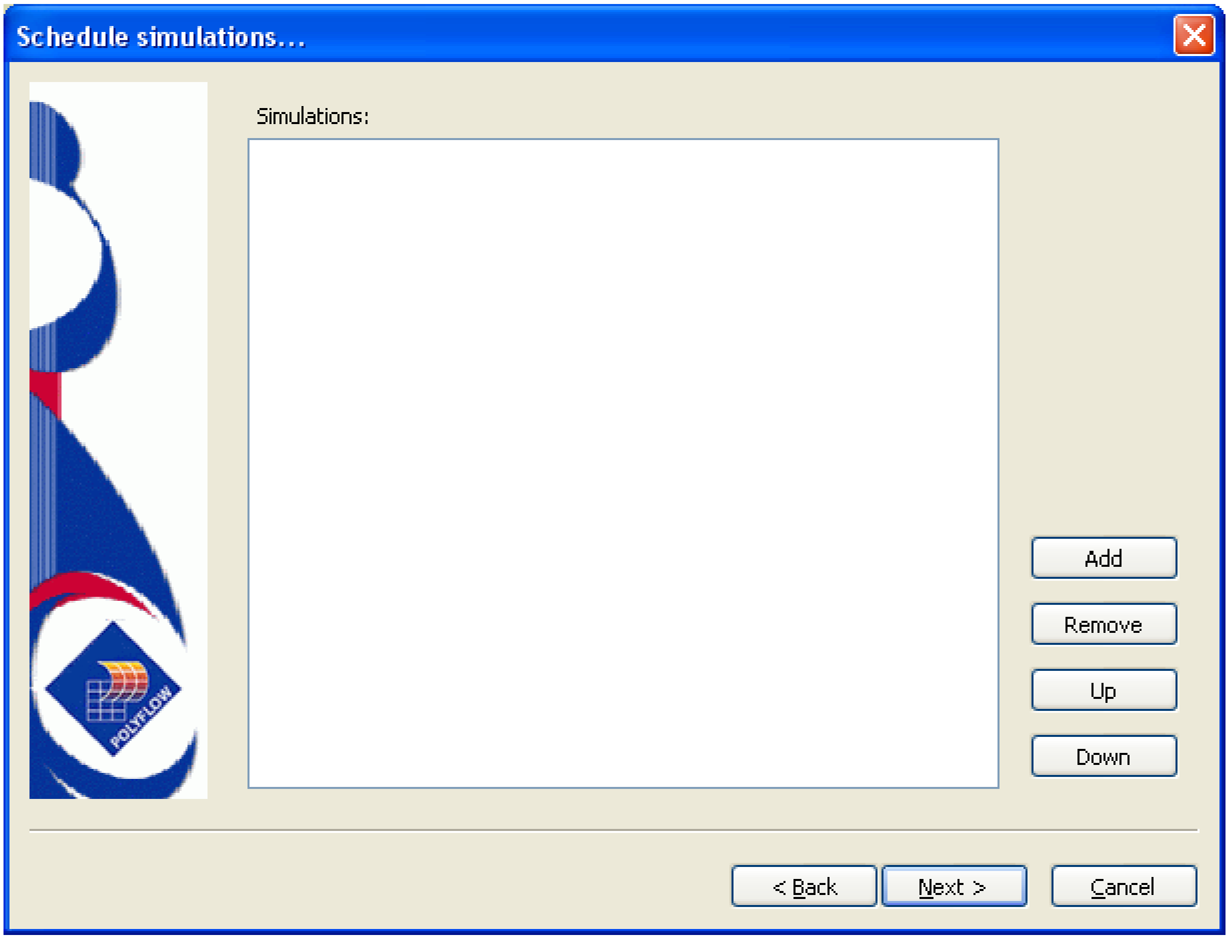
NOTE: it is not possible to schedule a simulation on a remote computer running Windows.

☑ Run locally

☑ Run immediatly

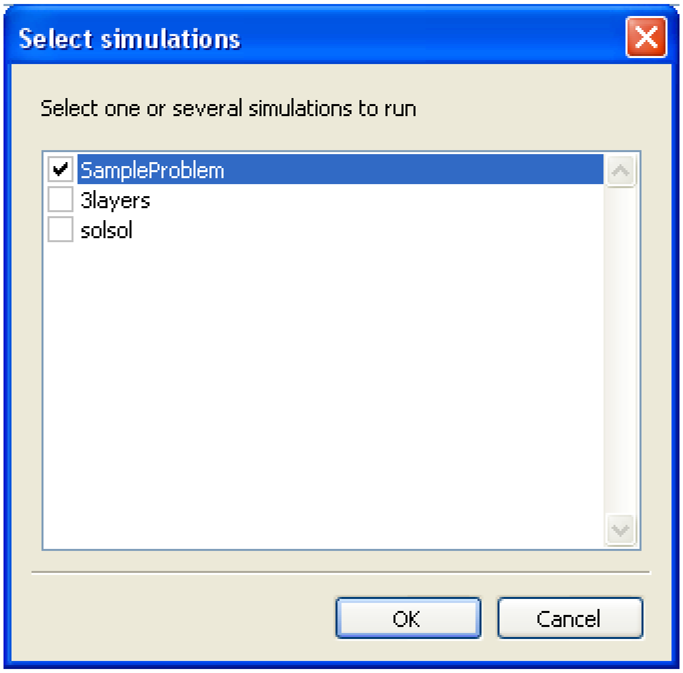
The second panel (*Figure 3.14* (p. 50)) allows you to specify the simulation(s) that will be launched.

Figure 3.14 The Second Schedule simulations... Panel



To add a simulation to the list, click **Add**. In the **Select simulations** dialog box that opens (*Figure 3.15* (p. 51)), click the simulation(s) you want to run and click **OK**.

Figure 3.15 The Select simulations... Panel



Select simulations

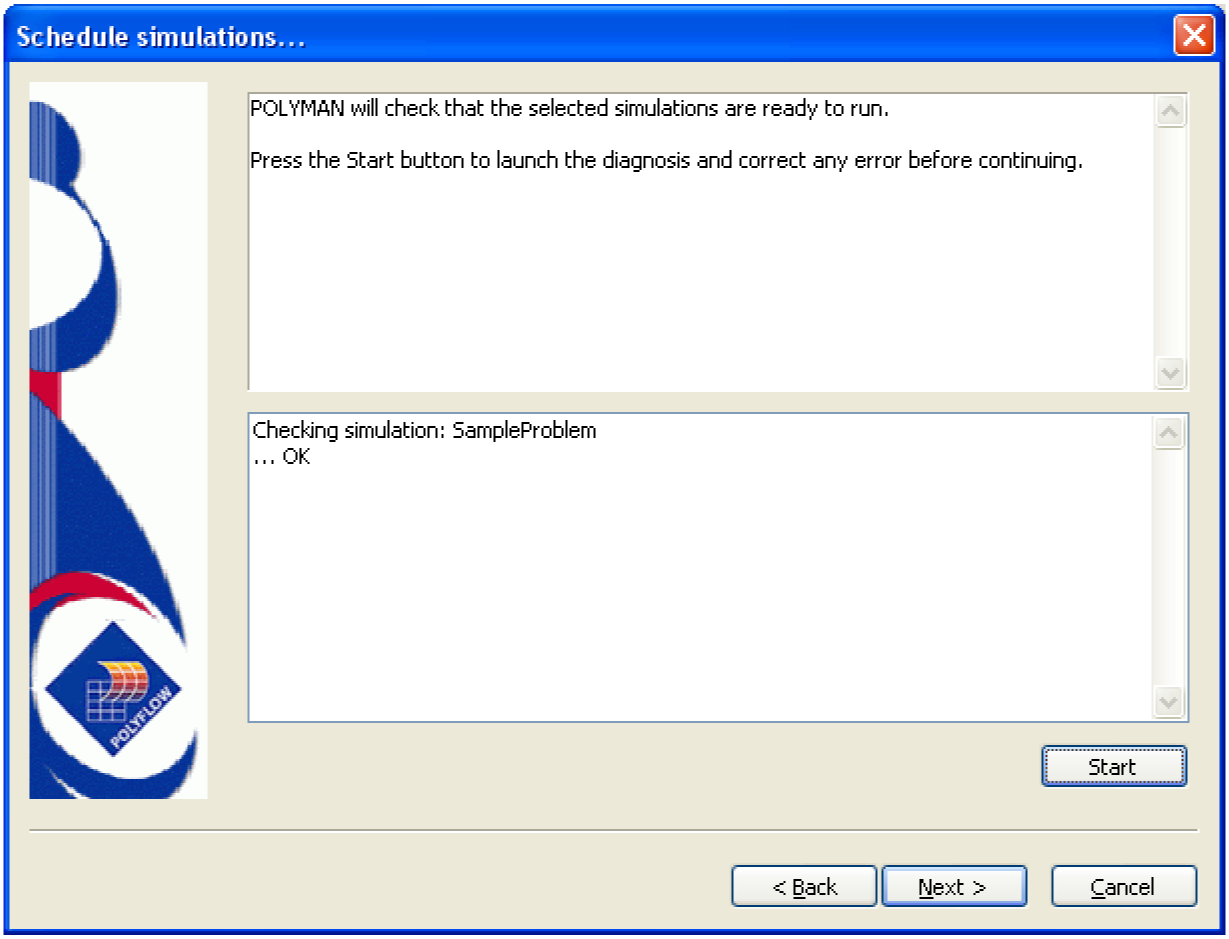
Select one or several simulations to run

☑ SampleProblem
☐ 3layers
☐ solsol

The second panel will display the simulations that are scheduled to run. If you want to remove a simulation, select it in the list and click **Remove**. You can also revise the order in which they are run by selecting a simulation in the list and clicking the **Up** or **Down** buttons. When you are done revising the simulation list, click the **Next** button.

The third panel (*Figure 3.16* (p. 52)) allows you to verify that the selected simulations are ready to run. Click the **Start** to run the diagnosis and then correct any errors. If the results of the diagnosis are acceptable (as shown in *Figure 3.16* (p. 52)), you can click **Next**.

Figure 3.16 The Third Schedule simulations... Panel



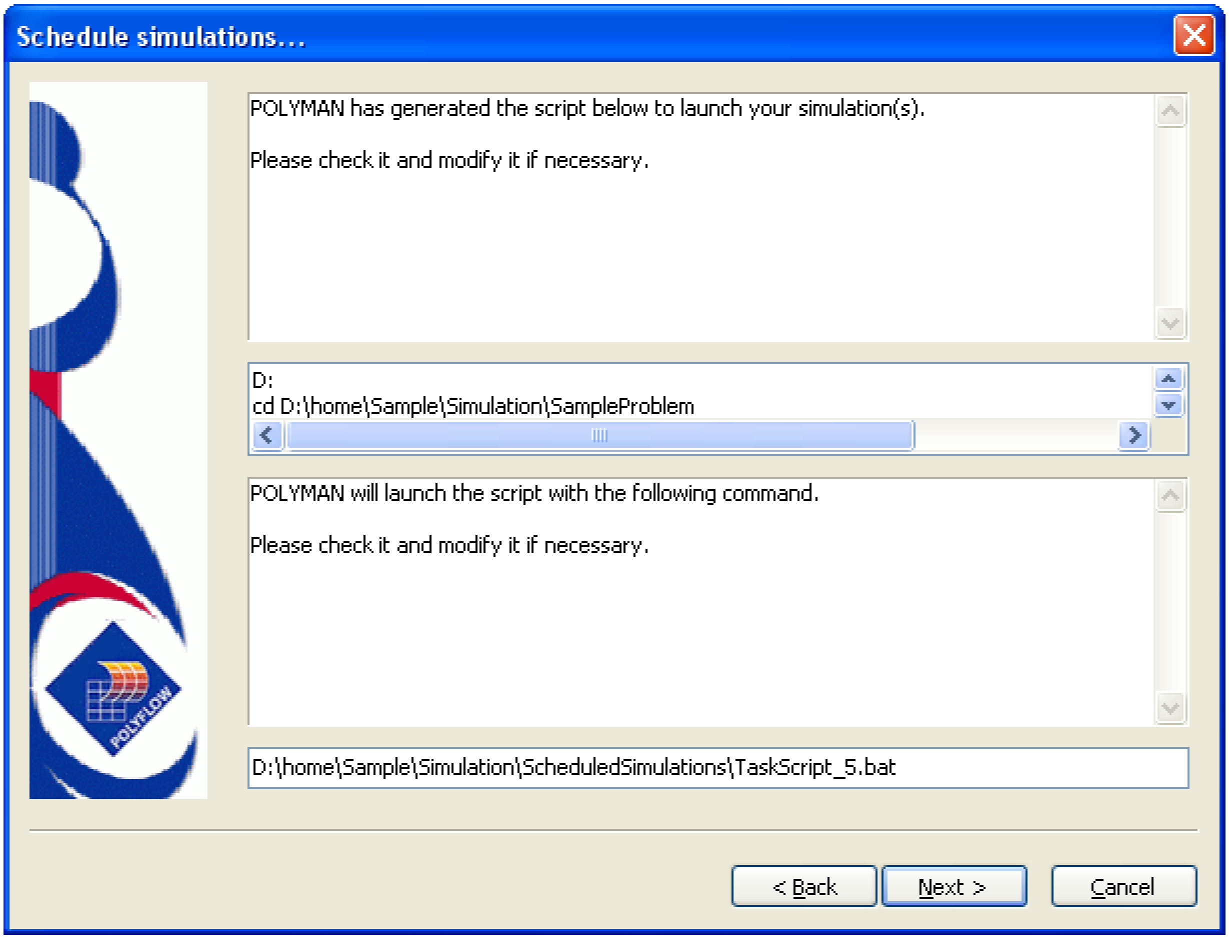
POLYMAN will check that the selected simulations are ready to run.

Press the Start button to launch the diagnosis and correct any error before continuing.

```
Checking simulation: SampleProblem
... OK
```

The fourth panel (*Figure 3.17* (p. 53)) displays the script that will launch your simulations, as well as the command that will launch the script. You can edit these as necessary, and then click **Next**.

Figure 3.17 The Fourth Schedule simulations... Panel



POLYMAN has generated the script below to launch your simulation(s).

Please check it and modify it if necessary.

```
D:
cd D:\home\Sample\Simulation\SampleProblem
```

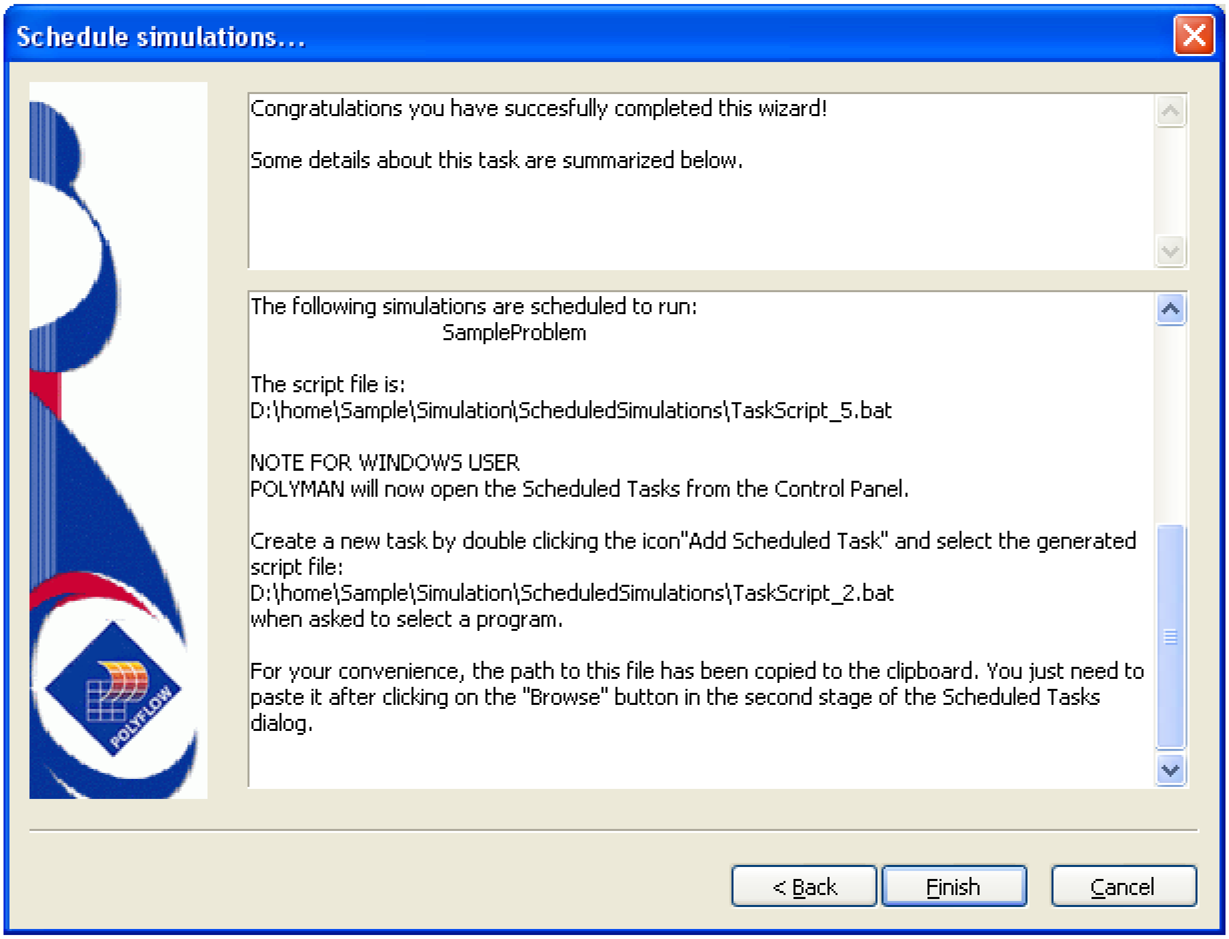
POLYMAN will launch the script with the following command.

Please check it and modify it if necessary.

```
D:\home\Sample\Simulation\ScheduledSimulations\TaskScript_5.bat
```

The fifth panel (*Figure 3.18* (p. 54)) summarizes the simulations that are scheduled to run. Note the name of the generated script file displayed in this panel, as you will use it in a later step. Click **Finish** to close the **Schedule simulations...** panel.

Figure 3.18 The Fifth Schedule simulations... Panel



Congratulations you have succesfully completed this wizard!

Some details about this task are summarized below.

The following simulations are scheduled to run:
SampleProblem

The script file is:
D:\home\Sample\Simulation\ScheduledSimulations\TaskScript_5.bat

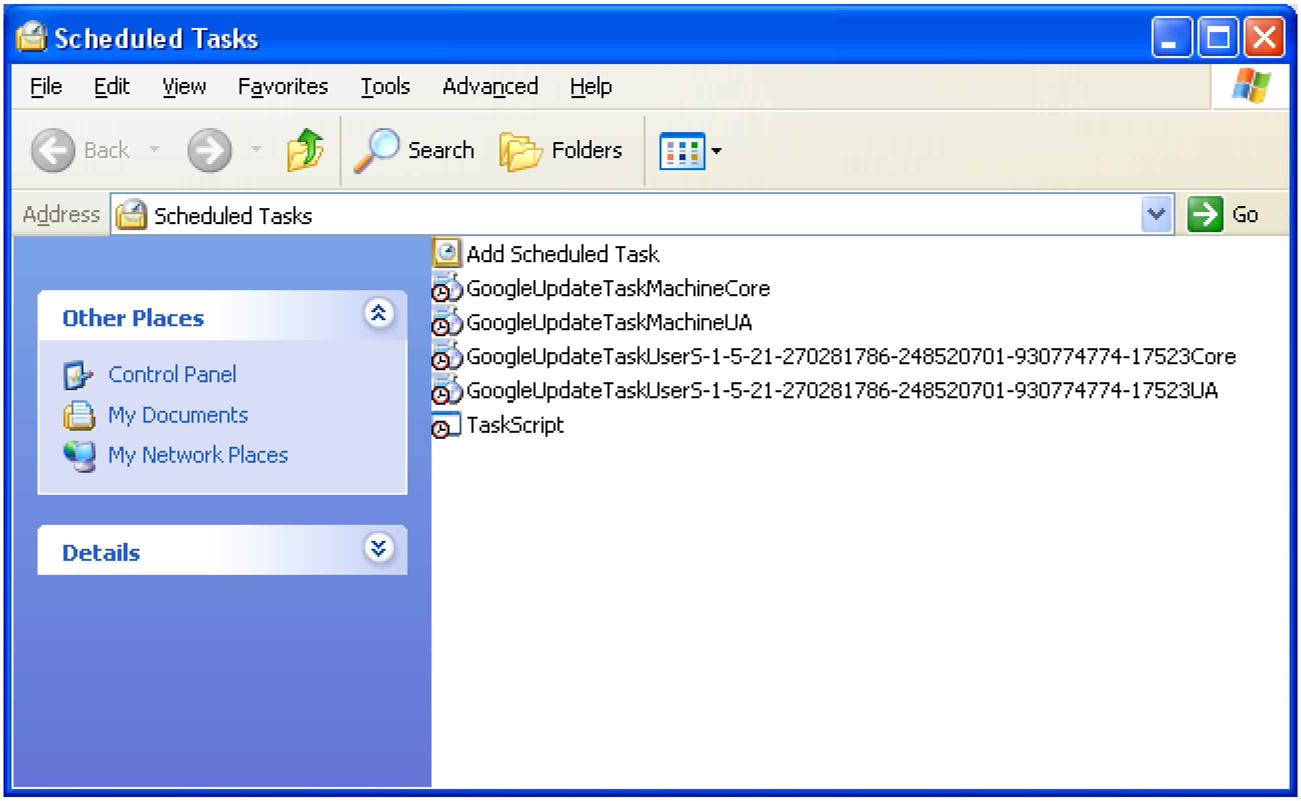
NOTE FOR WINDOWS USER
POLYMAN will now open the Scheduled Tasks from the Control Panel.

Create a new task by double clicking the icon"Add Scheduled Task" and select the generated script file:
D:\home\Sample\Simulation\ScheduledSimulations\TaskScript_2.bat
when asked to select a program.

For your convenience, the path to this file has been copied to the clipboard. You just need to paste it after clicking on the "Browse" button in the second stage of the Scheduled Tasks dialog.

In the **Scheduled Tasks** window that opens (*Figure 3.19* (p. 55)), click **Add Scheduled Task** to open a wizard that allows you to schedule a task.

Figure 3.19 The Scheduled Tasks Window

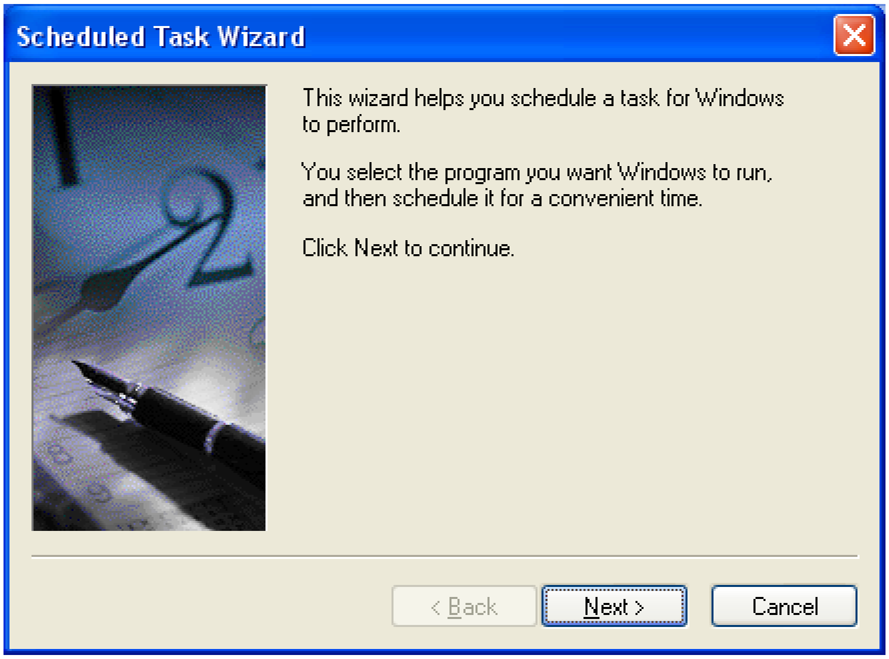


Scheduled Tasks

Add Scheduled Task
GoogleUpdateTaskMachineCore
GoogleUpdateTaskMachineUA
GoogleUpdateTaskUserS-1-5-21-270281786-248520701-930774774-17523Core
GoogleUpdateTaskUserS-1-5-21-270281786-248520701-930774774-17523UA
TaskScript

Click **Next** in the **Scheduled Task Wizard** that opens (*Figure 3.20* (p. 55)).

Figure 3.20 The First Scheduled Task Wizard Panel



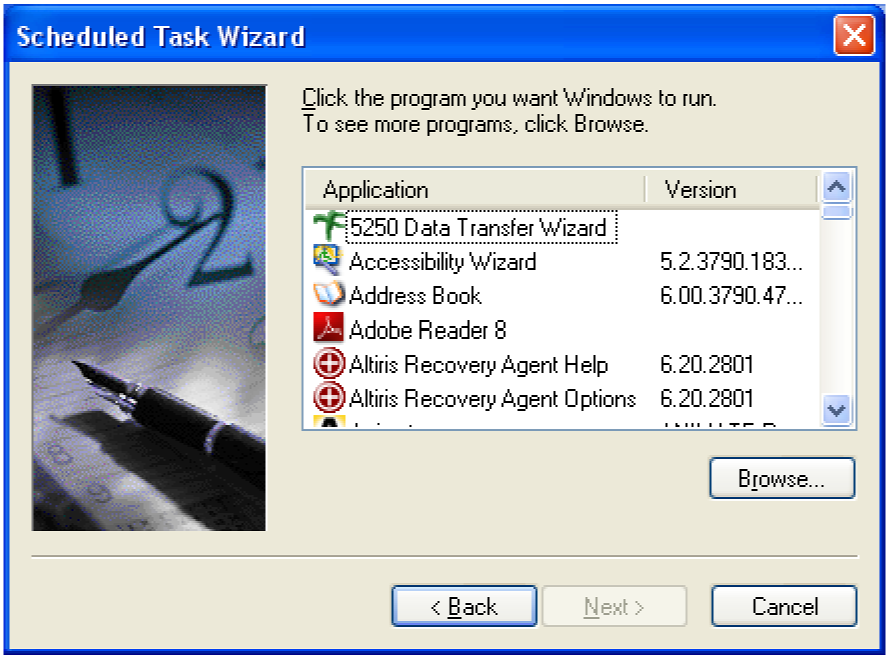
This wizard helps you schedule a task for Windows to perform.

You select the program you want Windows to run, and then schedule it for a convenient time.

Click Next to continue.

Click the **Browse...** button in the second panel of the **Scheduled Task Wizard** (*Figure 3.21* (p. 56)).

Figure 3.21 The Second Scheduled Task Wizard Panel

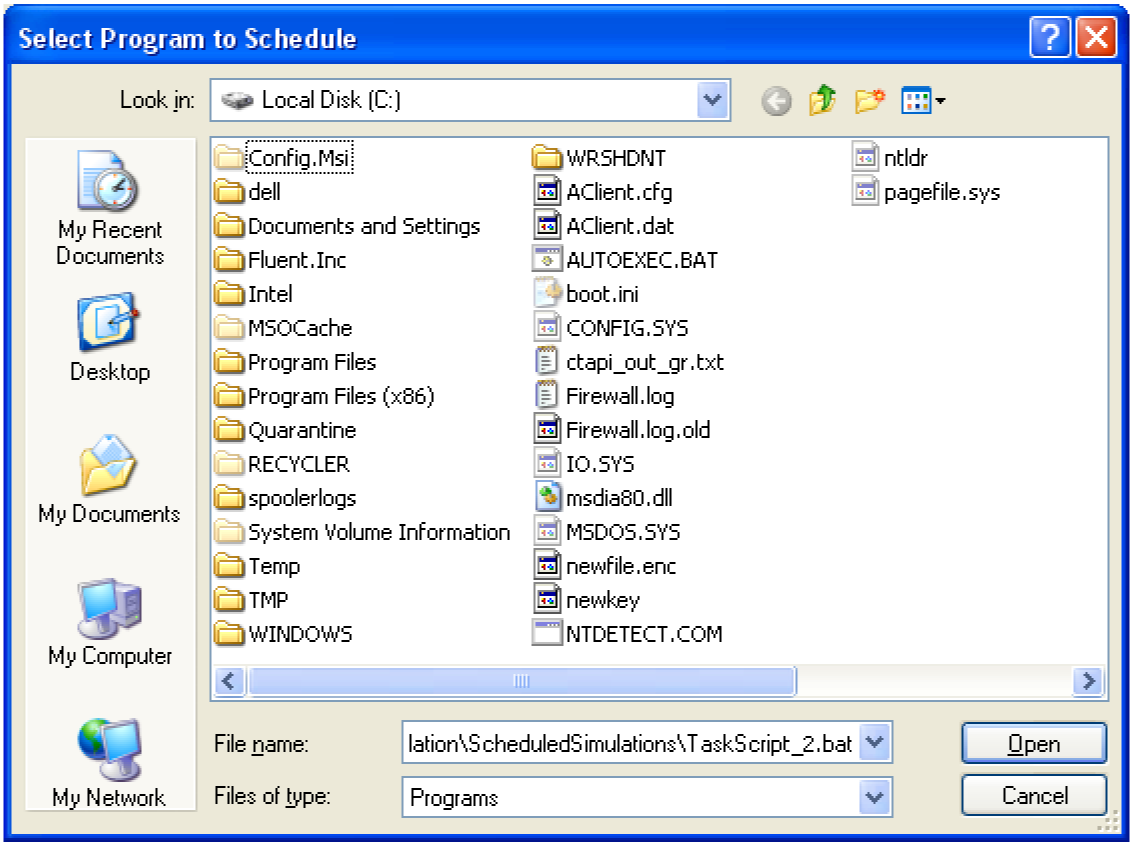


Click the program you want Windows to run.
To see more programs, click Browse.

Application	Version
5250 Data Transfer Wizard	
Accessibility Wizard	5.2.3790.183...
Address Book	6.00.3790.47...
Adobe Reader 8	
Altiris Recovery Agent Help	6.20.2801
Altiris Recovery Agent Options	6.20.2801

In the **Select Program to Schedule** dialog box that opens, enter the generated script file that was displayed in the **Schedule simulations...** panel for **File name:** and click **Open**.

Figure 3.22 The Select Program to Schedule



Look in: Local Disk (C:)

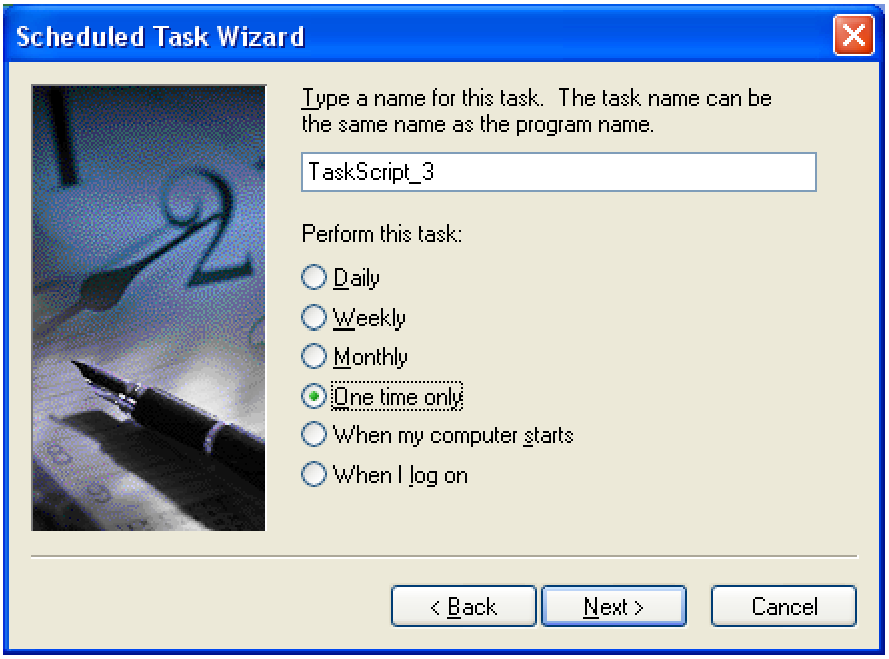
Config.Msi
dell
Documents and Settings
Fluent.Inc
Intel
MSOCache
Program Files
Program Files (x86)
Quarantine
RECYCLER
spoolerlogs
System Volume Information
Temp
TMP
WINDOWS
WRSHDNT
AClient.cfg
AClient.dat
AUTOEXEC.BAT
boot.ini
CONFIG.SYS
ctapi_out_gr.txt
Firewall.log
Firewall.log.old
IO.SYS
msdia80.dll
MSDOS.SYS
newfile.enc
newkey
NTDETECT.COM
ntldr
pagefile.sys

File name: lation\ScheduledSimulations\TaskScript_2.bat

Files of type: Programs

In the third panel of the **Scheduled Task Wizard**, select **One time only** and click **Next**.

Figure 3.23 The Third Scheduled Task Wizard Panel



Scheduled Task Wizard

Type a name for this task. The task name can be the same name as the program name.

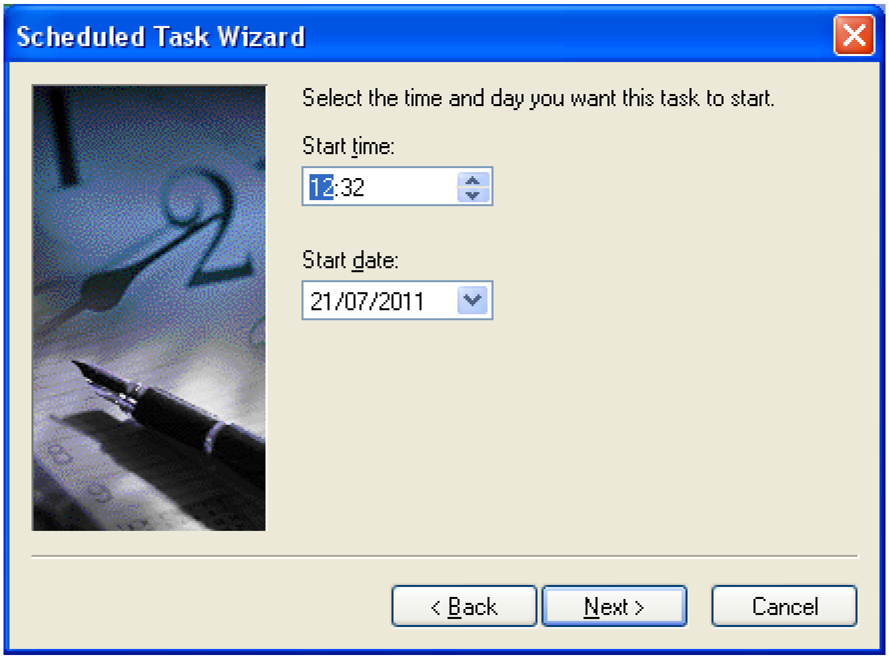
TaskScript_3

Perform this task:

○ Daily
○ Weekly
○ Monthly
● One time only
○ When my computer starts
○ When I log on

Enter the **Start time** and **Start date** in the fourth panel (*Figure 3.24* (p. 57)), and click **Next**.

Figure 3.24 The Fourth Scheduled Task Wizard Panel



Scheduled Task Wizard

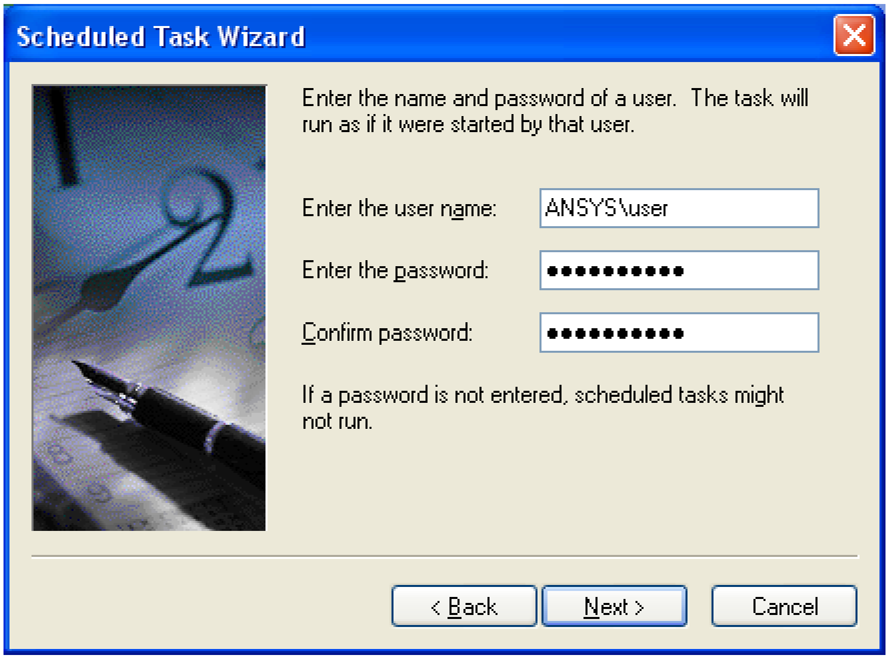
Select the time and day you want this task to start.

Start time: 12:32

Start date: 21/07/2011

Enter your user name, and then enter and confirm the password in the fifth panel (*Figure 3.25* (p. 58)). Then click **Next**.

Figure 3.25 The Fifth Scheduled Task Wizard



Enter the name and password of a user. The task will run as if it were started by that user.

Enter the user name: ANSYS\user

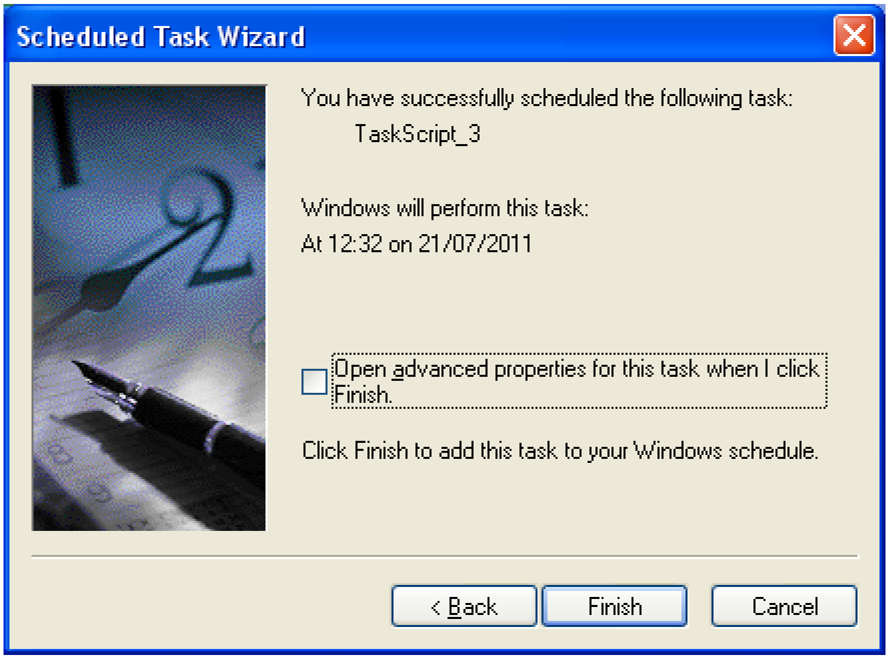
Enter the password: ●●●●●●●●●●●

Confirm password: ●●●●●●●●●●●

If a password is not entered, scheduled tasks might not run.

Click **Finish** in the sixth panel (*Figure 3.26* (p. 58)).

Figure 3.26 The Sixth Scheduled Task Wizard



You have successfully scheduled the following task:

TaskScript_3

Windows will perform this task:

At 12:32 on 21/07/2011

☐ Open advanced properties for this task when I click Finish.

Click Finish to add this task to your Windows schedule.

3.14. Getting Help

The **Help** pull-down menu gives you access to information that can help you use POLYMAN.

Getting started
opens the **Help about...** dialog box, which provides information about using ANSYS POLYMAN.

ANSYS Inc on the Web
opens a submenu that provides the following options:

ANSYS Inc Home Page
provides a shortcut to the ANSYS, Inc. home page, www.ansys.com.

ANSYS Customer Portal
provides a shortcut to the ANSYS, Inc. Customer Portal, www.ansys.com/customerportal.

Polyflow Home Page
provides a shortcut to the POLYFLOW home page, www.ansys.com/products.

POLYFLOW User's Guide...
opens the ANSYS help viewer to the table of contents of the POLYFLOW User's Guide.

POLYMAT User's Guide...
opens the ANSYS help viewer to the table of contents of the POLYMAT User's Guide.

POLYFLOW in Workbench User's Guide...
opens the ANSYS help viewer to the table of contents of the POLYFLOW in Workbench User's Guide.

POLYFLOW in Workbench Tutorial...
opens the ANSYS help viewer to the table of contents of the POLYFLOW in Workbench Tutorial.

PDF
opens a submenu, that allows you to access PDF files of the POLYFLOW User's Guide, POLYMAT User's Guide, POLYSTAT User's Guide, POLYFLOW in Workbench User's Guide, and POLYFLOW in Workbench Tutorial. The PDF files will be displayed in Adobe Reader, which you can download it (at no cost) from http://www.adobe.com.

About
gives you information about the version of POLYMAN and some details about its development.

3.15. Exiting ANSYS POLYMAN

To exit ANSYS POLYMAN, either select the **File** > **Exit** menu item:

File → **Exit**

or type **Ctrl**-**X**. You will be asked to confirm that you really want to exit.

Chapter 4: Sample Session

To demonstrate the use of ANSYS POLYMAN to create and manage a project, ANSYS POLYDATA to set up a problem, ANSYS POLYFLOW to solve it, and CFD-Post to postprocess the results, a sample session is provided in this section. You can follow along as the problem is set up and solved, and the results are examined. The problem solved here is presented again as Tutorial 1 in the ANSYS POLYFLOW Tutorial Guide. After you read through this sample session, you can set up and solve the problem yourself, following the more detailed instructions provided in Tutorial 1.

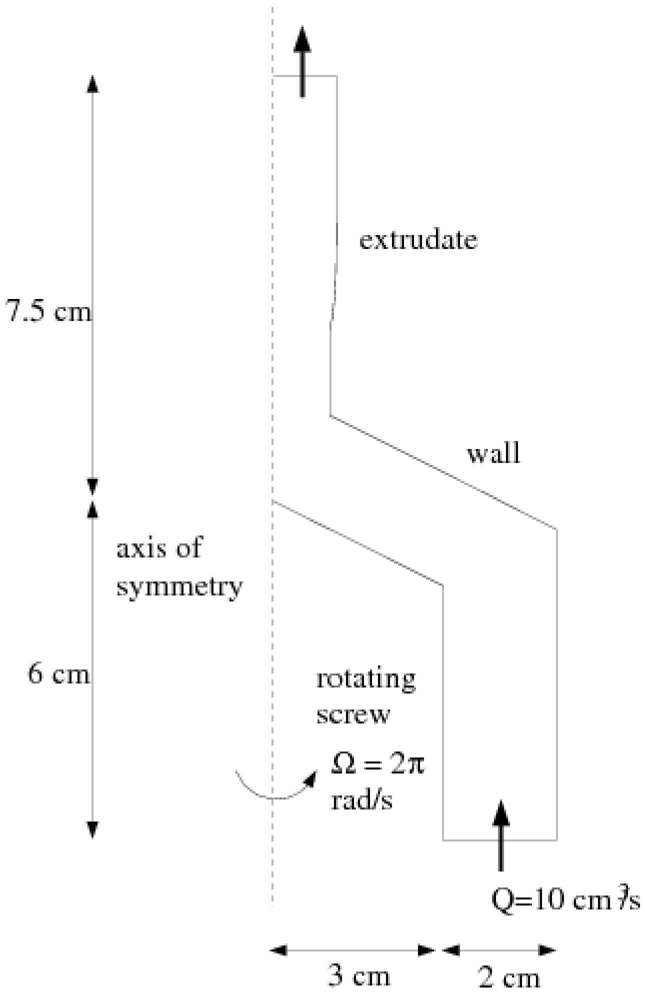
> **Important**
>
> This sample session is designed to give you a general idea of how ANSYS POLYMAN, ANSYS POLYDATA, ANSYS POLYFLOW, and CFD-Post work. It will not supply detailed explanations for the modeling options that are selected, and it should not be the only example you read before you attempt to solve your own problem. The ANSYS POLYFLOW Tutorial Guide contains a number of more complex examples, presented with more complete explanations of the selected options.

The various aspects of the sample session are described in the following sections:

4.1. Problem Description

The problem solved here is illustrated in *Figure 4.1* (p. 62). The fluid enters the domain at a flow rate of 10 cm^3/s, and the screw rotates at an angular velocity of 2π rad/s. In the upper part of the domain, a free surface is used to model the extrudate going out of the extrusion die; the position of the free surface is unknown. A portion of the mesh will be affected by this unknown boundary, so a remeshing technique will be applied on this part of the mesh.

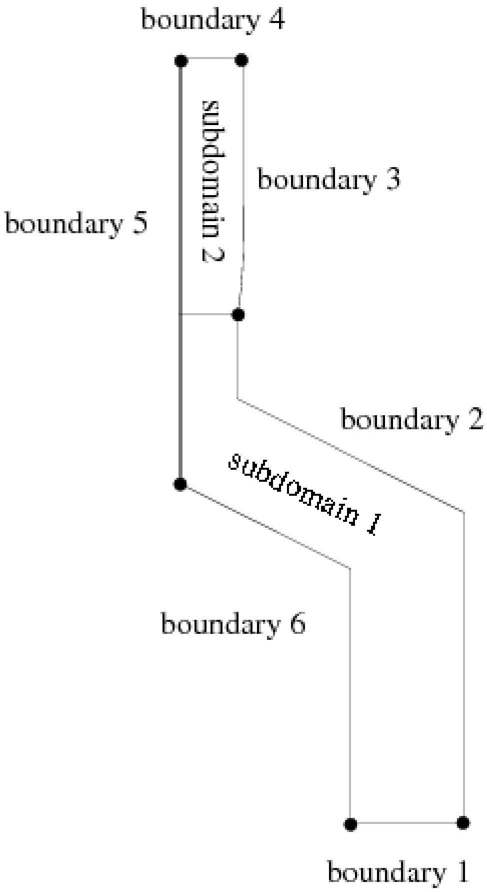
Figure 4.1 Problem Description



4.2. Outline of Procedure

The extrusion problem shown in *Figure 4.1* (p. 62) is a simple isothermal problem involving a single fluid material. You will therefore need to define a task with a single sub-task. Since the problem involves a free surface, the domain is divided into two subdomains: one for the region near the free surface and one for the rest of the domain, as shown in *Figure 4.2* (p. 63).

Figure 4.2 Subdomains and Boundary Sets for the Sample Problem



The two subdomains have been created during the mesh generation, and are stored in the mesh file. You can read the mesh file, with all this information, into ANSYS POLYDATA.

The steps you will follow in this sample session are reduced to the following:

1. Create a project in ANSYS POLYMAN.
2. Start ANSYS POLYDATA from ANSYS POLYMAN.
3. Create a new task.
4. Create a sub-task.
5. Set the fluid properties.
6. Specify the flow boundary conditions.
7. Define the remeshing.
8. Assign the stream function.
9. Save the data file and exit ANSYS POLYDATA.
10. Calculate a solution with ANSYS POLYFLOW.
11. Examine the results with CFD-Post.

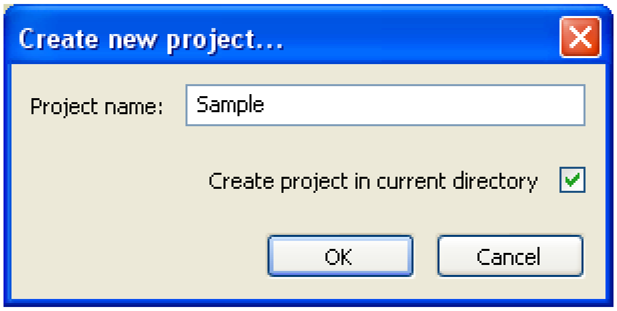
4.3. Creating a Project and Importing the Mesh

Start ANSYS POLYMAN by typing `polyman` at the command prompt, as described in *Starting ANSYS POLYMAN* (p. 29). Select **File/New/Project** to create a new project.

File → **New** → **Project**

The **Create new project** panel (*Figure 4.3* (p. 64)) will open.

Figure 4.3 Create new project Panel



Project name: Sample

Create project in current directory ☑

Then perform the following steps:

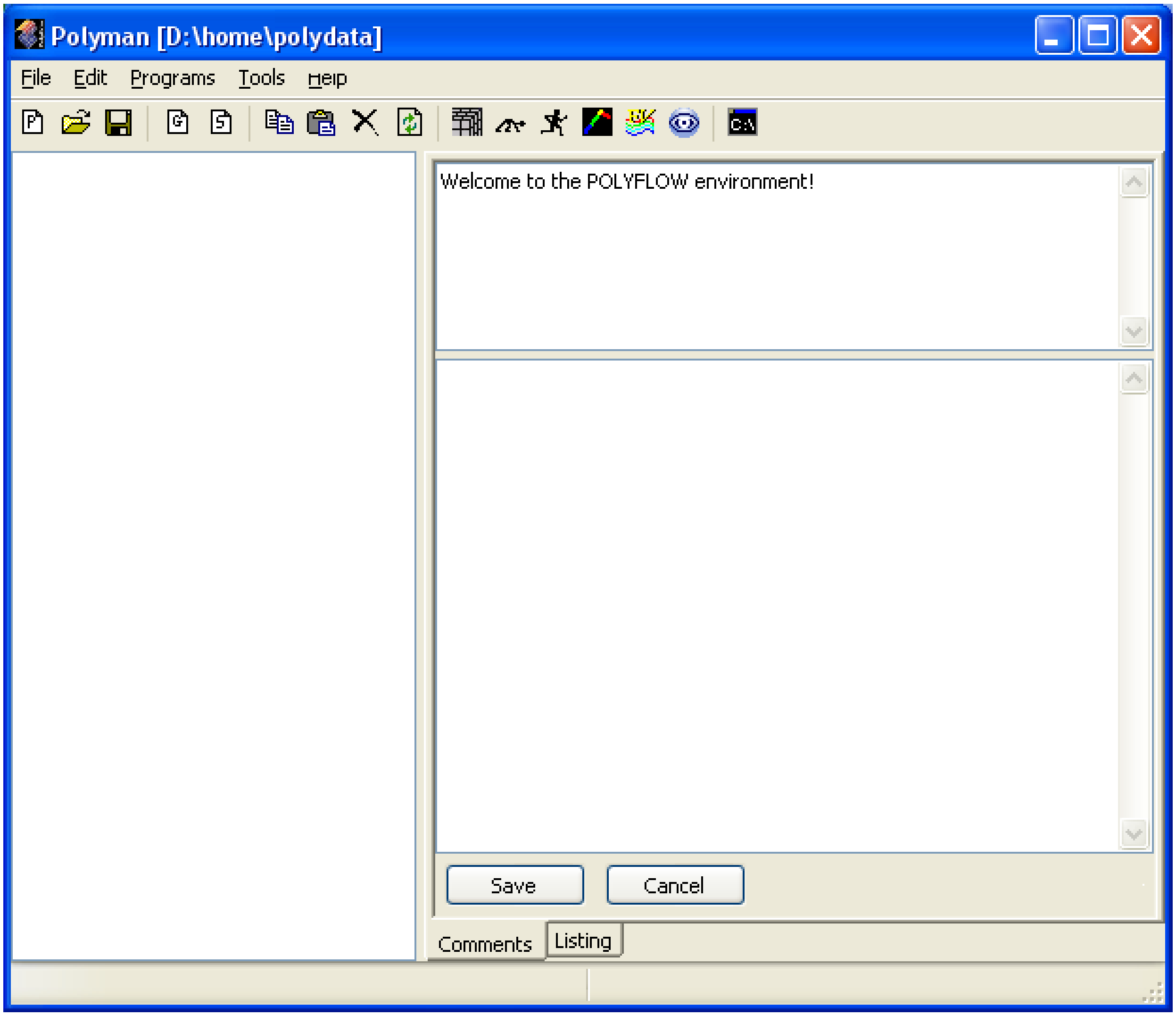
1. Enter `Sample` for the **Project name** and click **OK**.

   ANSYS POLYMAN creates the project and displays it in the list tree (see *Figure 4.4* (p. 65)), showing two branches: **Geometry** and **Simulations**. It creates a directory called `Sample` and a file called `Sample.prj` in your working directory.

   Note

   To reopen this project in a later ANSYS POLYMAN session, you can use the **File/Open...** menu item and select the **Sample.prj** file.

Figure 4.4 Initial Project List Tree



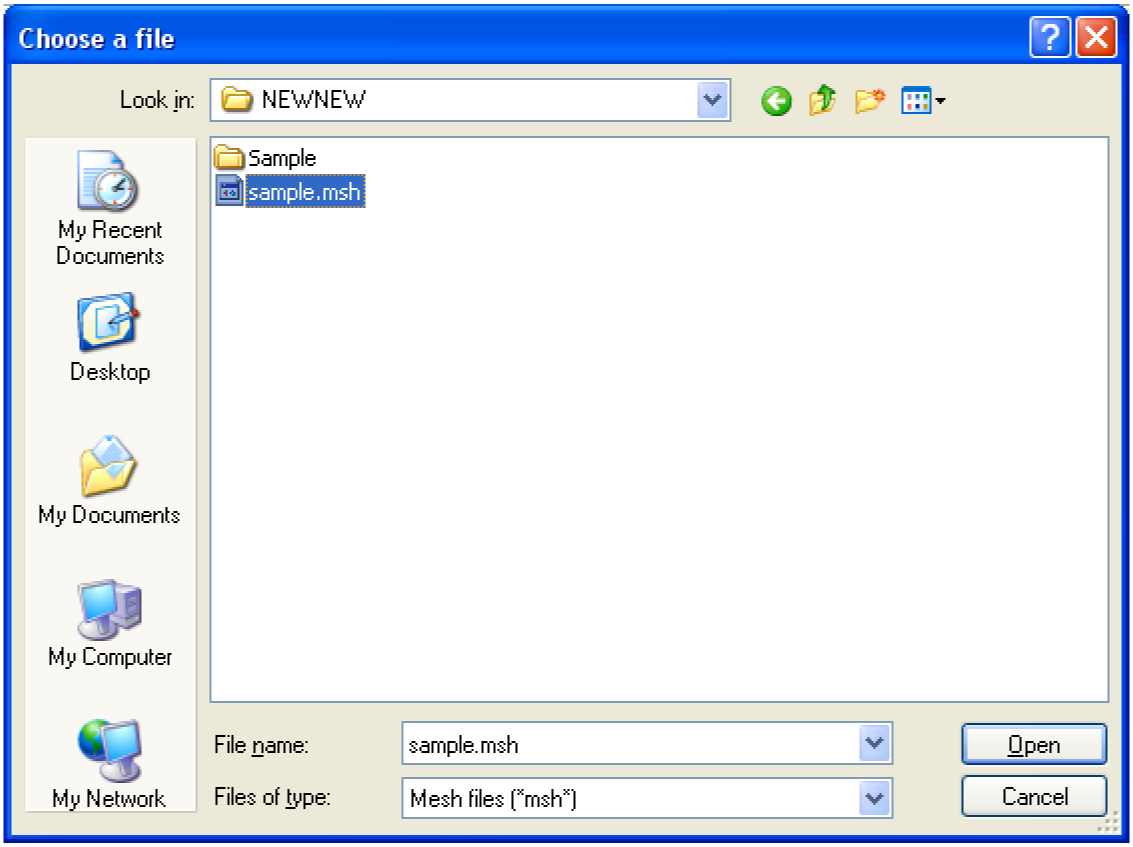
Welcome to the POLYFLOW environment!

2. Import the mesh file into the project by selecting the **File/Import/Polyflow mesh** menu item.

   File → **Import** → **Polyflow mesh**

3. In the **Choose a file** panel (*Figure 4.5* (p. 66)), select **sample.msh** and click **OK**.

Figure 4.5 Choose a file Panel



Choose a file

Look in: NEWNEW

Sample

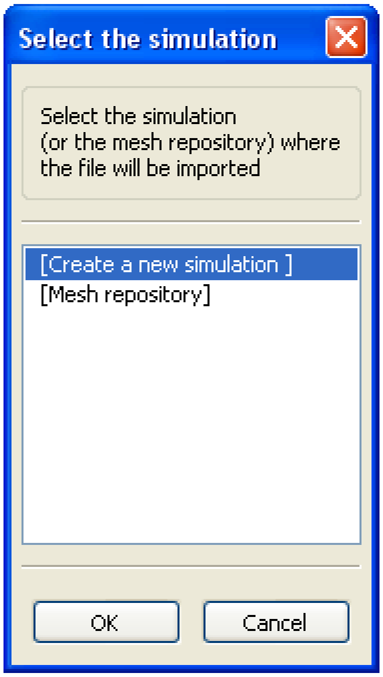
sample.msh

File name: sample.msh

Files of type: Mesh files (*msh*)

4. In the **Select the simulation** panel (*Figure 4.6* (p. 66)), select **[Create a new simulation]** and click **OK** to open the **Add new item** panel (*Figure 4.7* (p. 67))

Figure 4.6 Select the simulation Panel



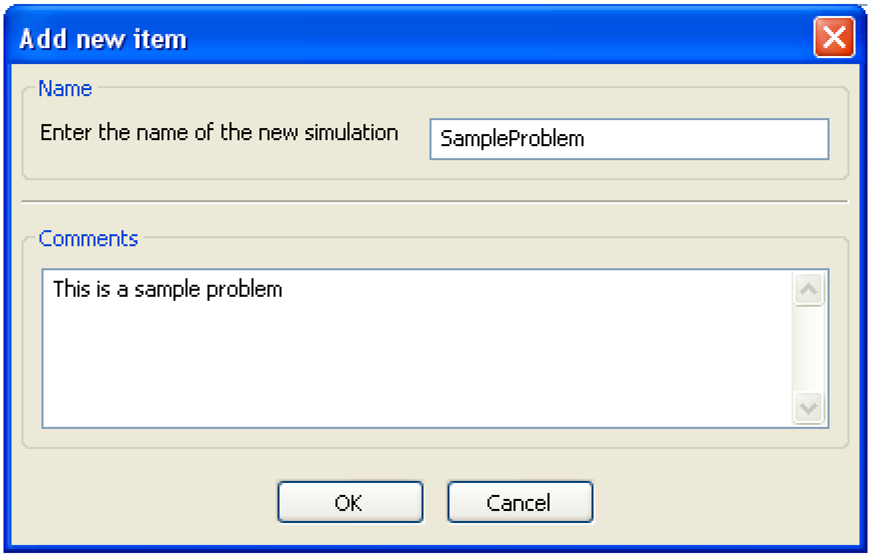
Select the simulation

Select the simulation (or the mesh repository) where the file will be imported

[Create a new simulation]

[Mesh repository]

Figure 4.7 Add new item Panel



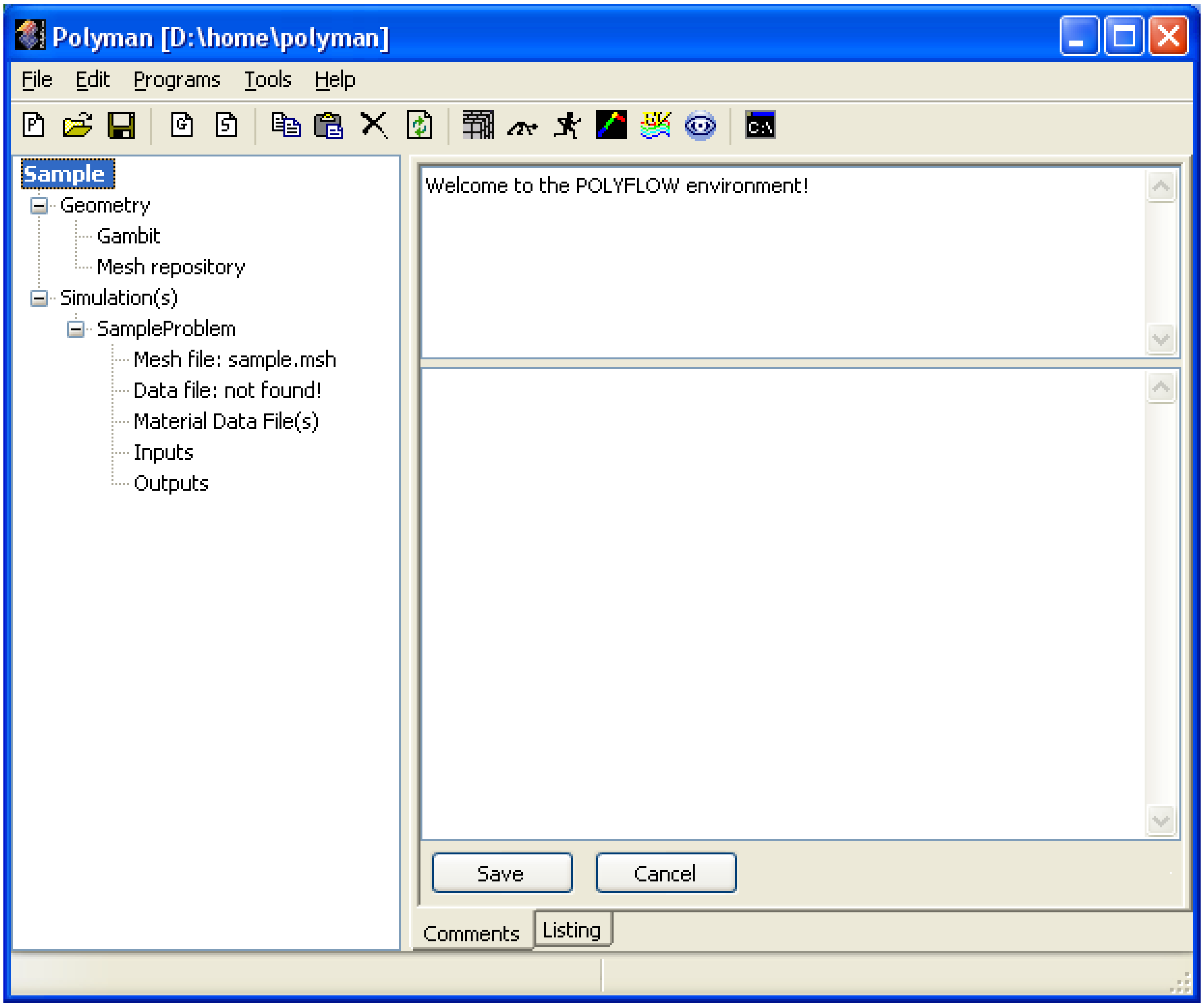
Enter the name of the new simulation: SampleProblem

This is a sample problem

5. Enter `SampleProblem` under **Name**, and an appropriate comment under **Comments**, as shown in *Figure 4.7* (p. 67), and click **OK**.

   The expanded project tree will appear as shown in *Figure 4.8* (p. 68).

Figure 4.8 Expanded Project List Tree



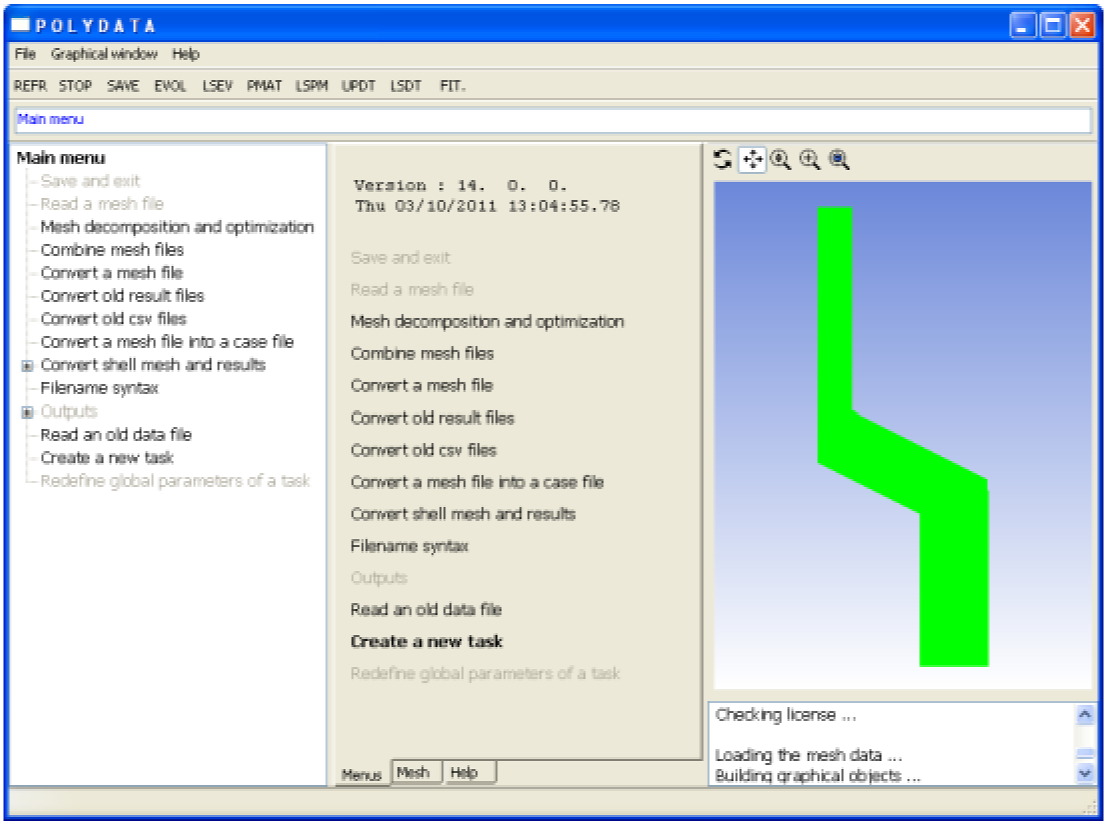
Welcome to the POLYFLOW environment!

4.4. Starting ANSYS POLYDATA and Reading the Mesh File

If you create your mesh in GAMBIT or a third-party CAD package, you will need to convert it before you read it into ANSYS POLYDATA. In this example, the mesh file has already been converted, so you can read the mesh file directly into ANSYS POLYDATA.

To start ANSYS POLYDATA and read the mesh file, double-click **Mesh file: sample.msh** in the project list tree. ANSYS POLYDATA will start up, read the mesh file, display a filled-in view of the domain (*Figure 4.9* (p. 69)), and highlight the **Create a new task** menu item in the ANSYS POLYDATA menu.

Figure 4.9 View of the Model in the Graphics Display Window



```
Version : 14.  0.  0.
Thu 03/10/2011 13:04:55.78
```

```
Checking license ...

Loading the mesh data ...
Building graphical objects ...
```

4.5. Creating a Task

You will next define a new task representing the 2.5D axisymmetric steady-state model, using the **Create a new task** menu item.

Create a new task

Select **F.E.M. task**, **Steady-state**, and **2D 1/2 axisymmetric geometry**. The **Current setup** (above the selected options) will be updated to reflect your selections. You can then select **Accept the current setup**. ANSYS POLYDATA will bring you to the next menu and highlight the **Create a sub-task** menu item.

4.6. Creating a Sub-Task and Specifying its Domain

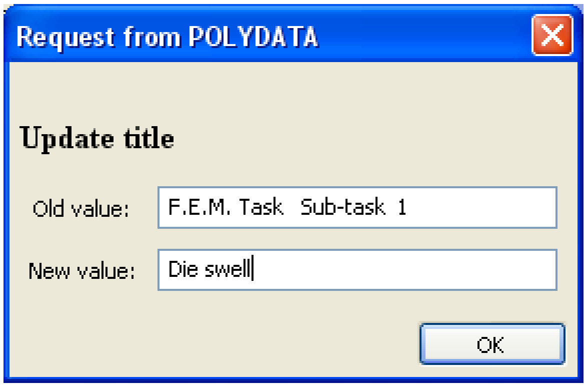
4.6.1. Creating a Sub-Task

Now you can define a sub-task for the isothermal flow, using the **Create a sub-task** menu item.

Create a sub-task

Select **Generalized Newtonian isothermal flow problem** and then enter the name `die swell` in the panel that ANSYS POLYDATA opens (*Figure 4.10* (p. 70)) and click **OK**. ANSYS POLYDATA will return to the main menu and highlight the **Domain of the sub-task** menu item.

Figure 4.10 Naming the Sub-Task



Request from POLYDATA

Update title

Old value: F.E.M. Task Sub-task 1

New value: Die swell

4.6.2. Specifying the Sub-Task's Domain of Definition

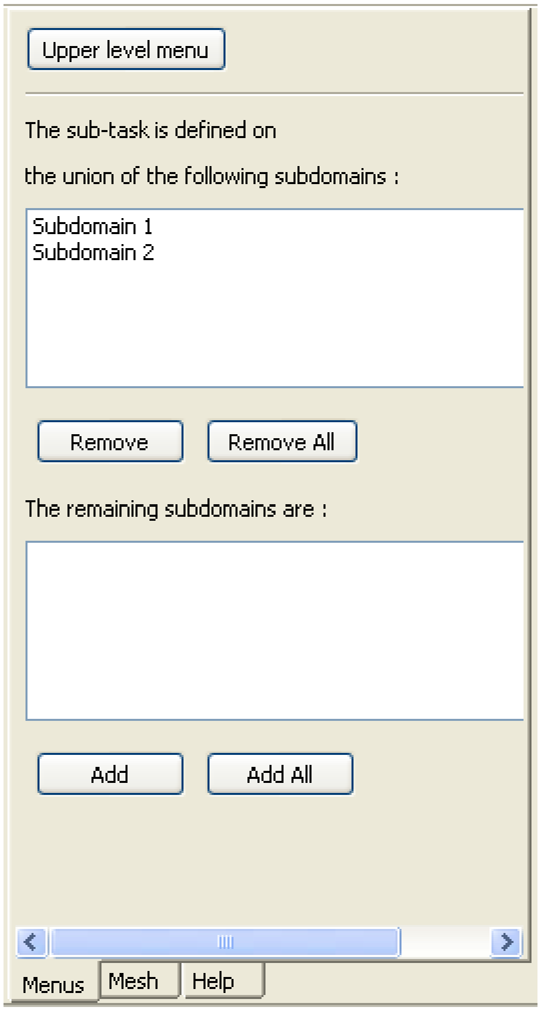
You will now define the domain where the sub-task applies, using the **Domain of the sub-task** menu item.

Domain of the sub-task

As mentioned earlier, the domain has been divided into two subdomains due to the presence of a free surface. In this problem, the isothermal flow sub-task applies to both subdomains, which is the default condition. You can simply accept the default setting (shown in *Figure 4.11* (p. 71)) by clicking on **Upper level menu** at the top of the panel.

ANSYS POLYDATA will return to the main menu and highlight the **Material data** menu item.

Figure 4.11 Defining the Domain of the Sub-Task


Upper level menu
The sub-task is defined on
the union of the following subdomains :
Subdomain 1
Subdomain 2
Remove
Remove All
The remaining subdomains are :
Add
Add All
Menus
Mesh
Help

4.7. Defining Material Properties for the Fluid

You will now specify the fluid properties, using the **Material data** menu item.

Material data

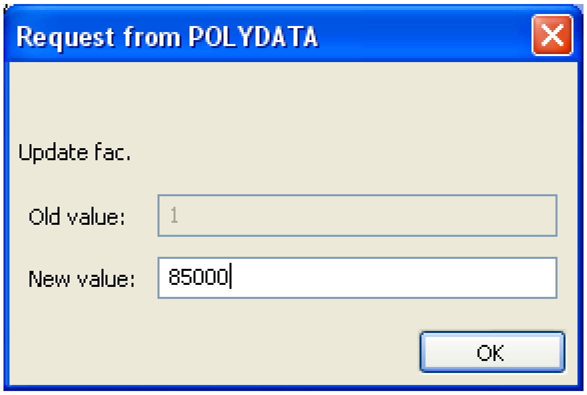
For this problem, you will define only the viscosity of the material. Select **Shear-rate dependence of viscosity**, and then choose the **Cross law**. You will need to specify three parameters for the Cross law: η_0, λ, and m. See *Cross Law* (p. 209) for details about the Cross Law and its applicability.

Specify the value for η_0.

Modify fac

Enter `85000` Poise as the **New value** (*Figure 4.12* (p. 72)) and click **OK**.

Figure 4.12 Specifying the Value of the First Parameter for the Cross Law



Update fac.

Old value: 1

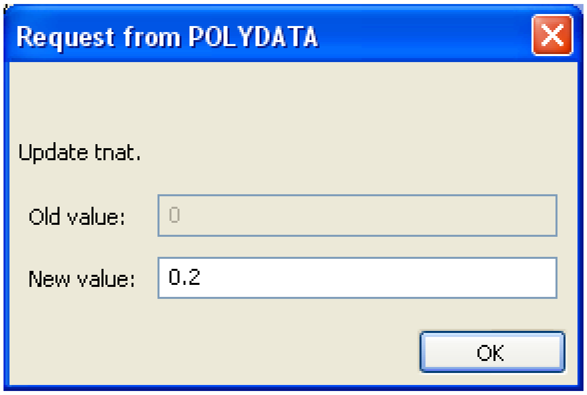
New value: 85000

Next, specify the value for λ.

Modify tnat

Enter `0.2` seconds as the **New value** (*Figure 4.13* (p. 72)) and click **OK**.

Figure 4.13 Specifying the Value of the Second Parameter for the Cross Law



Update tnat.

Old value: 0

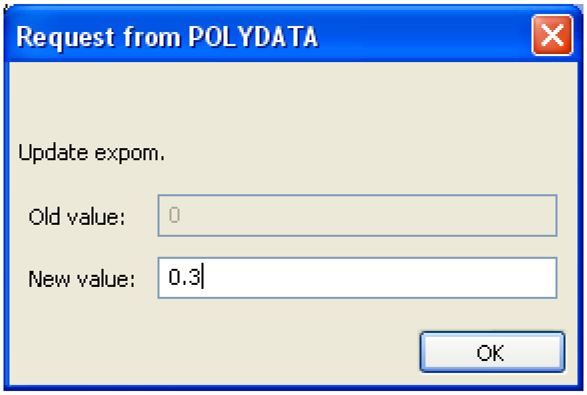
New value: 0.2

Finally, specify the value for m.

Modify expom

Enter `0.3` as the **New value** (*Figure 4.14* (p. 73)) and click **OK**.

Figure 4.14 Specifying the Value of the Third Parameter for the Cross Law



Request from POLYDATA

Update expom.

Old value: 0

New value: 0.3

Check that the values of the constants are correct at the top of the **Cross law** menu, and repeat the previous steps if you need to modify the constants again. Then click on **Upper level menu** three times to complete the material data specification. ANSYS POLYDATA will highlight the **Flow boundary conditions** menu item for you.

4.8. Defining Flow Boundary Conditions

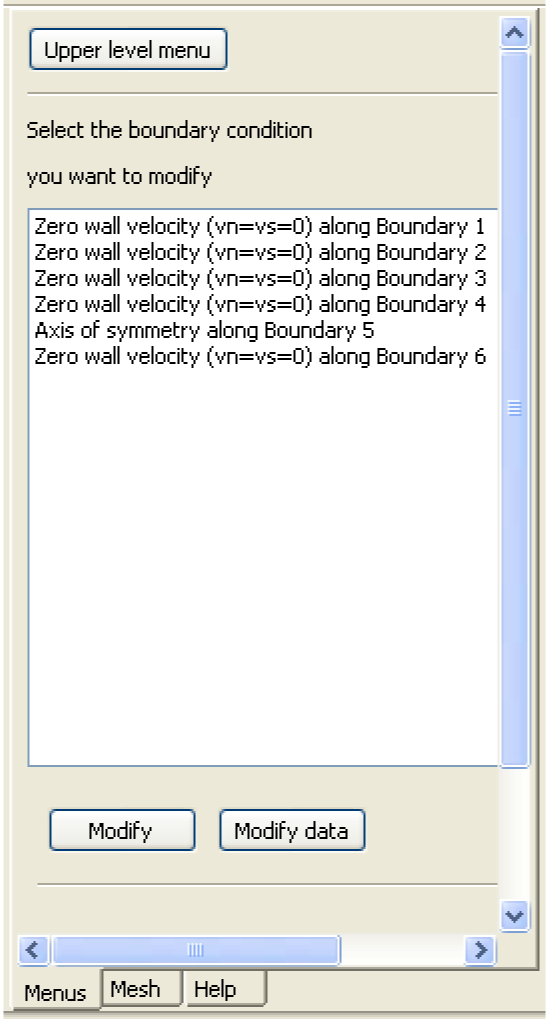
See *Figure 4.2* (p. 63) to determine where each boundary is located. Select each boundary and modify its conditions as needed. Set the conditions at each of the boundaries of the domain:

- boundary 1: flow inlet
- boundary 2: outer wall
- boundary 3: free surface
- boundary 4: flow exit
- boundary 5: symmetry axis
- boundary 6: rotating screw

Selecting the **Flow boundary conditions** menu item will open the panel shown in *Figure 4.15* (p. 74).

☰ **Flow boundary conditions**

Figure 4.15 Default Boundary Conditions



Select the boundary condition

you want to modify

Zero wall velocity (vn=vs=0) along Boundary 1
Zero wall velocity (vn=vs=0) along Boundary 2
Zero wall velocity (vn=vs=0) along Boundary 3
Zero wall velocity (vn=vs=0) along Boundary 4
Axis of symmetry along Boundary 5
Zero wall velocity (vn=vs=0) along Boundary 6

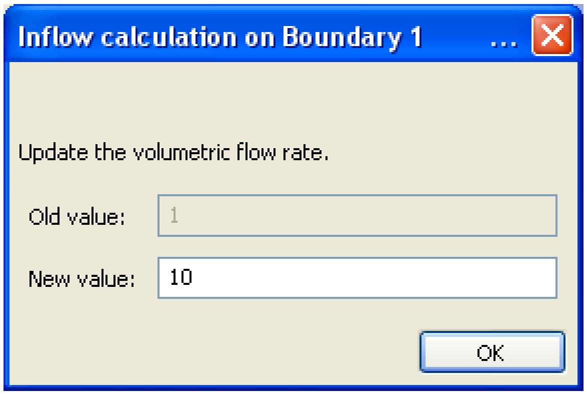
4.8.1. Flow Inlet

Set the conditions at the flow inlet (boundary 1).

1. Select the **Zero wall velocity (vn=vs=0) along Boundary 1** option in the **Flow boundary conditions** panel.
2. Click **Modify**.
3. Select the **Inflow** option from the list.

   ANSYS POLYDATA will prompt you for the volumetric flow rate or mass flow rate, as shown in Figure 4.16 (p. 75).
4. Enter `10` as the **New value** and click **OK**.

Figure 4.16 Specifying the Flow Rate at the Inlet



Inflow calculation on Boundary 1 ...

Update the volumetric flow rate.

Old value: 1

New value: 10

5. Specify the method and type of flow rate (mass or volumetric).

   *In this case, keep the default type, **Volumetric flow rate**.*

4.8.2. Outer Wall

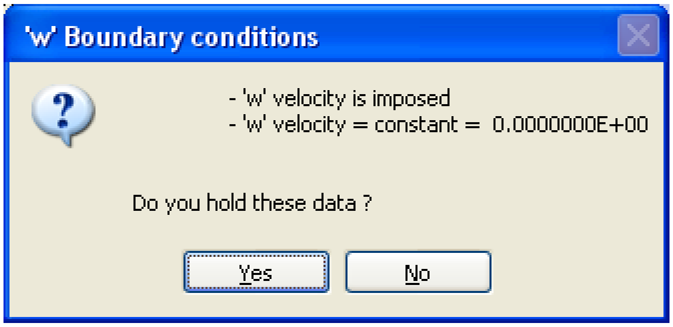
Set the conditions on the outer wall (boundary 2).

1. Select the **Zero wall velocity (vn=vs=0) along Boundary 2** (default) option.
2. Click **Modify**.
3. Select **Normal and tangential velocities imposed (vn, vs)**, which is the default setting from the list of choices.

   ANSYS POLYDATA will request input of the normal and tangential velocities.
4. Click **Upper level menu** to accept the default value of **0** for the normal velocity (**vn**).
5. Repeat the procedure to accept the default value of **0** for the tangential velocity (**vs**).

   ANSYS POLYDATA will then ask you to confirm that the angular velocity is equal to 0 (Figure 4.17 (p. 75)).
6. Click **Yes** to accept this default condition.

Figure 4.17 Confirming an Angular Velocity of 0 on the Outer Wall



'w' Boundary conditions

- 'w' velocity is imposed
- 'w' velocity = constant = 0.0000000E+00

Do you hold these data ?

4.8.3. Free Surface

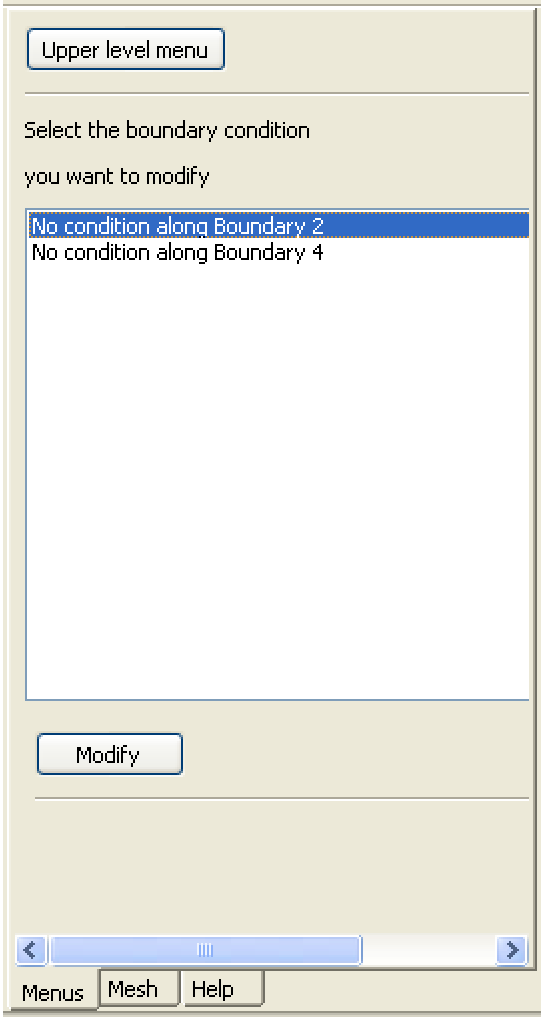
The free surface (boundary 3) requires an initial condition at its starting point, which in this case is located at the intersection of boundaries 2 and 3. The steps to set the conditions on the free surface are as follows:

1. Select the **Zero wall velocity (vn=vs=0) along Boundary 3** option.
2. Click **Modify**.
3. Select the **Free surface** option from the list of choices.

   In this case, you need to modify only the boundary conditions.
4. Select the **Boundary conditions on the moving surface** option.

   ANSYS POLYDATA will prompt you for the boundary on which the free surface position will be imposed (Figure 4.18 (p. 76)).

Figure 4.18 Specifying the Boundary on which the Free Surface Position is Imposed



Select the boundary condition

you want to modify

No condition along Boundary 2
No condition along Boundary 4

5. Select **Boundary 2** and click **Modify**.
6. Select the **Position imposed** option.
7. Click **Upper level menu** three times to return to the other boundary conditions.

4.8.4. Flow Exit

For the flow exit, a uniform velocity profile with no forces acting is assumed. The steps to set the conditions at the flow exit (boundary 4) are as follows:

1. Select the **Zero wall velocity (vn=vs=0) along Boundary 4** option.
2. Click **Modify**.
3. Select the **Normal and tangential forces imposed (fn, fs)** option.

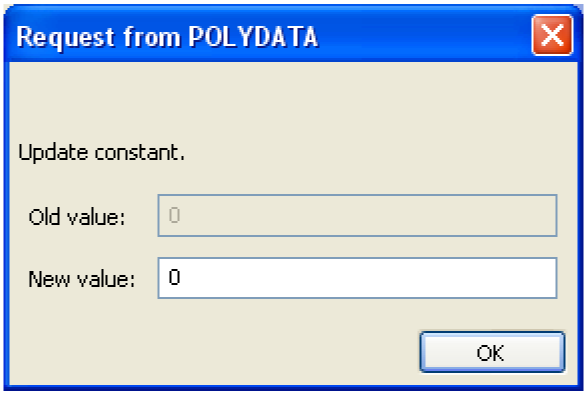
   ANSYS POLYDATA will then request input of the normal and tangential forces.
4. Click **Upper level menu** to accept the default value of **0** for the normal force (**fn**).
5. Repeat the procedure to accept the default value of **0** for the tangential velocity (**fs**).

   *ANSYS POLYDATA will ask you to confirm that the rotational velocity is equal to **0**.*
6. Click **No**.

   *Here, you are specifying a rotational force of **0** and not a rotational velocity of **0**.*
7. Select **'w' force imposed** followed by **'w' force = constant**.

   ANSYS POLYDATA will then prompt you for the value of the rotational force (Figure 4.19 (p. 77)).

Figure 4.19 Accepting a Rotational Force of 0 at the Flow Exit



Request from POLYDATA

Update constant.

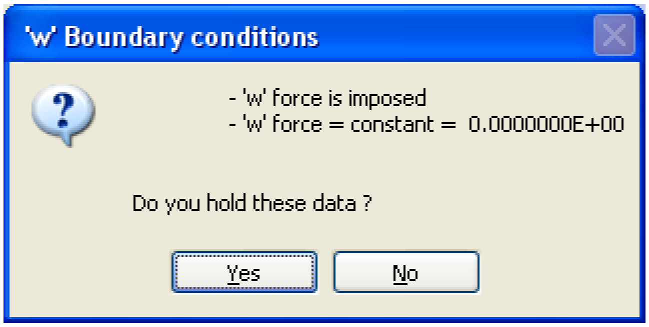
Old value: 0

New value: 0

8. Click **OK** to Accept the default value of **0**.

   *ANSYS POLYDATA will ask you to confirm that the rotational force is **0** (Figure 4.20 (p. 77)).*
9. Click **Yes**.

Figure 4.20 Confirming a Rotational Force of 0 at the Flow Exit



'w' Boundary conditions

- 'w' force is imposed
- 'w' force = constant = 0.0000000E+00

Do you hold these data ?

4.8.5. Symmetry Axis

For the symmetry axis (boundary 5), the default condition of **Axis of symmetry** is correct, so there are no inputs required from you.

4.8.6. Rotating Screw

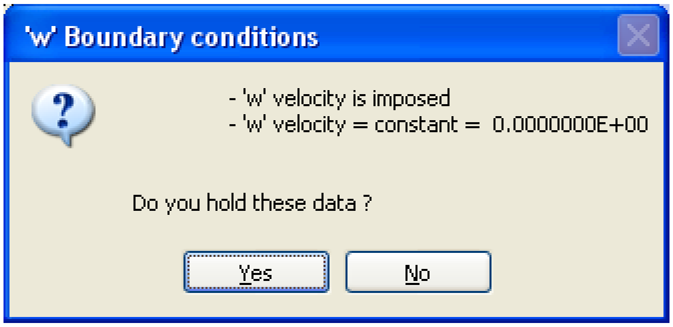
For the rotating screw (boundary 6), you will define a rotational velocity that increases linearly with respect to the distance from the axis of symmetry: $w = 6.2832r$. The steps to set the conditions for the rotating screw are as follows:

1. Select the **Zero wall velocity (vn=vs=0) along Boundary 6** option.
2. Click **Modify**.
3. Select the **Normal and tangential velocities imposed (vn, vs)** (default) option.

   ANSYS POLYDATA will then request input of the normal and tangential velocities.
4. Click **Upper level menu** to accept the default value of **0** for the normal velocity (**vn**).
5. Repeat the procedure to accept the default value of **0** for the tangential velocity (**vs**).

   ANSYS POLYDATA will then ask you to confirm that the rotational velocity is equal to 0 (Figure 4.21 (p. 78)).

Figure 4.21 Rejecting the Default Rotational Velocity for the Screw



'w' Boundary conditions

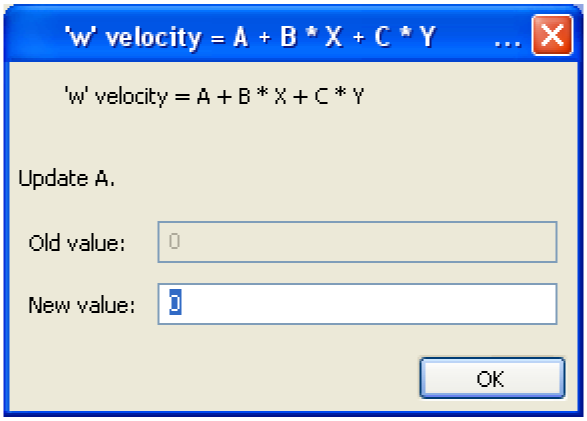
- 'w' velocity is imposed
- 'w' velocity = constant = 0.0000000E+00

Do you hold these data ?

6. Click **No** as you want to specify a linear function for rotational velocity.
7. Select **Velocity w imposed** followed by **'w' velocity = linear function of coordinates**.

   ANSYS POLYDATA will prompt you for the coefficients of the rotational velocity function, $w = A + BX + CY$, where X denotes the radial direction and Y denotes the axial direction. First, you will be prompted for the value of A (Figure 4.22 (p. 79)).

Figure 4.22 Setting Coefficient A for Rotational Velocity of the Screw

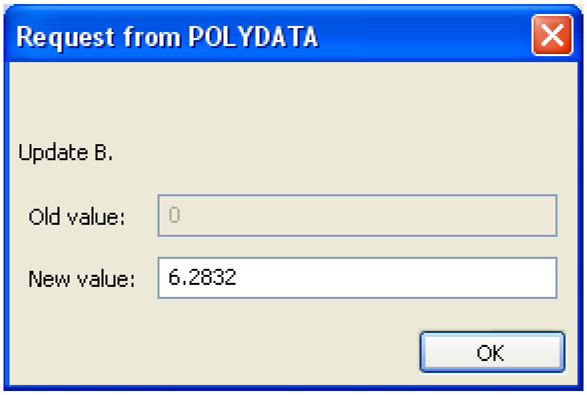

'w' velocity = A + B * X + C * Y ...
'w' velocity = A + B * X + C * Y
Update A.
Old value:
0
New value:
0
OK

8. Click **OK** to accept the default value of **0**.

   ANSYS POLYDATA will prompt you for the value of B (*Figure 4.23* (p. 79)).

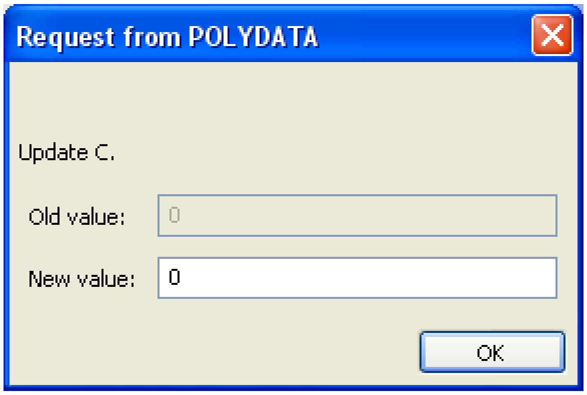
9. Enter `6.2832` as the **New value** and click **OK**.

Figure 4.23 Setting Coefficient B for Rotational Velocity of the Screw



10. When ANSYS POLYDATA will prompt you for the value of C (*Figure 4.24* (p. 79)), accept the default value of **0** by clicking **OK**.

Figure 4.24 Setting Coefficient C for Rotational Velocity of the Screw



ANSYS POLYDATA will ask you to confirm that the rotational velocity equation is correct.

11. Check that the values of the coefficients are correct and then click **Yes**.
12. Click **Upper level menu** at the top of the panel to return to the main menu.

    *ANSYS POLYDATA will highlight the **Global remeshing** menu item for you.*

4.9. Defining the Remeshing

This model includes a free surface, the position of which is unknown. A portion of the mesh will be affected by the free surface's changes in position, so a remeshing technique will be applied on this part of the mesh. Since the free surface is entirely contained within subdomain 2, the remeshing will apply only to subdomain 2. You will define the remeshing parameters with the **Global remeshing** menu item.

Global remeshing

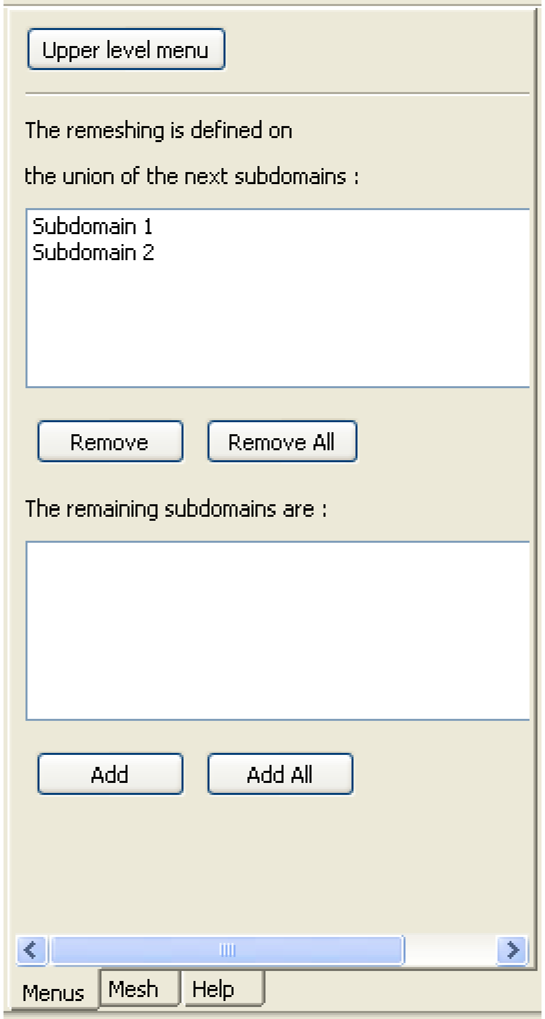
4.9.1. Specifying the Region To Be Remeshed

To specify the region where the remeshing is to be performed, choose the **1-st local remeshing** menu item.

1-st local remeshing

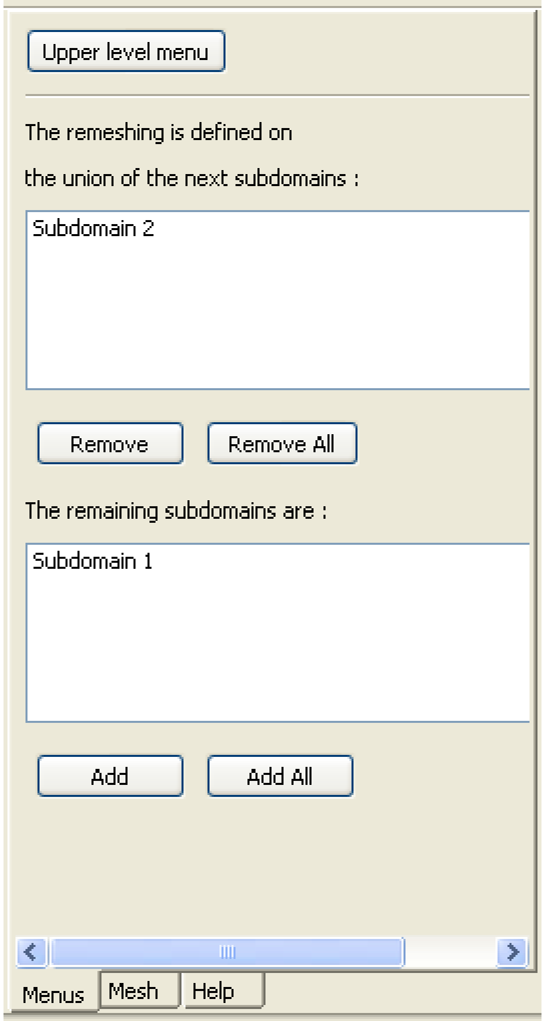
By default, the remeshing is defined on both subdomains, as shown in *Figure 4.25* (p. 81).

Figure 4.25 Default Definition of the Region to be Remeshed


Upper level menu
The remeshing is defined on
the union of the next subdomains :
Subdomain 1
Subdomain 2
Remove
Remove All
The remaining subdomains are :
Add
Add All
Menus
Mesh
Help

Select **Subdomain 1** and click **Remove**, and it will be moved from the top list to the bottom list, as shown in *Figure 4.26* (p. 82), indicating that only subdomain 2 will be remeshed. Click **Upper level menu** at the top of the panel to return to the main menu and continue the remeshing definition. ANSYS POLYDATA will highlight the **Method of Spines** menu item for you.

Figure 4.26 Defining the Remeshing for Subdomain 2 Only



The remeshing is defined on

the union of the next subdomains :

Subdomain 2

The remaining subdomains are :

Subdomain 1

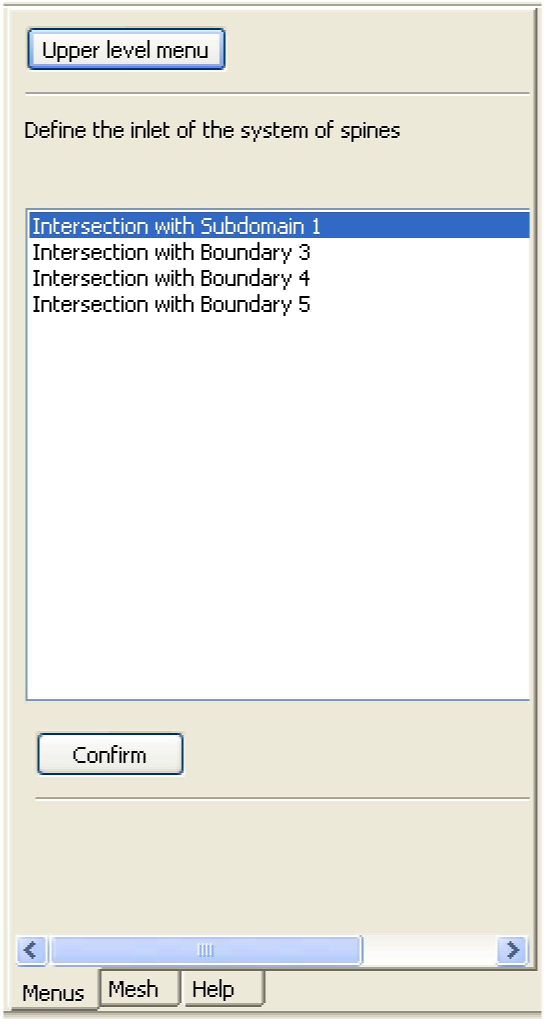
4.9.2. Specifying the Parameters for the System of Spines

The default remeshing method in 2D is called the method of spines. In this method, mesh nodes are organized along lines of remeshing (called spines). ANSYS POLYDATA requires the specification of the first and last spines that the fluid encounters (inlet of spines and outlet of spines, respectively). In this case, the inlet of spines is the intersection of subdomain 2 with subdomain 1, and the outlet of spines is the intersection of subdomain 2 with the flow exit (boundary 4). You will define these parameters using the **Method of Spines** menu item.

Method of Spines

To specify the inlet for the system of spines, select **Intersection with subdomain 1** and click **Confirm** (*Figure 4.27* (p. 83)).

Figure 4.27 Specifying the Inlet for the System of Spines

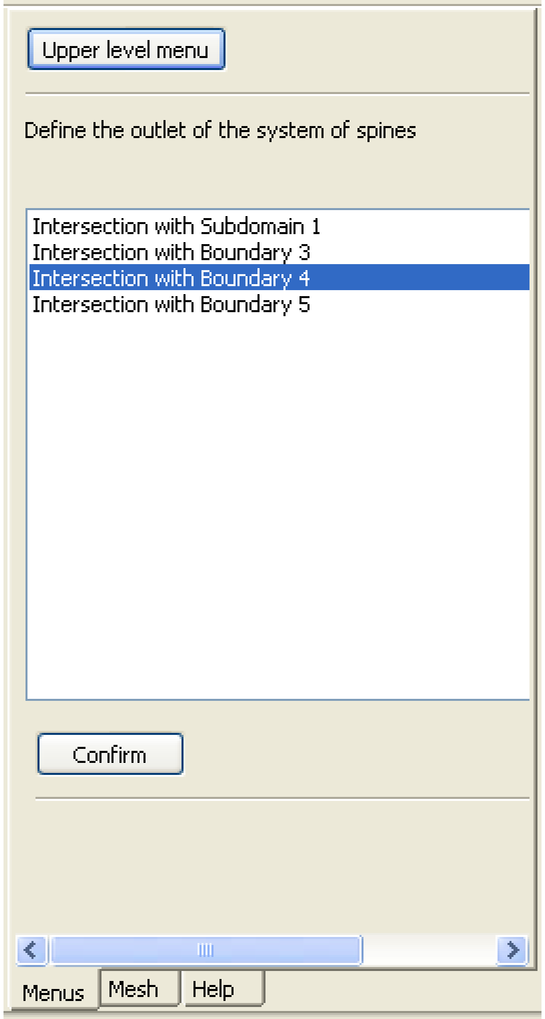


Define the inlet of the system of spines

Intersection with Subdomain 1
Intersection with Boundary 3
Intersection with Boundary 4
Intersection with Boundary 5

To specify the outlet for the system of spines, select **Intersection with boundary 4** and click **Confirm** (*Figure 4.28* (p. 84)). Then click on **Upper level menu** three times to return to the appropriate menu for the next step.

Figure 4.28 Specifying the Outlet for the System of Spines


Upper level menu
Define the outlet of the system of spines
Intersection with Subdomain 1
Intersection with Boundary 3
Intersection with Boundary 4
Intersection with Boundary 5
Confirm
Menus
Mesh
Help

4.10. Assigning the Stream Function Value

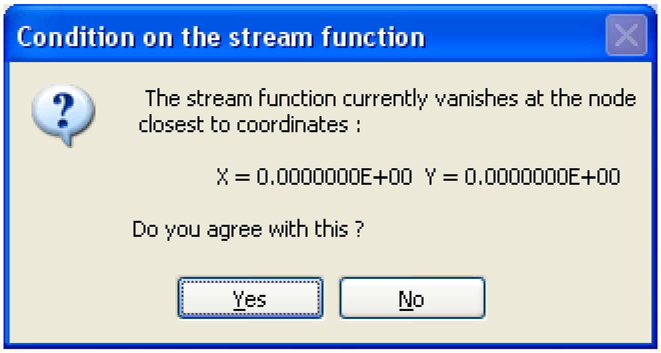
During the ANSYS POLYFLOW calculation, once the velocity field is known, the stream function is computed automatically. This calculation requires a reference point where the stream function vanishes, so that ANSYS POLYFLOW will know which stream function values are positive and which are negative. By default, ANSYS POLYDATA chooses the location (0,0), but you can change the location using the **Assign the stream function** menu item.

☰ Assign the stream function

In this case, you will set the vanishing point to be the location where the stationary outer wall (boundary 2) meets the flow inlet (boundary 1), since the stream function is equal to zero at this point (5,0).

Choose **Condition on the stream function for field 1** and click **No** when ANSYS POLYDATA prompts you to confirm the default location (*Figure 4.29* (p. 85)).

Figure 4.29 Rejecting the Default Vanishing Point for the Stream Function



Condition on the stream function

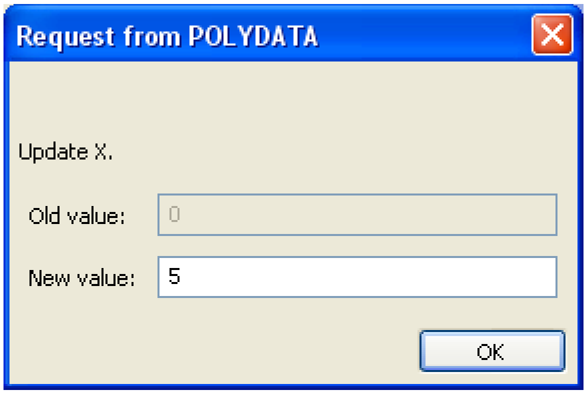
The stream function currently vanishes at the node closest to coordinates :

X = 0.0000000E+00 Y = 0.0000000E+00

Do you agree with this ?

Then enter 5 as the **New value for X**, and click **OK** (*Figure 4.30* (p. 85)).

Figure 4.30 Specifying the X Coordinate of the Vanishing Point



Request from POLYDATA

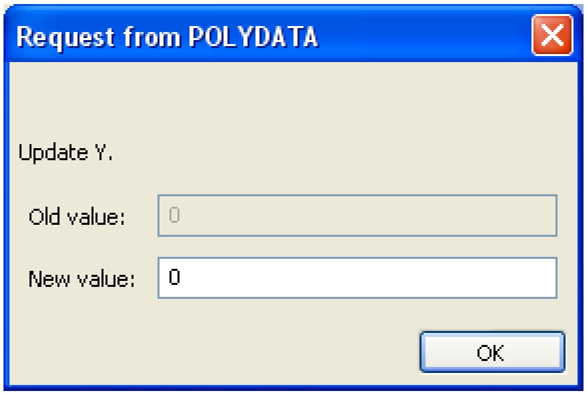
Update X.

Old value: 0

New value: 5

Accept the default value of 0 for **Y** by clicking **OK** (*Figure 4.31* (p. 85)).

Figure 4.31 Specifying the Y Coordinate of the Vanishing Point



Request from POLYDATA

Update Y.

Old value: 0

New value: 0

Finally, click **Upper level menu** twice to return to the top-level menu.

Accept the default value of 0 for **Y** by clicking **OK**. Click **Upper level menu** twice to return to the top-level menu.

4.11. Define the Units for Simulation

You need to specify the units for CFD-Post required for simulation. The default unit is **metric_MKSA+Kelvin**. The procedure to set the units is as follows:

1. Select **Outputs** from the top-level ANSYS POLYDATA menu.

   Outputs

2. Click **Set units for CFD-Post** .

   Set units for CFD-Post

   *This will display the **Change System of Units for CFD-Post** option.*

3. Select **Modify system of Units**.

   Modify system of Units

4. Select **Set to metric_cm/g/s/A+Celsius**.

   Set to metric_cm/g/s/A+Celsius

5. Click **Upper level menu** option twice to return to the **Save and exit** option.

   Upper level menu

4.12. Saving the Data File and Exiting from ANSYS POLYDATA

Now that you have finished defining your model in ANSYS POLYDATA, you have to save the information to a data file that will be read by ANSYS POLYFLOW. To save the data file and exit from ANSYS POLYDATA, select the **Save and exit** menu item.

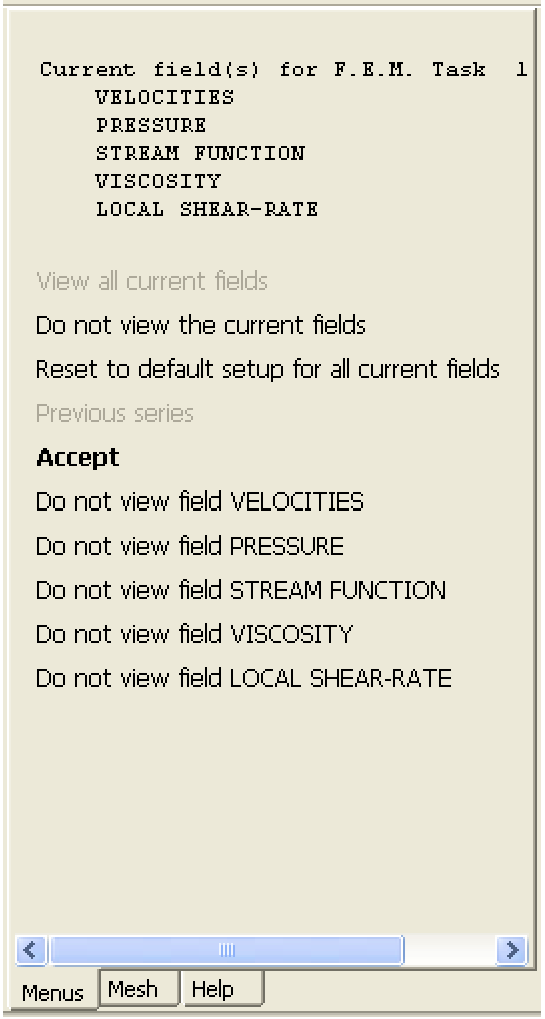
Save and exit

Note

When not started from ANSYS POLYMAN, ANSYS POLYDATA asks for the name of the new data file and by default, it creates a data file named `dat`.

ANSYS POLYDATA will ask you to confirm the fields that are to be saved to the results file for postprocessing (*Figure 4.32* (p. 87)).

Figure 4.32 Accepting the Default Fields to be Saved



```
Current field(s) for F.E.M. Task  1
    VELOCITIES
    PRESSURE
    STREAM FUNCTION
    VISCOSITY
    LOCAL SHEAR-RATE
```

View all current fields

Do not view the current fields

Reset to default setup for all current fields

Previous series

Accept

Do not view field VELOCITIES

Do not view field PRESSURE

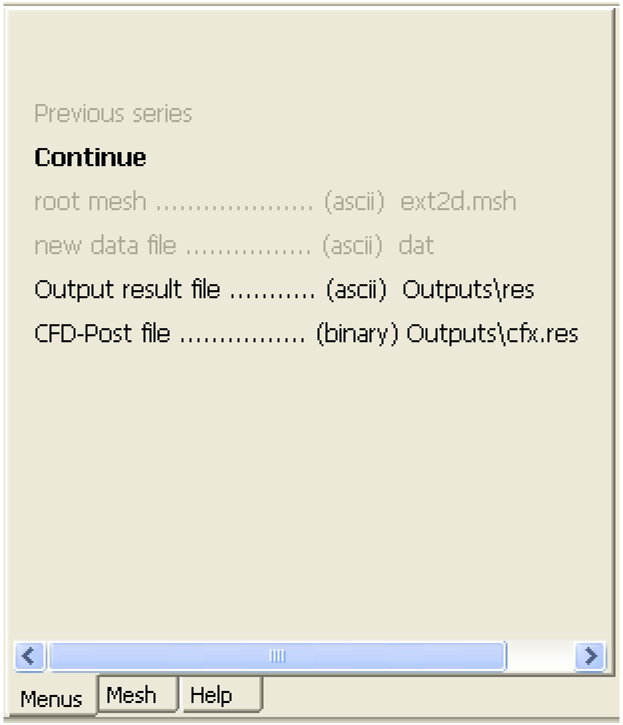
Do not view field STREAM FUNCTION

Do not view field VISCOSITY

Do not view field LOCAL SHEAR-RATE

Select **Accept** to confirm that the default **Current fields** are correct. Then select **Continue** to accept the default names of the mesh and results files that are to be saved for postprocessing (*Figure 4.33* (p. 88)).

Figure 4.33 Accepting the Default Names for the Files to be Saved



Previous series

Continue

root mesh .... (ascii) ext2d.msh

new data file .... (ascii) dat

Output result file .... (ascii) Outputs\res

CFD-Post file .... (binary) Outputs\cfx.res

4.13. Calculating a Solution with ANSYS POLYFLOW

Now that the problem definition with ANSYS POLYDATA is complete, you can calculate a solution with ANSYS POLYFLOW.

To run ANSYS POLYFLOW on the data file you just created, simply select the **Programs/Polyflow** menu item from the ANSYS POLYMAN menu bar.

Programs → **Polyflow**

ANSYS POLYFLOW will calculate a solution for the data file, and write information regarding the calculation and convergence to a listing file called `listingFile`. You can select **Tools/View listing** to view the listing file and check that the solution is converged before proceeding to examine the results.

Tools → **View listing**

4.14. Examining the Results with CFD-Post

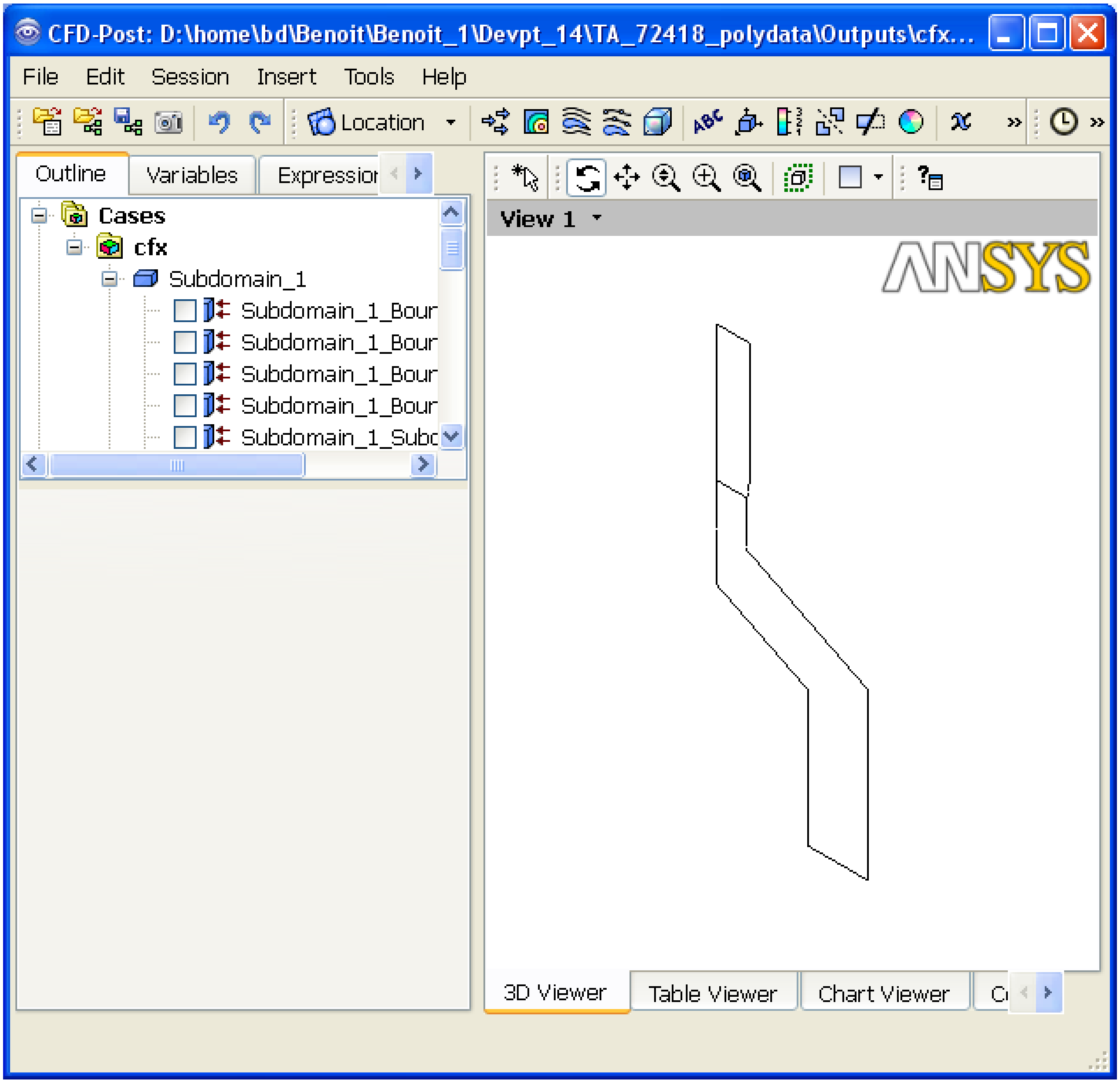
4.14.1. Starting CFD-Post and Reading the Results

To start CFD-Post and read the results files saved by ANSYS POLYFLOW, double-click the CFD-Post file: `cfx.resf` under **Outputs** in the ANSYS POLYMAN list tree.

CFD-Post will read the mesh information, then the solution fields that were saved to the results file.

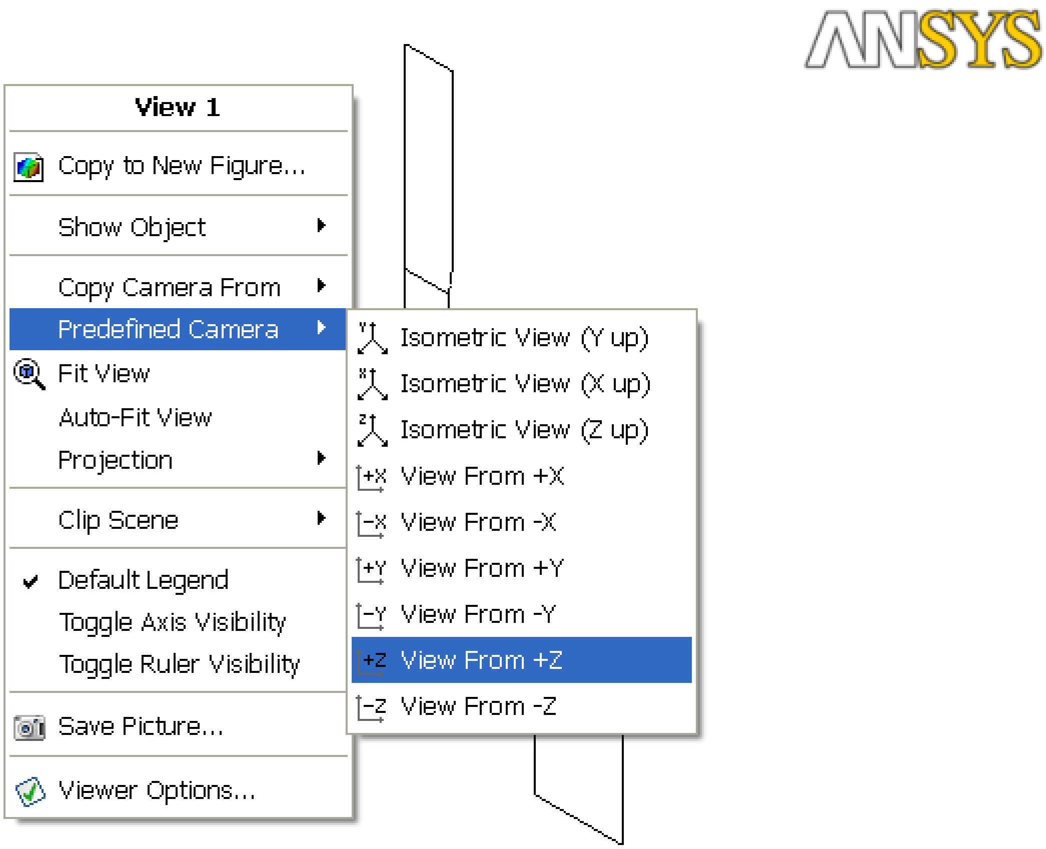
The CFD-Post environment is displayed in *Figure 4.34* (p. 89).

Figure 4.34 The CFD-Post Environment



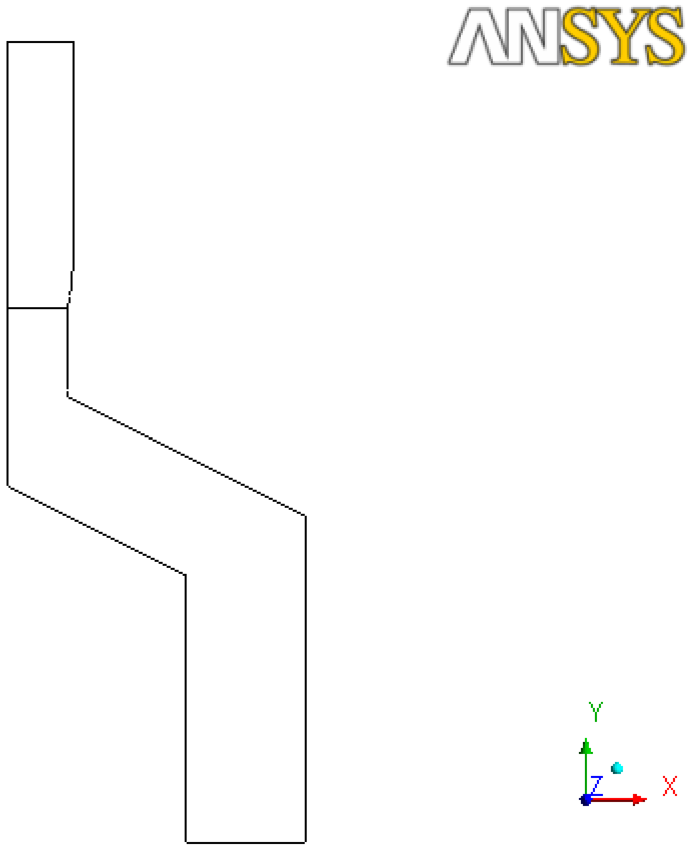
To align the view, right-click the graphics window to display the context menu, then choose **Predefined Camera** and **View From +Z**.

Figure 4.35 Aligning the View Using the Context Menu



The new view alignment is displayed in the graphics window (*Figure 4.36* (p. 91)).

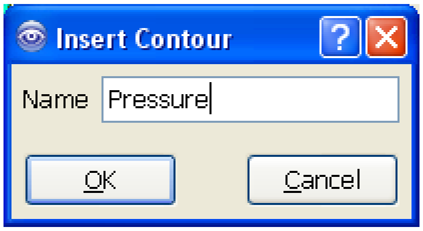
Figure 4.36 Viewing the New Alignment of the Geometry



4.14.2. Displaying Contours of Pressure

To display the contours of static pressure, click the **Contour** button (🖼). In the **Insert Contour** dialog box that opens (*Figure 4.37* (p. 91)), enter `Pressure` for **Name** and click **OK**.

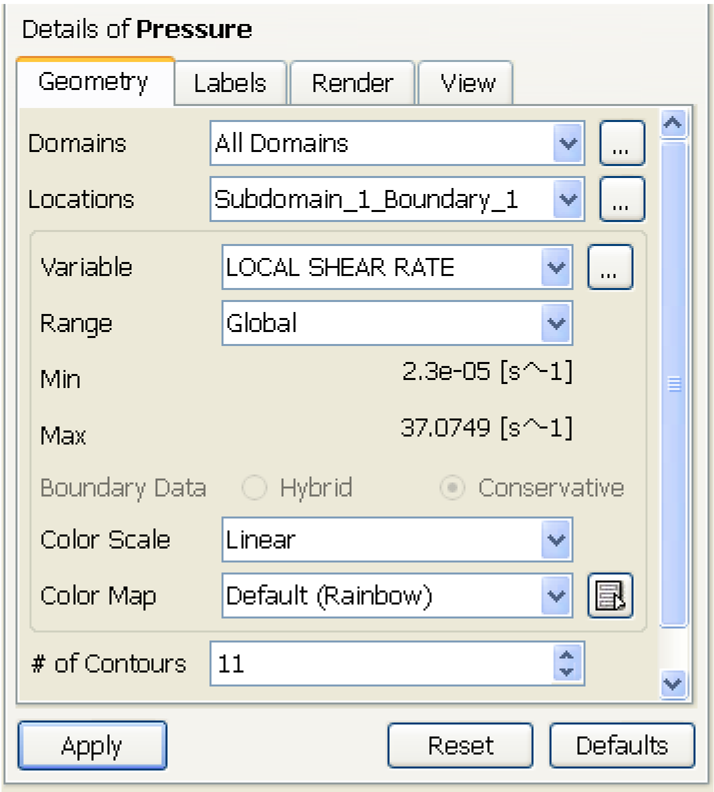
Figure 4.37 The Insert Contour Dialog Box



Name: Pressure

In the **Geometry** tab of the **Details of Pressure** windows (*Figure 4.38* (p. 92)), click the **Location editor** button (...) to the right of the **Locations** drop-down list.

Figure 4.38 Selecting Domains



Details of **Pressure**

Domains: All Domains

Locations: Subdomain_1_Boundary_1

Variable: LOCAL SHEAR RATE

Range: Global

Min: 2.3e-05 [s^-1]

Max: 37.0749 [s^-1]

Boundary Data: ○ Hybrid ● Conservative

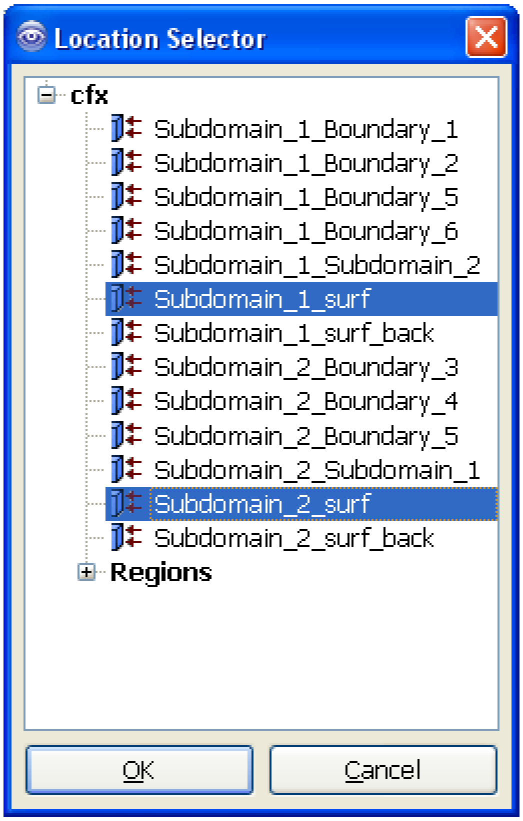
Color Scale: Linear

Color Map: Default (Rainbow)

# of Contours: 11

Select **Subdomain_1_surf** and **Subdomain_2_surf** in the **Location Selector** dialog box (*Figure 4.39* (p. 93)) and click **OK**.

Figure 4.39 The Location Selector Dialog Box

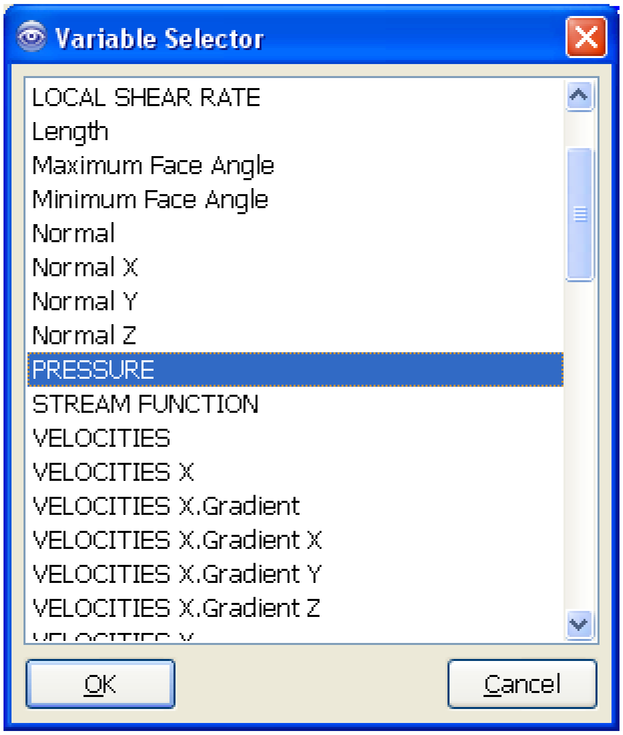


cfx
- Subdomain_1_Boundary_1
- Subdomain_1_Boundary_2
- Subdomain_1_Boundary_5
- Subdomain_1_Boundary_6
- Subdomain_1_Subdomain_2
- Subdomain_1_surf
- Subdomain_1_surf_back
- Subdomain_2_Boundary_3
- Subdomain_2_Boundary_4
- Subdomain_2_Boundary_5
- Subdomain_2_Subdomain_1
- Subdomain_2_surf
- Subdomain_2_surf_back

Regions

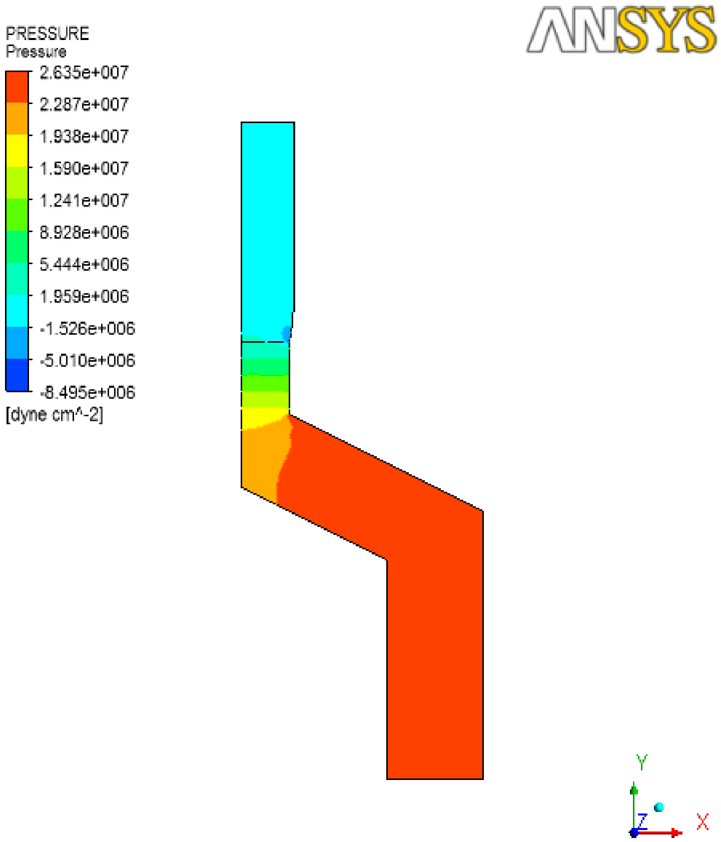
In the **Geometry** tab of the **Details of Pressure** window, click the **More Variables** button (...) to the right of the **Variable** drop-down list. Then select **PRESSURE** in the **Variable Selector** dialog box (*Figure 4.40* (p. 94)) and click **OK**.

Figure 4.40 The Variable Selector Dialog Box



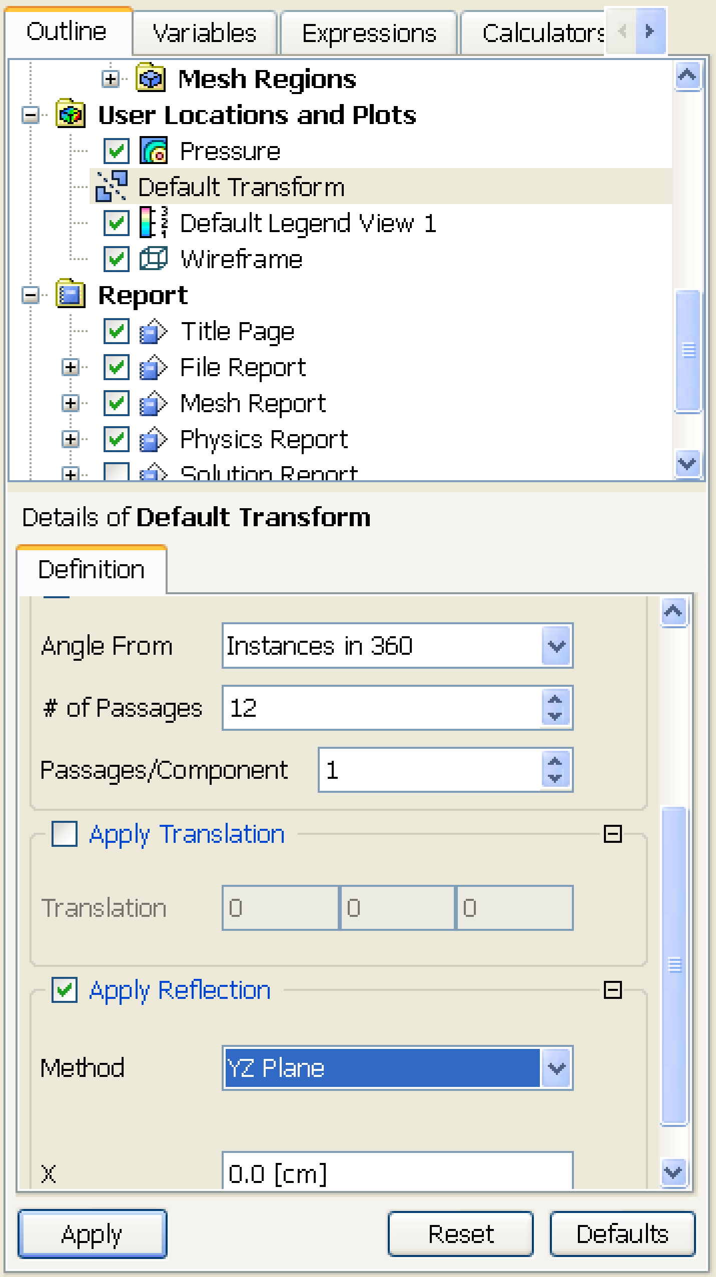
Click the **Apply** button in the **Details of Pressure** window to display the pressure contours in the graphic display window (*Figure 4.41* (p. 95)).

Figure 4.41 Pressure Contours



Double-click the **Default Transform** node in the **Outline** tab (*Figure 4.42* (p. 96)). Then, in the **Definition** tab of the **Details of Default Transform** window, disable the **Instancing Info From Domain** option and the **Apply Rotation** option. Next, enable the **Apply Reflection** option and select **YZ Plane** from the **Method** drop-down list in the related group box. Click **Apply** and then the **Fit View** button (🔍) in the graphics toolbar to display the contours with reflection (*Figure 4.43* (p. 97)).

Figure 4.42 Mirroring a Domain



Details of **Default Transform**

Angle From: Instances in 360

# of Passages: 12

Passages/Component: 1

□ Apply Translation

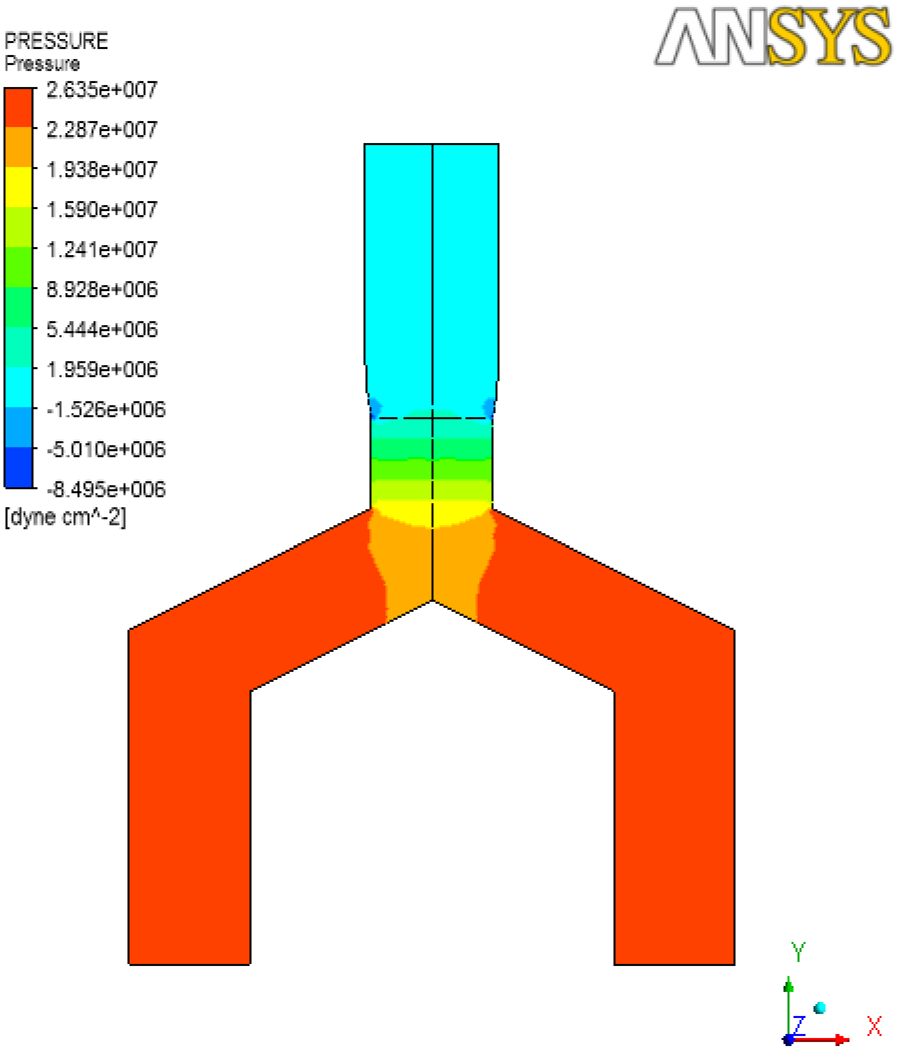
Translation: 0 0 0

☑ Apply Reflection

Method: YZ Plane

X: 0.0 [cm]

Figure 4.43 Pressure Contours with Reflection



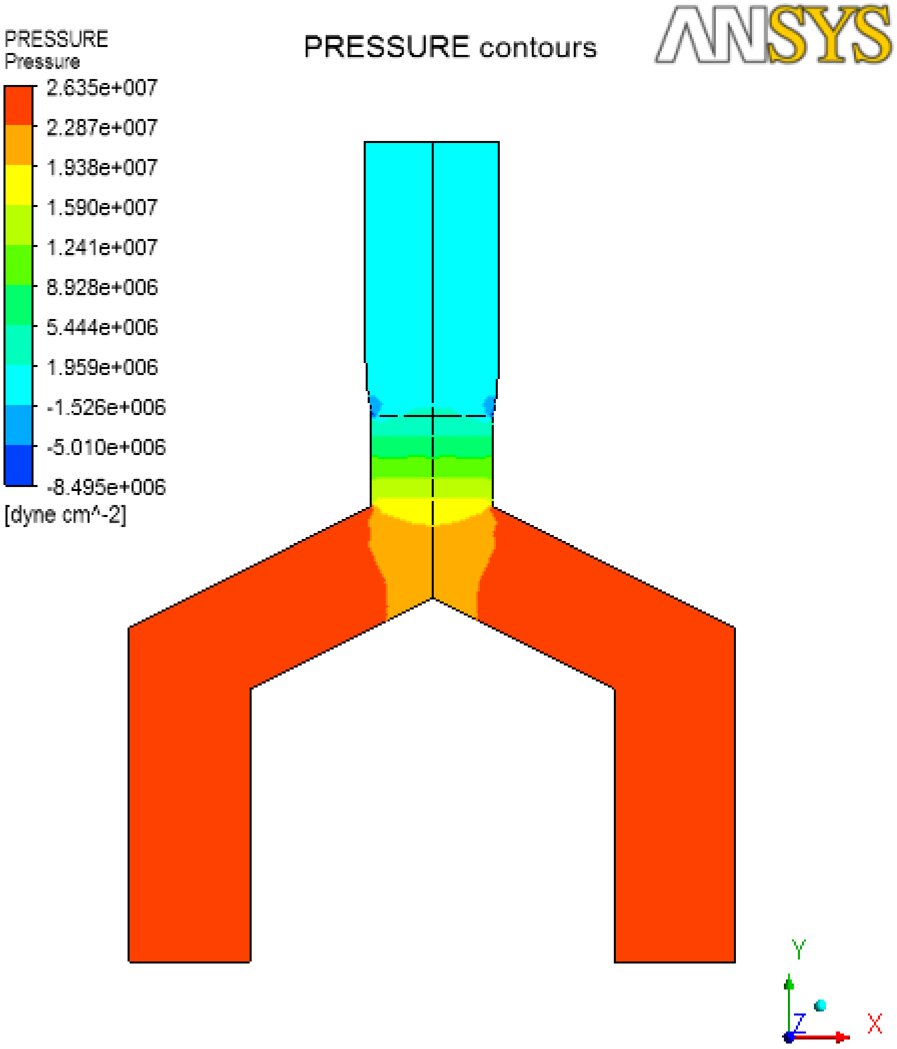
To annotate the display, select the **Text** option from the **Insert** menu.

Insert → **Text**

Click **OK** in the **Insert Text** dialog box that opens.

In the **Definition** tab of the **Details of Text 1** window, enter `PRESSURE contours` for **Text String**. Then click **Apply**.

Figure 4.44 Pressure Contours with Reflection and Annotation



4.14.3. Displaying Velocity Vectors

To display the velocity vectors, perform the following steps.

Select the **Vector** option under the **Insert** menu.

Insert → **Vector**

This displays the **Insert Vector** dialog box.

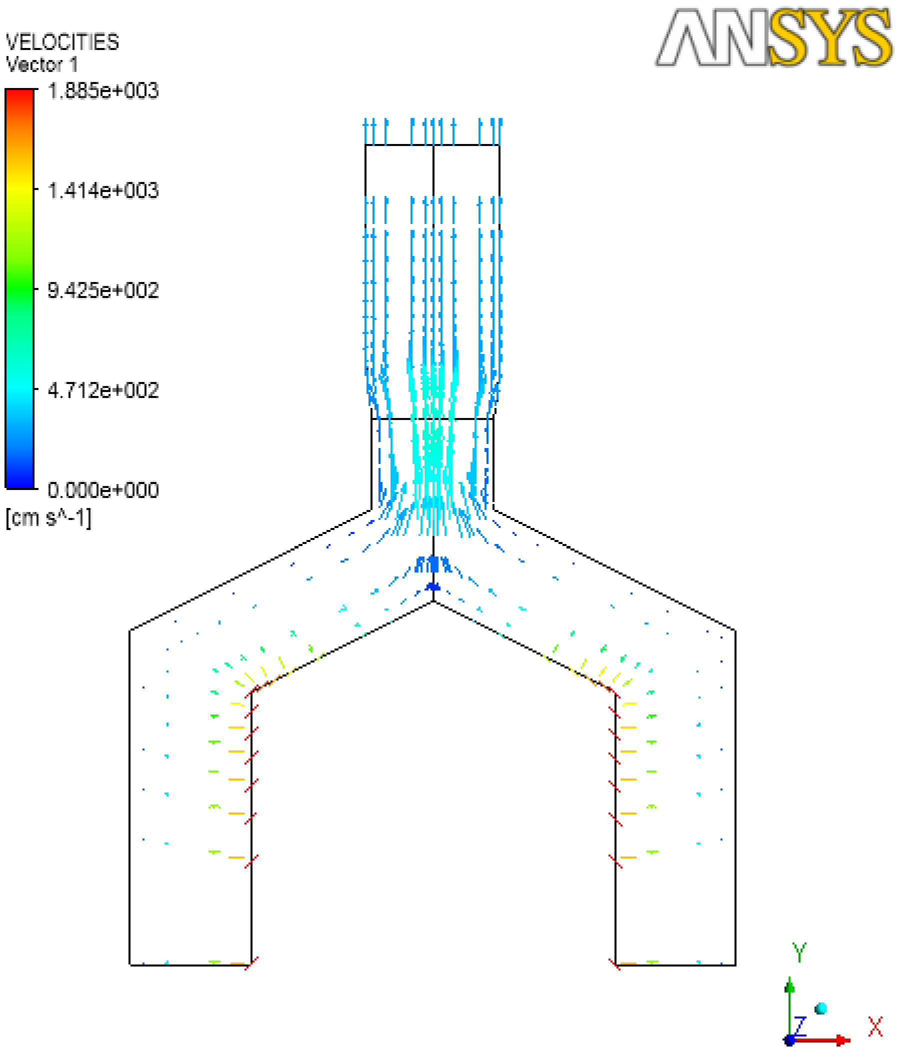
Click **OK** in the **Insert Vector** dialog box.

In the **Geometry** tab of the **Details of Vector 1** window (*Figure 4.38* (p. 92)), click the **Location editor** button (...) to the right of the **Locations** drop-down list. Then select **Subdomain_1_surf** and **Subdomain_2_surf** in the **Location Selector** dialog box (*Figure 4.39* (p. 93)) and click **OK**.

Click **Apply**.

In the **Outline** tab, deselect **Pressure** under **User Locations and Plots** and deselect **Text 1** under **Default Transform**.

Figure 4.45 Velocity Vectors with Reflection



The velocity vectors take all components of the velocity into account. Along the screw tip, the rotational component is important, leading to rather long vectors that are not in the XY plane. After the die exit, a rearrangement of the velocity field takes place: the flow slows down along the axis of symmetry and accelerate on the outside. This makes the particles to go towards the free surface, creating the swelling.

Click the rotate button in the toolbar (*Figure 4.46* (p. 100)) and rotate the view around the Y axis using the left mouse button to see the vectors that point out of the XY plane around the screw tip as shown in *Figure 4.47* (p. 100).

Figure 4.46 The Rotate Button in the Graphics Toolbar

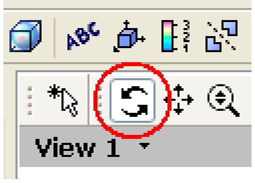
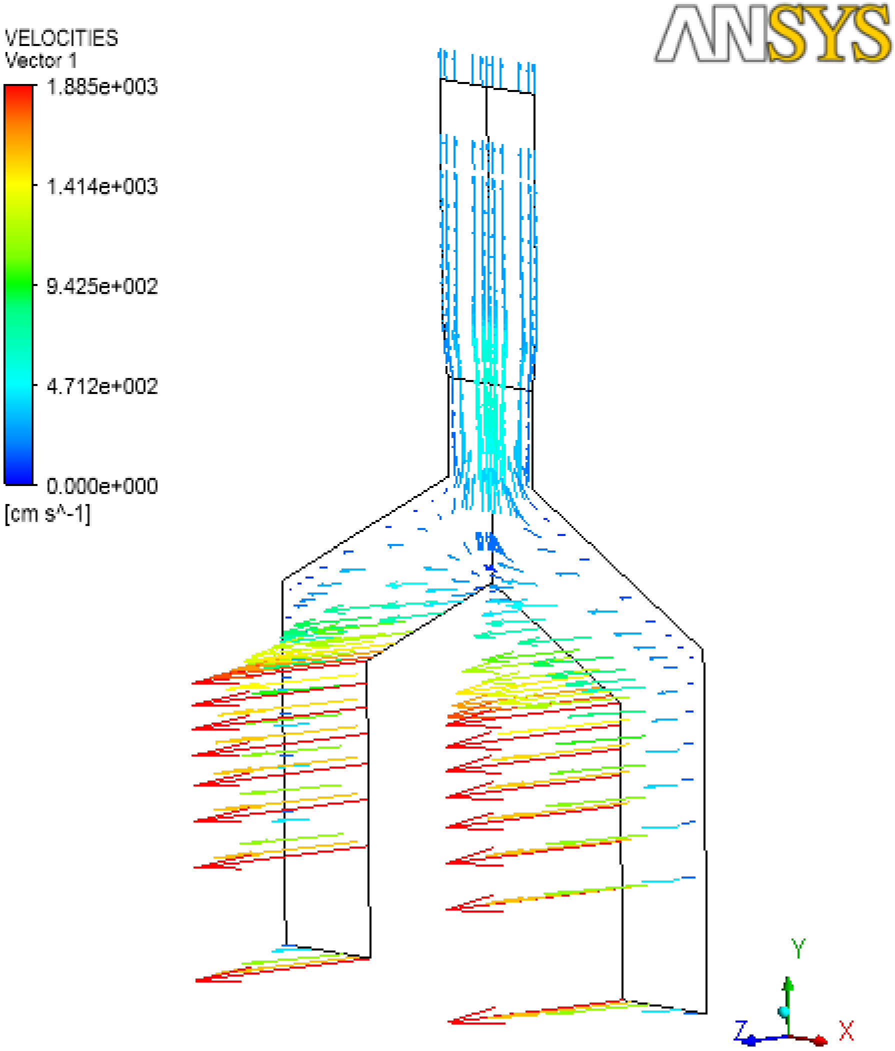


Figure 4.47 Velocity Vectors Pointing Out of the Computation Plane



Important

For more information about manipulating the view in CFD-Post, see the CFD-Post Users Guide.

4.14.4. Exiting from CFD-Post

When you are finished examining the results, you can end the CFD-Post session by selecting **File/Quit**.

File → **Quit**

Click **Quit** in the final dialog box.

Chapter 5: Reading and Writing Files

This chapter describes the different types of files that are used during an ANSYS POLYFLOW simulation. See *Managing ANSYS POLYFLOW Projects Using ANSYS POLYMAN* (p. 29) for information about reading, importing, and exporting files using ANSYS POLYMAN.

Information in this chapter is presented in the following sections:

5.1. Files Written and Read by ANSYS POLYDATA and ANSYS POLYFLOW

During an ANSYS POLYDATA session, you will generally need to read and write several kinds of files. Similarly, ANSYS POLYFLOW itself reads and writes a number of different files, including files for exporting data to various visualization and postprocessing tools. Information about all of these files is provided in this chapter.

Table 5.1: Files Written and Read by ANSYS POLYDATA and ANSYS POLYFLOW (p. 103) lists the files that ANSYS POLYDATA and ANSYS POLYFLOW can read and/or write. You can use this table to get an overview of the files you may be using, to find out which codes write a particular file, and to see where to look for more information on each file. Note that some of the files included in the table are actually converted to the appropriate format when they are read, but no distinction is made here. The section that discusses each file will address this issue, where appropriate.

Table 5.1 Files Written and Read by ANSYS POLYDATA and ANSYS POLYFLOW

File Type	Created By	Used By	Default Name or Suffix	See...
Mesh	POLYMESH, ANSYS POLYDATA, ANSYS POLYFUSE	ANSYS POLYDATA, ANSYS POLYFLOW, ANSYS POLYSTAT, ANSYS POLYFUSE	`.msh`	*Mesh Files Created by POLYMESH or Converted by ANSYS POLYDATA*
FLUENT Mesh	GAMBIT, ANSYS Meshing, TGrid	ANSYS POLYDATA	`.msh`	*Mesh Files Created by Other Programs*
GAMBIT Neutral	GAMBIT	ANSYS POLYDATA	`.neu`	*Mesh Files Created by Other Programs*
ANSYS FIDAP Neutral	ANSYS FIDAP	ANSYS POLYDATA	`.neu`	*Mesh Files Created by Other Programs*

File Type	Created By	Used By	Default Name or Suffix	See...
I-deas Universal	I-deas	ANSYS POLYDATA	`.stb`	*Mesh Files Created by Other Programs*
PATRAN Neutral	PATRAN	ANSYS POLYDATA	`.pat`	*Mesh Files Created by Other Programs*
ICEM Neutral	ANSYS ICEM CFD	ANSYS POLYDATA	`.poly`	*Mesh Files Created by Other Programs*
Mesh	ANSYS Meshing	ANSYS POLYDATA	`.poly`	*Mesh Files Created by Other Programs*
Mesh	ANSYS POLYDATA	ANSYS FLUENT	`.cas`	*Optional Conversion to a Case File*
Data	ANSYS POLYDATA	ANSYS POLYDATA, ANSYS POLYFLOW	`.dat`	*Reading and Writing ANSYS POLYDATA Data Files*, *Reading Mesh, Data, and Results Files into ANSYS POLYFLOW*
Session	ANSYS POLYDATA	ANSYS POLYDATA	`.ses`	*Reading and Writing ANSYS POLYDATA Session Files*
Results	ANSYS POLYFLOW	ANSYS POLYFLOW, ANSYS POLYSTAT	`res`, `.res`	*Reading Mesh, Data, and Results Files into ANSYS POLYFLOW*, *Writing an ANSYS POLYFLOW Results File*
Restart	ANSYS POLYFLOW	ANSYS POLYFLOW	`rst`	*Reading Mesh, Data, and Results Files into ANSYS POLYFLOW*, *Writing an ANSYS POLYFLOW Results File*
Mixing files	ANSYS POLYFLOW	ANSYS POLYSTAT	.mix	ANSYS POLYSTAT User's Guide
Listing	ANSYS POLYFLOW	user	none	*Writing an ANSYS POLYFLOW Listing File*
Mesh and Results	ANSYS POLYFLOW	FLUENT/Post	`.flum`, `.flur`	*Exporting Mesh and Solution Data*
Mesh and Results	ANSYS POLYFLOW	ANSYS POLYPLOT	`.p2m`, `.p2r`; `.p3m`, `.p3r`	*Exporting Mesh and Solution Data*
Mesh and Results	ANSYS POLYFLOW	ANSYS FIDAP	`.neu`	*Exporting Mesh and Solution Data*
Mesh and Results	ANSYS POLYFLOW	PATRAN	`.patm`, `.patr`	*Exporting Mesh and Solution Data*

File Type	Created By	Used By	Default Name or Suffix	See...
Mesh and Results	ANSYS POLYFLOW	I-deas	`.stbm`, `.stbr`	*Exporting Mesh and Solution Data*
Mesh, Geometry, Results	ANSYS POLYFLOW	CFView-PF	`cfv.connec`, `cfv.geom`, `cfv.s`, `cfv.v`	*Exporting Mesh and Solution Data*
Wave	ANSYS POLYFLOW	DataVisualizer	`.wave`	*Exporting Mesh and Solution Data*
IGES	ANSYS POLYFLOW	CAD/CAM programs	`.iges`	*Exporting Mesh and Solution Data*
CSV	ANSYS POLYFLOW, ANSYS POLYSTAT, spreadsheet programs	ANSYS POLYFLOW, spreadsheet programs	`.csv`	*Exporting Mesh and Solution Data*
Results	ANSYS POLYFLOW	FieldView	.uns	*Exporting Mesh and Solution Data*
Mesh, Geometry, Results	ANSYS POLYFLOW	CFD-Post	`_cfx.res`	*Exporting Mesh and Solution Data*
Mesh, Geometry, Results	ANSYS POLYFLOW	EnSight	`.case`	*Exporting Mesh and Solution Data*
Mesh, Geometry, Results	ANSYS POLYFLOW	ANSYS Mechanical APDL	`.cdb`	*Exporting Mesh and Solution Data*
Results	ANSYS POLYFLOW	ANSYS Mechanical	`.xml`, `.txt`	*Exporting Mesh and Solution Data*
Mesh	HYPERMESH	ANSYS POLYDATA	`.hma` or `.hmascii`	*Mesh Files Created by Other Programs*

5.2. Reading Mesh Information into ANSYS POLYDATA

As shown in *Table 5.1: Files Written and Read by ANSYS POLYDATA and ANSYS POLYFLOW* (p. 103), ANSYS POLYDATA can get mesh information from many different sources. This section describes how to read different types of mesh files into ANSYS POLYDATA. For information about combining multiple meshes into a single mesh, see *Combining Meshes with ANSYS POLYFUSE* (p. 151).

5.2.1. Mesh Files Created by POLYMESH or Converted by ANSYS POLYDATA

Mesh files that were created using POLYMESH and mesh files that have been saved from within ANSYS POLYDATA (i.e., mesh files that ANSYS POLYDATA converted from another format as described in *Mesh Files Created by Other Programs* (p. 106)) can be read directly into ANSYS POLYDATA using the **Read a mesh file** menu item in the top-level ANSYS POLYDATA menu.

Read a mesh file

When prompted, specify the name of the file to be read.

5.2.1.1. Optional Conversion to a Case File

ANSYS POLYDATA allows you to reuse an ANSYS POLYFLOW mesh for a simulation with ANSYS FLUENT. Note that there are two limitations to this conversion:

- The conversion is not possible if the mesh is subdivided.
- The PMeshes are skipped.

After a mesh has been read, it is possible to convert it into a case file. When prompted, specify the name of the case file.

5.2.2. Mesh Files Created by Other Programs

Mesh files created using other mesh generators or CAD/CAE programs must be converted to the ANSYS POLYFLOW/ANSYS POLYDATA format before they can be used. To convert one of the external-format mesh files listed in *Table 5.1: Files Written and Read by ANSYS POLYDATA and ANSYS POLYFLOW* (p. 103), do the following:

1. Select the **Convert a mesh file** menu item in the top-level ANSYS POLYDATA menu.

   Convert a mesh file

   You will now be in the **Convert a mesh file** menu.
2. Choose one of the following menu items, depending on which type of file you want to convert:

   - **Convert a FIDAP Neutral File**
   - **Convert an IDEAS Universal File**
   - **Convert a PATRAN Neutral File**
   - **Convert an ICEM Neutral File (*.poly)**
   - **Convert an ANSYS Meshing File (*.poly)**
   - **Convert a GAMBIT Neutral File**
   - **Convert a HYPERMESH File (*.hmascii)**
   - **Convert a FLUENT Mesh File (*.msh)**
3. When prompted, specify the name of the file to be converted.
4. When prompted again, specify the name of the mesh file to which ANSYS POLYDATA should save the file after performing the conversion.
5. Review the conversion information that is displayed. Click **OK** to continue.
6. Read the converted mesh file following the instructions in *Mesh Files Created by POLYMESH or Converted by ANSYS POLYDATA* (p. 105).

5.3. Reading and Writing ANSYS POLYDATA Data Files

The data file contains the information that defines your model. You will generally define the model using ANSYS POLYDATA, and then save a data file. This data file is then used as the input to the ANSYS POLYFLOW calculation. It is also possible to read the data file back into ANSYS POLYDATA so that you can make some modifications to the problem definition.

5.3.1. Writing a Data File

When you have completed your problem definition in ANSYS POLYDATA, you can save it to a data file using the following procedure:

1. Select the **Save and exit** menu item in the top-level ANSYS POLYDATA menu.

   Save and exit

2. When prompted, specify the name of the file where you want to save the data.
3. When ANSYS POLYDATA asks you (in the **Field Management** menu) to confirm the fields that are to be saved when ANSYS POLYFLOW completes the calculation, select **Accept** if the fields listed under **Current fields** are correct. If you want to remove one of the fields so that it will not be saved, select the **Do not view field** menu item for that field. Conversely, if you want to add a field to the list of saved fields, select the **View field** menu item for that field. When you are satisfied with your changes, select **Accept**.
4. Next you will have a chance to specify the name of the results file and other output files for postprocessing, in the **File Management** menu. For the ANSYS POLYFLOW results file and for each output file that has been requested (as described in *Exporting Files for Postprocessing and Additional Simulations* (p. 117)), information about its format and name will be listed. All files are formatted (i.e., ASCII or text files, which you can read and understand yourself using a text editor, and which can be transferred between different types of machines), as indicated by a **yes**. To the right of the **yes** is the name of the file.

   If you are satisfied with the settings, select **Continue**. Otherwise, select the file to be modified and make the desired changes to the file name. ANSYS POLYDATA will save the data file and exit.

Note that ANSYS POLYDATA checks the consistency of the problem setup and will not allow you to save a data file if the problem definition is incomplete. An incomplete problem definition can be saved to a session file, as described in *Reading and Writing ANSYS POLYDATA Session Files* (p. 108), so that you can complete the problem setup in a later ANSYS POLYDATA session.

5.3.2. Reading a Data File into ANSYS POLYDATA

If you have a data file that you would like to modify, you can read it into ANSYS POLYDATA by following the procedure below:

1. Read the mesh file, as described in *Mesh Files Created by POLYMESH or Converted by ANSYS POLYDATA* (p. 105).
2. Select the **Read an old data file** menu item in the top-level ANSYS POLYDATA menu.

   Read an old data file

3. Specify the name of the data file to be read.

You can then continue the problem definition in ANSYS POLYDATA as usual.

5.3.3. Contents of the Data File

A data file consists of two parts.

- The first part contains the data that you specified through ANSYS POLYDATA during the problem definition (including default settings that you did not modify). When you read a data file into ANSYS POLYDATA, only the information in this first part will be read.
- The second part contains information that is used only by ANSYS POLYFLOW.

For example, variables that are introduced implicitly based on your selection of a particular boundary condition in ANSYS POLYDATA will appear in this second part as fields. Also, all numerical parameters appear in the second part of the data file. If you are examining a data file and want to see the different parts, the first part of the data file is contained between the keywords `BEGIN PA3MN` and `ENDOF PA3MN`, and the second part is contained between the keywords `BEGIN OPEN` and `ENDOF OPEN`.

An incomplete problem definition cannot be saved to a data file, but can instead be saved to a session file. This session file contains only the first part of the data file, so it can be read only by ANSYS POLYDATA. See *Reading and Writing ANSYS POLYDATA Session Files* (p. 108) for details.

5.4. Reading and Writing ANSYS POLYDATA Session Files

If you want to stop your current ANSYS POLYDATA session and continue it another time (i.e., if your problem definition is incomplete, so you cannot save a data file), you can save the current session data to a file by clicking the **SAVE** button at the top of the ANSYS POLYDATA menu. The resulting file will contain a subset of the information in a complete data file, as discussed in *Contents of the Data File* (p. 108).

You can restart ANSYS POLYDATA later and read in this file (using the **Read an old data file** menu item, as described in *Reading a Data File into ANSYS POLYDATA* (p. 107)) to continue your session. Note that you cannot read the session file into ANSYS POLYFLOW. The data file (see *Reading and Writing ANSYS POLYDATA Data Files* (p. 107)) is the way to transfer your problem definition to ANSYS POLYFLOW to calculate a solution.

5.5. Reading Mesh, Data, and Results Files into ANSYS POLYFLOW

ANSYS POLYFLOW requires at least the following two sets of information as inputs so that it can compute a solution:

- mesh information (mesh file)
- problem definition (data file)

The mesh file is created by the mesh generator, and the data file is created by ANSYS POLYDATA. It is also possible to give ANSYS POLYFLOW an initial solution from which to start the calculation, by providing a results file as well, but this is not required.

You do not read these files into ANSYS POLYFLOW manually, as you do with ANSYS POLYDATA. Instead, you will provide a file name as a command line input when you start ANSYS POLYFLOW. Furthermore, you do not explicitly specify the mesh (or results) file, only the data file. The data file contains the location of the mesh file (and the results file, if provided), and points ANSYS POLYFLOW to the file(s) for you.

5.5.1. Reading a Data File into ANSYS POLYFLOW

To have ANSYS POLYFLOW read in a data file called `my.dat`, which points to a mesh file called `my.msh`, follow these steps:

1. Be sure that the mesh and data files are both in your working directory.
2. Start ANSYS POLYFLOW by typing:

   ```
   polyflow < my.dat
   ```

ANSYS POLYFLOW will read the mesh and data information from `my.msh` and `my.dat`, and perform the calculation. See *Writing an ANSYS POLYFLOW Listing File* (p. 110). for information about saving the messages that ANSYS POLYFLOW prints during the reading and solution process to a listing file.

5.5.2. Starting an ANSYS POLYFLOW Calculation from an Existing Results File

If you have a results file already, and you want ANSYS POLYFLOW to start the calculation using the results file as an initial solution, you will make this request and specify the name of the results file in ANSYS POLYDATA. Then, when you read the data file into ANSYS POLYFLOW (as described in *Reading a Data File into ANSYS POLYFLOW* (p. 109)), ANSYS POLYFLOW will start the calculation from the initial solution in the results file.

The procedure is as follows:

1. In ANSYS POLYDATA, select the **Numerical parameters** menu item for the task (e.g., in the **F.E.M. Task 1** menu).

   Numerical parameters

2. Select the **Start from an old result file** menu item.

   Start from an old result file

3. When prompted, specify the name of the old results file (e.g., `myold.res`).
4. Return to the top-level ANSYS POLYDATA menu and save the data file (as described in *Writing a Data File* (p. 107)).
5. Follow the instructions in *Reading a Data File into ANSYS POLYFLOW* (p. 109) to read the data file into ANSYS POLYFLOW and start the calculation, after ensuring that the results file is in your working directory with the mesh and data files.

For evolution or time-dependent calculations, you will usually start the calculation from an existing results file and an existing restart file. See *Starting an Evolution or Time-Dependent Calculation from Existing Results and Restart Files* (p. 109) for details.

5.5.3. Starting an Evolution or Time-Dependent Calculation from Existing Results and Restart Files

As described in *Starting an ANSYS POLYFLOW Calculation from an Existing Results File* (p. 109), you can have ANSYS POLYFLOW start the calculation using an existing results file as an initial solution. For an evolution or time-dependent calculation, you will also need the value of S or t for the given results. This information is provided in an additional file, called the restart file, which is automatically saved during an evolution or time-dependent calculation. See *Initial Conditions* (p. 526) and *Initial Conditions* (p. 547) for more information about restart files.

You have to provide the names of the results and restart files in ANSYS POLYDATA. The procedure is as follows:

1. In ANSYS POLYDATA, select the **Numerical parameters** menu item for the task (e.g., in the **F.E.M. Task 1** menu).

   Numerical parameters

2. Select the **Start from old result and restart files** menu item.

   Start from old result and restart files

3. When prompted, specify the name of the old results file (e.g., `myold.res`).
4. When prompted, specify the name of the restart file (e.g., `rst`).
5. Return to the top-level ANSYS POLYDATA menu and save the data file (as described in *Writing a Data File* (p. 107)).
6. Follow the instructions in *Reading a Data File into ANSYS POLYFLOW* (p. 109) to read the data file into ANSYS POLYFLOW and start the calculation, after ensuring that the results file and restart file are in your working directory with the mesh and data files.

5.6. Writing an ANSYS POLYFLOW Results File

When ANSYS POLYFLOW completes the calculation, it will automatically write the node coordinates and solution data to a results file. ANSYS POLYFLOW gets the name of this results file, as well as its format, from the ANSYS POLYDATA data file. See *Writing a Data File* (p. 107) for details. To save solution data for postprocessing, see *Exporting Files for Postprocessing and Additional Simulations* (p. 117).

Note

The ANSYS POLYFLOW results file can be read only by ANSYS POLYFLOW itself.

5.7. Writing an ANSYS POLYFLOW Listing File

During an ANSYS POLYFLOW session, ANSYS POLYFLOW prints a number of messages regarding its progress. By default, these messages are printed to the screen in the window where you started ANSYS POLYFLOW.

It is convenient to have ANSYS POLYFLOW write these messages to a file (called a listing file), so that you can view them more easily and store them for further review of the solution convergence history.

5.7.1. Saving ANSYS POLYFLOW Messages to a Listing File

The procedure for writing a listing file is as follows:

1. Be sure the mesh and data files (and the results file, if you are starting from an old one) are in your working directory.
2. Start ANSYS POLYFLOW by typing

```
polyflow < my.dat > my.lst
```

Note that a listing file will be created automatically if you start ANSYS POLYFLOW from ANSYS POLYMAN, as described in *Starting the Programs* (p. 40).

ANSYS POLYFLOW will read the specified data file (and the mesh and results files it points to) and perform the calculation, writing all messages about its progress to the listing file `my.lst`. When the calculation is complete, you can view the listing file with your text editor, so that you can review what ANSYS POLYFLOW has done. For example, you can check the listing file to see if the solution has converged. See *Contents of a Sample Listing File* (p. 111) for a description of the listing file's contents.

5.7.2. Controlling the Amount of Information in the Listing File

By default, ANSYS POLYFLOW will save a subset of the information listed in *Contents of a Sample Listing File* (p. 111) to the listing file. If you prefer, you can increase or decrease the amount of information:

1. In the top-level ANSYS POLYDATA menu, select **Outputs**.

   Outputs

2. Specify the amount of information in the listing file by choosing one of the following menu items:

   - **Listing: none** disables all listing messages except errors and warnings.
   - **Listing: min** prints less extensive information about meshes, number of elements, and problem parameters.
   - **Listing: max** prints extensive information, as shown in the example in *Contents of a Sample Listing File* (p. 111).

3. When you are satisfied with your selection, click **Upper level menu** to return to the top-level menu.

5.7.3. Contents of a Sample Listing File

A sample listing file is shown below. The first part of the file contains information about the version of ANSYS POLYFLOW that is being used.

```
  Polyflow 32-bit build 'Tue 09/15/2009 4:16:31.38'
 Startup file is p3rc_polyflow

  Polyflow running on JDOEPC with 1 processor
 Arguments of Polyflow : -ES full -s p3rc_polyflow -MGR
 IP solver available

PPPPPP   OOOOO  LL  YY      YY     FFFFFFF LL      OOOOO  WW     WW
PP   PP OO   OO LL  YY      YY    FF      LL     OO   OO WW     WW
PP   PP OO   OO LL   YY    YY    FF      LL     OO   OO WW     WW
PPPPPP  OO   OO LL    YY  YY    FFFFF   LL     OO   OO WW     WW
PP      OO   OO LL     YYYY    FF      LL     OO   OO WW     WW
PP      OO   OO LL      YY    FF      LL     OO   OO WW  WW WW
PP      OO   OO LL      YY   FF      LL     OO   OO WWW   WWW
PP       OOOOO  LLLLLLL YY  FF      LLLLLLL OOOOO  WW     WW

 *****************************************
 *                                       *
 * POLYFLOW s.a.                         *
 * Avenue Pasteur, 4                     *
 * B-1300 WAVRE                          *
 * BELGIUM                               *
 *                                       *
```

```
  * TEL : 32-(0)10-452861                *
  * FAX : 32-(0)10-453009                *
  *                                      *
  * URL : www.ansys.com                  *
  *  www.ansys.com/products              *
  *                                      *
  * Customer portal :                    *
  *  www.ansys.com/customerportal        *
  *                                      *
  ****************************************

  *************************
  * Version 14. 0. 0      *
  * .alpha                *
  *                       *
  ************************
 Third Party Attributions:

 This product contains the following licensed software
 which requires reproduction of the following notices:

 FLEXlm and FLEXnet are trademarks of Macrovision Corporation.

 The METIS Software Package is copyrighted as follows:
 Copyright 1998, Regents of the University of Minnesota

 The Intel(R) Math Kernel Library is copyrighted as follows:
 Copyright(C) 2000-2007,Intel Corporation. All rights reserved.

 GCC, the GNU Compiler Collection is copyrighted as follows:
 Copyright (C) Free Software Foundation, Inc.,
 51 Franklin St, Fifth Floor, Boston, MA 02110, USA.

 The sed (cygwin) software is copyrighted as follows:
 Copyright 1996, 1997, 1998, 1999, 2000, 2001, 2002 Red Hat, Inc.
```

Next, there is information about the topological operations in the TOPO section. For each subdomain, this section reports the number of elements, faces, segments, and nodes, as well as the spatial dimension.

For each boundary set and each intersection between subdomains, it reports the number of faces, segments, and nodes, and the dimension.

```
  *************************
  *                       *
  *          TOPO         *
  *                       *
  *************************

   root mesh
            Space Dim    .:     2
            Num. of faces :   200
            Num. of segm. :   430
            Num. of nodes :   231

  S1.
            Space Dim.    :     2
            Num. of faces :   100
            Num. of segm. :   220
            Num. of nodes :   121

  S2.
            Space Dim.    :     2
            Num. of faces :   100
            Num. of segm. :   220
            Num. of nodes :   121

 (S1+S2).
            Space Dim.    :     2
            Num. of faces :   200
            Num. of segm. :   430
            Num. of nodes :   231
 (S1*S2).
            Space Dim.    :     1
            Num. of segm. :    10
            Num. of nodes :    11

 (S1*B1).
            Space Dim.    :     1
            Num. of segm. :    10
            Num. of nodes :    11

 (S1*B4).
            Space Dim.    :     1
            Num. of segm. :    20
            Num. of nodes :    21

 (S2*S1).
            Space Dim.    :     1
            Num. of segm. :    10
            Num. of nodes :    11

 (S2*B1).
            Space Dim.    :     1
            Num. of segm. :    10
            Num. of nodes :    11

 (S2*B2).
            Space Dim.    :     1
            Num. of segm. :    10
            Num. of nodes :    11

 (S2*B3).
            Space Dim.    :     1
            Num. of segm. :    10
            Num. of nodes :    11
```

The FIELDS section has information about the fields used by ANSYS POLYFLOW. Some fields (e.g., coordinates or velocity) are directly defined by sub-tasks, while others (e.g., normal/tangential field) are required for boundary conditions or other constraints.

P1;C0 indicates a linear continuous interpolation, P2;C0 a quadratic continuous interpolation, etc.

```
  **************************
  *                        *
  *          FIELDS        *
  *                        *
  **************************
  COORDINATES
             Abreviated as   :        XY
             Support         : root mesh
             Interp. Type    :     P1;C0
             Tensor Type     :         1
             Num. of Comp.   :         2
             Num. of Var.    :       462

 TEMPERATURE
             Abreviated as   :         T
             Support         :  (S1+S2).
             Interp. Type    :     P2;C0
             Tensor Type     :         0
             Num. of Comp.   :         1
             Num. of Var.    :       861
```

The PROBLEMS section prints material properties for all problems currently defined, whether they are referenced by the solver or not.

```
  **************************
  *                        *
  *        PROBLEMS        *
  *                        *
```

```
**************************

Navier-Stokes 2D
           Support          :          S1.
           Coordinates      :  COORDINATES
           Input Fields     :            -
           Output Fields    :  TEMPERATURE

Navier-Stokes 2D and 2D 1/2
thermal problem
plane geometry

specific mass : ro  = 0.00000E+00

coefficients of conductivity law :
 condu = a + b * (t-t0) + c * (t-t0)**2 + d * (t-t0)**3
  a = 1.00000E+01
  b = 0.00000E+00
  c = 0.00000E+00
  d = 0.00000E+00
  t0 = 0.00000E+00

 coefficients of heat capacity law :
 Cp = a + b * (t-t0) + c * (t-t0)**2 + d * (t-t0)**3
  a = 0.00000E+00
  b = 0.00000E+00
  c = 0.00000E+00
  d = 0.00000E+00
  t0 = 0.00000E+00

compressibility neglected

Neumann for 2D
         Support          :     (S1*S2).
         Coordinates      :  COORDINATES
         Input Fields     :            -
         Output Fields    :  TEMPERATURE

natural boundary conditions 2D and 2D 1/2
plane geometry

Navier-Stokes 2D
         Support          :          S2.
         Coordinates      :  COORDINATES
         Input Fields     :            -
         Output Fields    :  TEMPERATURE

Navier-Stokes 2D and 2D 1/2
thermal problem
plane geometry

specific mass : ro  = 0.00000E+00
coefficients of conductivity law :
 condu = a + b * (t-t0) + c * (t-t0)**2 + d * (t-t0)**3
  a = 3.50000E+01
  b = 0.00000E+00
  c = 0.00000E+00
  d = 0.00000E+00
  t0 = 0.00000E+00

coefficients of heat capacity law :
 Cp = a + b * (t-t0) + c * (t-t0)**2 + d * (t-t0)**3
  a = 0.00000E+00
  b = 0.00000E+00
  c = 0.00000E+00
  d = 0.00000E+00
  t0 = 0.00000E+00

compressibility neglected

Neumann for 2D
           Support       :    (S2*S1).
           Coordinates   :  COORDINATES
           Input Fields  :            -
           Output Fields :  TEMPERATURE

natural boundary conditions 2D and 2D 1/2
plane geometry
```

Next, the `Boundary Conditions` are listed, including information about the mesh boundary on which the conditions apply, the problems for which they apply, and the type of constraint.

```
**************************
*                        *
*  Boundary Conditions   *
*                        *
**************************
 Tp imposed
           Field         :    TEMPERATURE
           Support       :       (S1*B4).
           Act. on Probs :
                         Navier-Stokes 2D
                           Neumann for 2D
                         Navier-Stokes 2D
                           Neumann for 2D
           For comp. 1   :  1.5000000E+02

 Tp imposed
           Field         :    TEMPERATURE
           Support       :       (S2*B2).
           Act. on Probs :
                         Navier-Stokes 2D
                           Neumann for 2D
                         Navier-Stokes 2D
                           Neumann for 2D
           For comp. 1   :  1.0000000E+02
```

Finally, the `SOLVER` section is printed. This section contains a list of the problems being solved, as well as information about the solution methods, mesh decomposition and optimization, memory usage, iteration count, and convergence.

Computation time and elapsed time are reported at the end. You should check this section to confirm that the solver has indeed converged before you begin to postprocess the results. Note that the contents and layout of this section may vary according to the specific linear solver used in the simulation.

```
*************************
*                       *
*        SOLVER         *
*                       *
*************************

F.E.M. Task 1

           Type of Evolu.:      Implicit
           Infl. on Evol.:      Influent
           Explicit part :       One pass
           Problem list  :
                  Navier-Stokes 2D
                    Neumann for 2D
                  Navier-Stokes 2D
                    Neumann for 2D

           Nitmax      :             10
           Static      :             T
           Conver. Crit. :  0.1000000E-03
           Diverg. Crit. :  0.1000000E+04
           Print Iter.   :              T
************************************
*                                  *
*   Automatic mesh decomposition   *
*         and optimization         *
```

```
 *                                  *
 ************************************

 List of active sub-domains :
   S1 S2

 Nb. of active elements :   200
 Nb. of sub-parts ..... :     1


 TOTAL COST EVALUATION

 node --------- cost ----- nvar ----- mvac ---- mvacf

  1           43000.        430.       10.         0.

  Estimation of total cost: 43000. * 1E+00 integer operations

  <!> End of automatic mesh decomposition and optimization
     Return to current SOLVER ...

 Frontal method preprocessor
 BLAS3 in use, Blocs =  20
 Total number of active variables  :   569
 Total number of static variables  :   230


 Solver memory
  Memory requirement before elimination       : 1.6 Mbytes
  Memory for factorized matrix (single prec.) : 0.1 Mbytes
  Memory for Schur Complements (double prec.) : 0.0 Mbytes
  Memory for work vectors                     : 0.0 Mbytes
  Memory for active matrices (RAM)            : 0.0 Mbytes

Global Memory Information :
  Swap space         :   1.7 Mbytes
  Disk space         :   0.0 Mbytes
  RAM                :   0.0 Mbytes
 Elimination cost    :   2.2 MFlop

 Iteration 1

 Frontal method information :
  Minimal pivot       :  0.1100337E+02
  Maximal pivot       :  0.6003032E+03
  log10 / sign (det.) :  0.1453584E+04 / 1
  Maximal rhs         :  -0.1517886E+05
  Relative var. of field TEMPERATURE 0.1333333E+01

 Iteration 2

 Frontal method information :
  Minimal pivot       :  0.1100337E+02
  Maximal pivot       :  0.6003032E+03
  log10 / sign (det.) :  0.1453584E+04 / 1
  Maximal rhs         :  0.3335229E-02
  Relative var. of field TEMPERATURE 0.9160221E-07
 Convergence assumed : Rel. var. LT 0.1000000E-03

 Memory information :
  Total Memory requirement                :  3. Mbytes
  Memory requirement for buffering        :  1. Mbytes
  Memory requirement for active matrices  :  1. Mbytes

 Cost information :
  Maximum elimination cost: 3. * 1E+06 floating operations

 Time information :
  CPU time     :   5.4 sec.
  Elapsed time :  10.5 sec.
```

```
Stop. Normal end of Polyflow
Polyflow running on irix53 with 1 processor
Arguments of Polyflow :
***********************************
*                                 *
*    Summary of the simulation    *
*                                 *
***********************************

The computation succeeded.
```

5.8. Exporting Files for Postprocessing and Additional Simulations

The results of your ANSYS POLYFLOW simulation can be used by a variety of other applications, in order to postprocess the results or perform additional simulations. Each application will require that you save the mesh and solution data in an appropriate format. You will specify the desired output format(s) in ANSYS POLYDATA, and this information will be passed to ANSYS POLYFLOW through the data file.

ANSYS POLYFLOW will then save the requested files when it completes the calculation (or at specified intervals during a time-dependent or evolution calculation, as described in *Output for Time-Dependent and Evolution Calculations* (p. 126)). The procedure for saving mesh and solution data for external post-processing or simulation is described in *Exporting Mesh and Solution Data* (p. 117).

It is also possible to save solution data at a given point. See *Saving Data at a Specified Point* (p. 124) for details.

Note

Upon reading ANSYS POLYFLOW output files into another application, warning messages about element quality are sometimes issued by the application, due to the fact that criteria for evaluating the element quality are software-dependent. Whether a corrective action is required is left to your discretion.

5.8.1. Exporting Mesh and Solution Data

The procedure for selecting the output formats you want to save is as follows:

1. In the top-level ANSYS POLYDATA menu, select **Outputs**.

   Outputs

2. Choose one or more of the items below to enable output for your postprocessor(s) / application of choice:

   - **Enable Polyplot output**
   - **Enable 3DCross output**
   - **Enable Patran output**
   - **Enable Ideas output**
   - **Enable DataVisualizer output**
   - **Enable CSV (Excel) output**

- **Enable CFView-PF output**
- **Enable Polyflow output**
- **Enable Iges file output**
- **Enable Fluent-Post (flum, flur) output**
- **Enable Fidap output**
- **Enable FieldView UNS output**
- **Enable CFD-Post output**
- **Enable EnSight output**
- **Enable Ansys Mechanical APDL output**
- **Enable Ansys Mapper output**

The **Polyflow** output refers to a results file that can be read by ANSYS POLYFLOW itself and is saved at each step of an evolution or time-dependent calculation. The **CFD-Post** format refers to CFD-Post, the standard postprocessor for ANSYS POLYFLOW. The **Fluent-Post (flum,flur)** format refers to FLUENT/Post, the **Fidap** format refers to **FIPOST**, and the **Ansys Mapper** format refers to files that are compatible with ANSYS Mechanical (note that only thickness and temperature data from POLYFLOW can be written in this format). See the descriptions that follow for information about the other output formats.

By default, the CFD-Post output is enabled, as listed under **Current output(s)** at the top of the menu. You can specify the system of units that corresponds with the data you entered in ANSYS POLYDATA, so that CFD-Post (as well as ANSYS Mechanical, if the **Ansys Mapper** format is enabled) provides the correct units when it displays your results. See step 4. for further details.

Important

Note that the information you provide for CFD-Post and/or ANSYS Mechanical regarding the system of units will not affect any of the data entered in your ANSYS POLYDATA session.

A final results file is always saved in ANSYS POLYFLOW format at the end of a calculation, so you need not enable this option to ensure that your final results are saved. You only need to enable this option if you want ANSYS POLYFLOW to save the results at each evolution or time step (or as often as specified by the triggering in *Output for Time-Dependent and Evolution Calculations* (p. 126)).

3. If you want to turn off any of the current outputs, select the corresponding **Disable** menu item. (Each **Enable** menu item will become a **Disable** item after you select it.)
4. If **CFD-Post** and/or **Ansys Mapper** output is enabled, you must approve the system of units that will be passed to CFD-Post and/or ANSYS Mechanical with the results. This approval is accomplished using the **Change System of Units for CFD-Post or Ansys Mapper** menu. Perform the following actions, starting in the **Outputs** menu:

a. Open the **Change System of Units for CFD-Post or Ansys Mapper** menu by clicking the **Set units for CFD-Post or Ansys Mapper** menu item.

   Set units for CFD-Post or Ansys Mapper

   Note that the **Change System of Units for CFD-Post or Ansys Mapper** menu will open automatically when you click **Enable CFD-Post output** or **Enable Ansys Mapper output**.

b. Review the list of units currently selected for each quantity, as displayed in the **Change System of Units for CFD-Post or Ansys Mapper** menu. If you want to revise any of the units listed under **Current**, click the **Modify system of Units** menu item.

   Modify system of Units

   The **Current System of Units** menu will open, where you can make your selections:

   - You can specify the units for all of the quantities at once by clicking the **Set to <system>** menu item, where **<system>** is the appropriate set of units for your inputs (e.g., **metric_MKSA+Kelvin** corresponds to meter, kilogram, second, Ampere and Kelvin). The units displayed at the top of the menu will be immediately updated.
   - You can specify the unit for an individual quantity by clicking the **Modify the <quantity> unit** menu item, where **<quantity>** is a quantity with inappropriate units (e.g., **length**). The menu will then display your options for this quantity; click the **Select <unit>** menu item, where **<unit>** is the appropriate unit for your inputs (e.g., **millimeter**). Then click **Upper level menu** to return to the previous menu display and to update the units listed at the top of the menu.

c. Click **Upper level menu** repeatedly to return to the **Outputs** menu.

Note that if you fail to perform step 4.(a), the **Change System of Units for CFD-Post or Ansys Mapper** menu will open automatically before you are allowed to navigate away from the **Outputs** menu or save the data file; this is to ensure that you approve of the system of units passed to CFD-Post and/or ANSYS Mechanical.

5. If CFD-Post output is enabled, you can specify whether the names of the boundaries and subdomains in the output are those defined via Named Selections in the ANSYS Meshing application (the default) or the short names generated by ANSYS POLYDATA. (For more information about Named Selections, see Named Selections and PMeshes in the separate POLYFLOW in Workbench User's Guide.) The currently selected convention is displayed near the top of the **Outputs** menu. To change the naming convention, click either of the following at the bottom of the menu, as appropriate:

   Switch to mode: using short names in CFD-Post

   or

   Switch to mode: using named selections in CFD-Post

6. When you are satisfied with the **Current output(s)**, click **Upper level menu** to return to the top-level menu.

When you save the data file, as described in *Writing a Data File* (p. 107), ANSYS POLYDATA will give you the opportunity to specify names for all of the output files that you have requested in the procedure above. See *Table 5.1: Files Written and Read by ANSYS POLYDATA and ANSYS POLYFLOW* (p. 103) for file types and default naming conventions corresponding to the different postprocessors.

5.8.1.1. ANSYS POLYPLOT, V3DMSH, and 3DCROSS Files

For 2D problems, the output for ANSYS POLYPLOT is contained in files named (by default) `p2m` and `p2r`. For multi-material problems, where different fields are computed in different domains, several files will be generated.

For example, consider a problem in which the energy and momentum equations are solved in subdomain 1, and only the energy equation is solved in subdomain 2. In this case, the mesh information for subdomain 1 will be written to `p2m`, the velocity, stream function, and temperature in subdomain 1 will be written to `p2r`, the entire mesh will be written to `p2m_2`, and the temperature field for both subdomains will be written to `p2r_2`.

For 3D problems, the default names of the output files will be `p3m` and `p3r`. The comment regarding output files for multi-material problems applies to these files as well. The `p3m` file can be examined using V3DMSH, and the `p3m` and `p3r` files can be used in 3DCROSS to generate 2D slices.

3DCROSS allows you to define the equation of a plane, and then saves ANSYS POLYPLOT files containing the intersection of this plane with the mesh and results.

5.8.1.2. PATRAN Files

The output files for PATRAN are a neutral file containing the connectivity tables and the nodal coordinates (called `patm`, by default), and a results file containing the nodal values of the fields (called `patr`, by default). The header of the results file contains a short description of each variable.

Since PATRAN does not understand all of the interpolation types available in ANSYS POLYFLOW, results will be converted according to *Table 5.2: Conversion of Interpolation Types for PATRAN Output* (p. 120).

Table 5.2 Conversion of Interpolation Types for PATRAN Output

	2D	**2D**	**3D**	**3D**
Interpolation in ANSYS POLYFLOW	P1_C0 P0_C-1	other	P1_C0 P0_C-1	other
Converted to	TRI3 QUAD4 (low order)	TRI6 QUAD9 (high order)	TET4 WED6 HEX8 (low order)	TET10 WED15 HEX20 (high order)

For mixed interpolation types (e.g., velocity-pressure), high-order elements will be generated for all fields whenever one of the fields in the PATRAN results file is interpolated with high-order elements. The file convention for multi-material problems described above for ANSYS POLYPLOT files applies to PATRAN files as well.

5.8.1.3. I-deas Files

The output files for I-deas are two universal files: one containing the connectivity tables and the nodal coordinates (called `stbm`, by default), and one containing the nodal values of the fields (called `stbr`)

Table 5.3 Conversion of Interpolation Types for I-deas Output

	2D	**2D**	**3D**	**3D**
Interpolation in ANSYS POLYFLOW	P1_C0 P0_C-1	other	P1_C0 P0_C-1	other
Converted to	linear (low order)	parabolic (high order)	linear (low order)	parabolic (high order)

Since I-deas does not understand all of the interpolation types available in ANSYS POLYFLOW, results will be converted according to *Table 5.3: Conversion of Interpolation Types for I-deas Output*(p. 121).

For mixed interpolation types (e.g., velocity-pressure), high-order elements will be generated for all fields whenever one of the fields in the I-deas results file is interpolated with high-order elements. Note that 2D 9-node quadrilaterals are converted to 8-node serendipity elements. 3D quadratic bricks are converted to 20-node elements, because of I-deas limitations.

The file convention for multi-material problems described above for ANSYS POLYPLOT files applies to I-deas files as well.

5.8.1.4. DataVisualizer Files

The output file for DataVisualizer is a wave file containing the solution fields. Note that the file convention for multi-material problems described above for ANSYS POLYPLOT files applies to DataVisualizer files as well.

5.8.1.5. CFView-PF Files

The output files for CFView-PF are, by default, `cfv.cfv`, `cfv.connec`, `cfv.geom`, `cfv.s.#`, and `cfv.v.#`, where # is an index for the number of the scalar and vector solution fields.

Note

The velocity field includes additional scalar fields containing the components of the velocity vector and the magnitude of the velocity.

The `cfv.cfv` file contains general information about the results, and pointers to the other files where additional information can be found. The `cfv.connec` and `cfv.geom`files contain information on connectivity between nodes and the geometrical positions of nodes. The `cfv.s.#` and `cfv.v.#` files contain the values of the scalar and vector fields at each node.

All of the files are saved in ASCII format, so they can be transferred between different types of machines. Before you read the files into CFView-PF, you will need to convert them to the appropriate binary format. See the CFView-PF documentation for details.

For evolution and time-dependent problems, the output is organized in a hierarchy of directories, starting from the current one. The connectivity file does not change, so it is written to the top-level directory. The remaining files, including the geometry file and the scalar and vector field files, are written to subdirectories named for each evolution or time step at which outputs are saved.

The file convention for multi-material problems described above for ANSYS POLYPLOT files applies to CFView-PF files as well.

5.8.1.6. IGES Files

The IGES output file contains lines that describe the geometry. If you save IGES files during a time-dependent or evolution calculation, they will describe the modified (updated) geometry calculated as part of the solution.

This is particularly useful for inverse extrusion problems, where the geometry of the die is calculated by ANSYS POLYFLOW. This IGES file contains the precise die geometry (adaptive section, constant section, inlet section, etc.), which can be postprocessed in most CAD/CAM programs, allowing for a direct interface to die cutting machines.

5.8.1.7. CSV Files

A CSV (comma separated variables) file is a common format for tabulated data that can be read into spreadsheet programs such as Excel. The CSV file that ANSYS POLYFLOW can read or write contains a list of data points and a list of values. Fields are recognized by their (case-sensitive) names.

When you save ANSYS POLYFLOW results in a CSV format, a data entry is written for each nodal location at which the quantity is defined. When ANSYS POLYFLOW reads a CSV file, it will interpolate the data (if they do not correspond exactly to the nodes in the mesh) between the listed values.

CSV files can be used for interpolating results between different meshes and setting velocity boundary conditions. See *Results Interpolation Onto Another Mesh* (p. 150) and *Velocity Profile from a CSV File* (p. 185) for details. You can also initialize solution variables with a CSV file, as described in *Using the CSV File to Initialize Solution Variables* (p. 151).

5.8.1.8. FieldView Files

The output file for FieldView contains the geometry and the computed fields. The suffix of the file is `.uns`. Actually, ANSYS POLYFLOW creates a `fv.uns` file for steady-state simulations and a series of `fv#.uns` files, where # is the step index for evolution or time-dependent problems. This numbering and syntax allows FieldView to load all the files of a transient calculation in one single step.

The.uns file is in a portable binary format i.e, a `.uns` file created under Linux can be read under Windows and vice-versa. FieldView supports linear interpolation only. All the fields of an ANSYS POLYFLOW simulation are thus converted into linear fields i.e, defined at the vertices of the FEM mesh.

5.8.1.9. CFD-Post Files

The output file for CFD-Post (identifiable by the `cfx.res` suffix) contains the geometry and the computed fields. By default, ANSYS POLYFLOW creates a `cfx.res` file for steady-state simulations and a series of `#_full.trn` files in a cfx directory, where # is the step index for evolution or time-dependent problems. This numbering and syntax allows CFD-Post to load all the files of a transient calculation in one single step.

The `cfx.res` file is in a portable binary format (i.e., a `cfx.res` file created under Linux can be read under Windows and vice-versa). CFD-Post supports linear interpolation only. All the fields of an ANSYS POLYFLOW simulation are thus converted into linear fields (i.e., defined at the vertices of the FEM mesh).

Note that names for CFD-Post files should be carefully selected. For enabling a proper file identification and for avoiding possible confusion, a short filename should be `cfx.res`, while a long file name should preferably and at least contain the string `_cfx.res`.

You must specify the units used by ANSYS POLYDATA and ANSYS POLYFLOW. CFD-Post will convert the results in the MKSA system of units when reading the results files. In the **Output** menu, select **Set units for CFD-Post**. You can also define the units by selecting **Enable CFD-Post output**, which opens a new menu named **Change System of Units**.

5.8.1.10. EnSight Files

The output files for the EnSight graphic postprocessor are based on the EnSight gold format. The typical output consists of a case file (`.case`) and several files for geometry and fields. The case file is the entry file to EnSight; it is an ASCII file that contains the key information about the simulation. For a steady simulation, the case file contains the file names of the files for geometry and fields.

For evolution and transient simulations, the content of a case file is richer. In particular, it contains generic names for the various files (geometry and fields), where wildcards or * characters refer to the sequence of calculation steps. The number of wildcards depends on the maximum number of transient/evolution steps that you define in the ANSYS POLYDATA session. The case file also contains the list of steps that produce the output files, as well as the sequence of corresponding values of time or evolution parameters. Note that a large number of files can be created, for individual fields and for individual transient/evolution steps.

Upon restarting a calculation, ANSYS POLYFLOW checks whether the former case file can be reused. If an incompatibility is detected in the early stage, the calculation will stop. The most typical incompatibility is caused when a risk exists of making former output files simply inaccessible for subsequent analysis. This typically occurs when the number of wildcards is modified. In most cases, the causes of incompatibility can be circumvented by selecting a new name for the EnSight output file when restarting a calculation.

All files created on Windows or Linux are portable. This concerns not only the case file, which is ASCII, but also the geometry and field files, which are in binary format. Binary format is used for reasons of reading efficiency. This argument is especially true when a large number of files is involved. In the production of EnSight output files, only the linear interpolation is available. All fields of an ANSYS POLYFLOW calculation are converted into linear fields, i.e., defined at the vertices of the mesh file. It is interesting to note that PMeshes are visible.

5.8.1.11. ANSYS Mechanical APDL Files

The ANSYS Mechanical APDL output file can be selected when you are interested in a subsequent structural analysis. The output file for ANSYS Mechanical APDL contains the geometry as well as temperature and thickness fields when available. The suffix of the file is `cdb`. You can import the `.cdb` file containing thickness and/or temperature as fields only in ANSYS Mechanical APDL **Classic**. By default, ANSYS POLYFLOW creates a `.cdb` file for steady-state simulations and a series of `cdb.#` files, where # is the step index for evolution or time-dependent problems. The `.cdb` file is in a portable ASCII format.

Next to the geometry, default values are also created for material properties. For nonisothermal cases, the temperature field is provided. For shell models, only information from the fluid region is provided, whereas the mold information is discarded (for more information about shell models, refer to *Blow Molding and Thermoforming* (p. 379)). For fluid domains with multiple layers, the total thickness distribution is provided.

For 2D applications, including shells, triangles and quadrilaterals are supported. In 3D, tetrahedrons, pyramids, prisms and brick elements are supported. However, if a pyramid element is encountered, it will be advised to take appropriate actions in ANSYS Mechanical APDL and selecting the quadratic interpolation for the entire geometry. When this occurs, a message, similar to the following, will appear in the ANSYS POLYFLOW listing file:

```
During the production of a results file for ANSYS,
at least one linear pyramid has been detected.
ANSYS will require a conversion towards element 186
with the commands ’ET,#,186’ and ’EMID,ADD,ALL’.
Please refer to ANSYS user’s manual.
```

5.8.1.12. ANSYS Mechanical Files

The **Ansys Mapper** format can be selected when you are interested in performing additional simulations in ANSYS Mechanical (e.g., in an Explicit Dynamics analysis system). For example, you could use the results of your POLYFLOW thermoforming or blow molding simulation in a subsequent Mechanical structural analysis.

Two kinds of output files are generated when the **Ansys Mapper** format is selected. The first kind of file has a `.txt` extension, and provides the coordinates for mesh nodes, along with the associated values of temperature and thickness (note that a node can have more than one thickness value in the case of a multilayer system). The second kind of file has a `.xml` extension, and provides information about the data contained in the `.txt` file. Both kinds of files are in a portable ASCII format. By default, POLYFLOW creates a single `.txt` and `.xml` file for steady-state simulations. For evolution or time-dependent problems, a series of `.txt` and `.xml` files are generated; the names of these files are appended with a number to indicate the step index.

Note that if you want to use your output files to initialize the temperature and the thickness for a Mechanical structural analysis, you need to load the files in the Mechanical system by selecting the thickness option during the geometry definition. For more information, see the Mechanical User's Guide in the Workbench online help, available from the **Help** menu or any of the links in the quick help or sidebar help in Workbench.

5.8.2. Saving Data at a Specified Point

There may be cases where you want to keep track of the value of one or more solution fields at a specific point in the domain (or at several points). You can do this by defining a data probe in ANSYS POLYDATA. The data probe definition will be used by ANSYS POLYFLOW to save the probe data at specified intervals during the calculation. The procedure for defining a data probe is as follows:

1. In the top-level ANSYS POLYDATA menu, select the **Outputs** menu item.

   Outputs

2. Select **Manage Probe(s) output** near the bottom of the menu.

   Manage Probe(s) output

You can specify a probe file name, a location, and a mode of probing (**Moving node** or **Fixed geometry location**). ANSYS POLYFLOW creates a set of `.prb` files which contains the evolution of a field as a function of time (or S, if evolution task). The mode of probing are as follows:

- **Moving node**: The closest node to the given location is determined in the initial configuration. You can get the evolution of fields for the closest node as a function of S or time.
- **Fixed geometry location**: You should determine value of the fields at the given location at every time/evolution step. As the mesh can deform, value of the fields will not be fixed at a given node.

3. Select **Creation of a new probe** to define a new data probe.

   Creation of a new probe

4. Specify the naming convention for the probe files, using the **Change prefix of probe files** menu item.

   Change prefix of probe files

   The probe file prefix will be used at the beginning of each probe file name. If, for example, the prefix is `myrun1` and the simulation contains velocity, pressure, and temperature fields, ANSYS POLYFLOW will write the following files:

   - `myrun1_XYZ.prb` for the coordinates field
   - `myrun1_UV.prb` for the velocity field
   - `myrun1_P.prb` for the pressure field
   - `myrun1_PSI.prb` for the stream function field (2D only)
   - `myrun1_T.prb` for the temperature field

5. Specify the location of the probe, using the **Change probe position** menu item.

   Change probe position

   When prompted for each, specify the value of the x, y, and (for 3D) z coordinates of the probe position.

6. Select **Upper level menu** to return to the **Manage probe(s) output** menu. You will see a new menu item for the probe you created (e.g., **1-st probe**).

   If you need to modify the probe's definition, select this item and repeat the appropriate step. If you want to create additional probes, repeat all steps starting with the selection of **Creation of a new probe**.

During a steady-state calculation, ANSYS POLYFLOW will save the probe files once at the end of the calculation. For a time-dependent or evolution calculation, the probe files will be saved according to your specification of the output triggers, described in *Output for Time-Dependent and Evolution Calculations* (p. 126). The files are written in ASCII format, so you can examine them yourself using a text editor, or manipulate the contents for use in an external plotting program or spreadsheet.

5.8.3. Output for Time-Dependent and Evolution Calculations

For time-dependent and evolution calculations, you can ask ANSYS POLYFLOW to save the output files (mesh and solution data, discussed in *Exporting Mesh and Solution Data* (p. 117), or probe data, discussed in *Saving Data at a Specified Point* (p. 124)) at specified intervals during the calculation. This will allow you to examine the progress of the solution, and study the time or evolution history of solution variables.

The interval for saving output files is specified in ANSYS POLYDATA as follows:

1. In the top-level ANSYS POLYDATA menu, select the **Outputs** menu item.

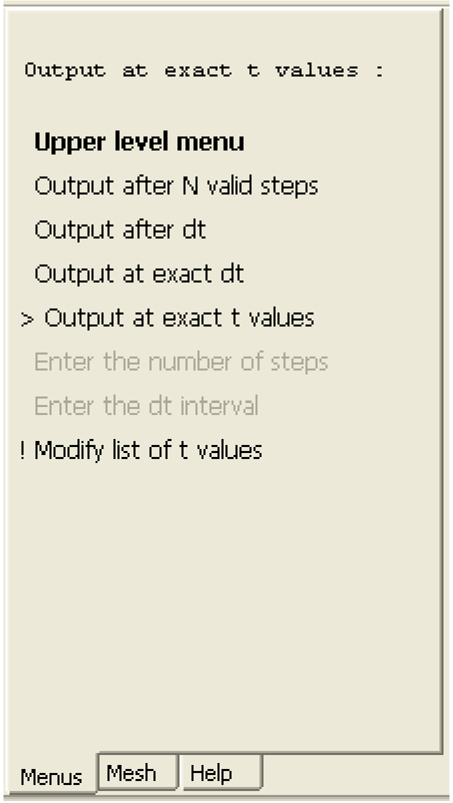
   Outputs

2. Select the **Output Triggering** menu item.

   Output Triggering

   This menu item will appear only for time-dependent or evolution problems.

   Figure 5.1 Output Triggering at t Values


Output at exact t values :
Upper level menu
Output after N valid steps
Output after dt
Output at exact dt
> Output at exact t values
Enter the number of steps
Enter the dt interval
! Modify list of t values
Menus
Mesh
Help

3. Specify when ANSYS POLYFLOW should save the output files. There are four options:

   - **Output after N valid steps** will save the output after every N time steps or evolution steps. After you select this option, choose **Enter the number of steps** to specify the value of N.

     For example, if you specify $N = 10$, ANSYS POLYFLOW will save the output files after every 10 time steps or evolution steps.

- **Output after dt** or **Output after ds** will save the output at intervals greater than or equal to a specified time or evolution-variable interval. After you select this option, choose **Enter the dt interval** or **Enter the ds interval** to specify the value of the time interval or the evolution-variable interval.

  For example, if you specify **dt**=0.02 or **ds**=0.02, ANSYS POLYFLOW will save the output files when the difference between the current time (or evolution-variable value) and the time (or evolution-variable value) of the last output is greater than or equal to the specified interval.

- **Output at exact dt** or **Output after ds** will save the output at intervals exactly equal to a specified time or evolution-variable interval. After you select this option, choose **Enter the dt interval** or **Enter the ds interval** to specify the value of the time interval or the evolution-variable interval.

  For example, if you specify **dt**=0.02 or **ds**=0.02, ANSYS POLYFLOW will save the output files when the difference between the current time (or evolution-variable value) and the time (or evolution-variable value) of the last output is exactly equal to the specified interval.

  Using this option will force the solver to compute a solution at $t_0 + dt$, $t_0 + 2dt$, $t_0 + 3dt$, etc., if this computation has not already been performed. If you do not need the results at such specific intervals, it may be better to use the **Output after dt** or **Output after ds option**, which will not require any additional calculations.

- **Output at exact t (or s) values** will save the results at the set of time (or evolution-variable) values specified by you. To specify the values, select **Modify list of t (or s) values** option. The results will be saved only at the specified values.

5.9. Filename Syntax

In addition to the control over file names that ANSYS POLYDATA gives when you save the data file (see *Writing a Data File* (p. 107)), you can modify the default general naming conventions by selecting the **Filename syntax** menu item in the top-level ANSYS POLYDATA menu.

Filename syntax

5.9.1. Naming Conventions

To check or modify the default naming conventions, choose the **File preferences** menu item next.

File preferences

In the **File preferences** menu, you can specify the naming convention for files by selecting one of the following options:

- **Short names** specifies that filenames will not consist of a prefix and suffix.

- **Prefix** specifies that filenames will consist of a prefix and a suffix (`prefix.suffix`). When you choose this option, you will be asked to specify the prefix. The prefix can have up to 80 characters, and the suffix is a three-character string that corresponds to the short name of the file (see *Table 5.1: Files Written and Read by ANSYS POLYDATA and ANSYS POLYFLOW* (p. 103)).

Chapter 6: Unit Systems

Information about unit systems for ANSYS POLYFLOW is presented in this chapter.

6.1. Overview of Units

By default, ANSYS POLYDATA (as well as ANSYS POLYFLOW) ignores the system of units that you are using. Hence, by default, all values must be specified in accordance with your system of units, and the numerical values will be interpreted accordingly. In other words, if you do not explicitly specify a system of units in ANSYS POLYDATA, you can use any system you want, as long as you are consistent.

A system of units in physics consists of a length unit, a mass unit, and a time unit. In most applications, the second (s) is used for the time unit. For convenience, the unit for length and mass, as well as the unit for temperature, may depend on the application being considered. You may therefore need to switch from one system of units to another.

The most common systems of units are the international MKS (meter, kilogram, second) unit system and its subset, the CGS (centimeter, gram, second) system. For temperature, the most commonly used units are Celsius and Kelvin. In an ANSYS POLYDATA session, you can switch from one system of units to a different one (e.g., from the MKS system to the CGS system, or vice versa). You can also specify different units for individual quantities (length, mass, time, temperature, or electric current), rather than using a standard system of units. This allows you to customize the units to match your data, thus simplifying the problem setup procedure by eliminating the need to convert the units yourself.

Important

Note that switching from one system of units to another will affect only inputs related to material properties. Parameters related to boundary conditions (velocity, flow rate, temperature, force, surface tension coefficient, etc.) will not be affected by a change in the system of units. In addition, if dependences are defined through the PMAT feature, the associated parameters will also not be affected by a change in the system of units. See *Restrictions on Units* (p. 131) for details.

6.2. Converting to a New Unit System

When converting to a new system of units, you have to specify the current system as well as the new one. Here, and only here, ANSYS POLYDATA assumes by default that the current system of units is the MKS system, although this is without any influence on the data you have entered. It is only when you specify a change in the system of units that the numerical values of the material data will be altered.

If you want to change to a new unit system (e.g., if you want to change from MKS to CGS), or keep the current system of units but change the unit just for one quantity (e.g., use Fahrenheit for temperature instead of Kelvin), follow the steps below.

> **Important**
>
> You should be particularly careful to ensure that the length unit you use is the same as the one used when you generated the mesh. If the mesh length unit is not appropriate, you can use ANSYS POLYFUSE to scale the mesh. See *Translating, Rotating, and Scaling a Mesh* (p. 154) for details.

1. Select **Change System of Units** in the **Material data** menu.

   Change System of Units

   ANSYS POLYDATA will show a menu that includes the currently assumed or selected units for length, mass, time, temperature, and electric current.

2. Specify the system of units that you are currently using. By default, the current system is MKSA + Kelvin unit for temperature. If you have been entering inputs using a different unit system (e.g., if you have been entering values based on a mass unit of grams instead of kilograms), you need to modify the current system to ensure that your past inputs are converted properly to the new unit system.

   a. Select **Define current system of Units**

      Define current system of Units

      (Note that this menu item is called **Modify current system of Units** if you have selected it before.)

      The new menu presents a set of common systems of units. Select the system that corresponds to your system of units. The standard systems are:

      - **American #1, using inch, pound, second, Ampere and Fahrenheit units**
      - **American #2, using feet, pound, second, Ampere and Fahrenheit units**
      - **Metric #1, using millimeter, gram, second, milliAmpere and Celsius units**
      - **Metric #2, using centimeter, gram, second, Ampere and Celsius units (CGSA + Celsius)**
      - **Metric #3, using meter, kilogram, second, Ampere and Celsius units (MKSA + Celsius)**
      - **Metric #4, using meter, kilogram, second, Ampere and Kelvin units (MKSA + Kelvin, the default system)**

   b. If none of the options corresponds to your case, select each quantity to be modified and choose its correct unit.

   c. When you are done defining units, select **Upper level menu** and review the current system of units. Your current system of units will be named `user defined system` in the **comment** part of the menu if it does not correspond to a standard system. If you want to return to a standard system, then select it in the menu.

3. Specify the system of units to which you want to convert your inputs (and continue to use for subsequent inputs after the conversion).

a. Select **Define new system of Units**.

   Define new system of Units

   (Note that this menu item will be called **Modify new system of Units** if you have selected it before.)

   The new menu presents the same set of common systems of units as described above. If one of the common systems corresponds to your system, then select it.

b. If none of the options corresponds to your case, select each quantity to be modified and choose its unit.

c. When you are done defining units, select **Upper level menu** and review the new system of units. Your new system of units will be named `user defined system` in the **comment** part of the menu if it does not correspond to a standard one. If you want to return to a standard system, then select it in the menu.

4. Convert from the current system of units to the new system.

   RUN

   ANSYS POLYDATA will convert the units, and open a separate window to report the progress and any errors or other messages. In particular, old and new values for the material parameters are displayed, and a warning is issued in connection with any data that have not been modified due to the change in units.

5. Press the **Return** key on your keyboard to close the window. ANSYS POLYDATA will then ask you to confirm that you want to change to the new unit system.

6. Click **Yes** to accept the new unit system (or click **No** to return to the original system of units).

The new system is now the current system. All future inputs to ANSYS POLYDATA will be in the new unit system.

6.3. Restrictions on Units

The following units are *not* converted when the system of units is changed:

- mesh units
- parameters of boundary condition laws
- parameters for evolution or time-marching schemes
- parameters that depend on field variables
- user-defined functions (Also, UDFs must be written in a system of units that is consistent with the rest of the problem definition, as any other material law.)
- initial thickness (for film casting or blow molding)
- initial temperature (for nonisothermal simulations)
- initial concentration (for species transport)
- material data for mass transfer sub-tasks

Chapter 7: Meshes

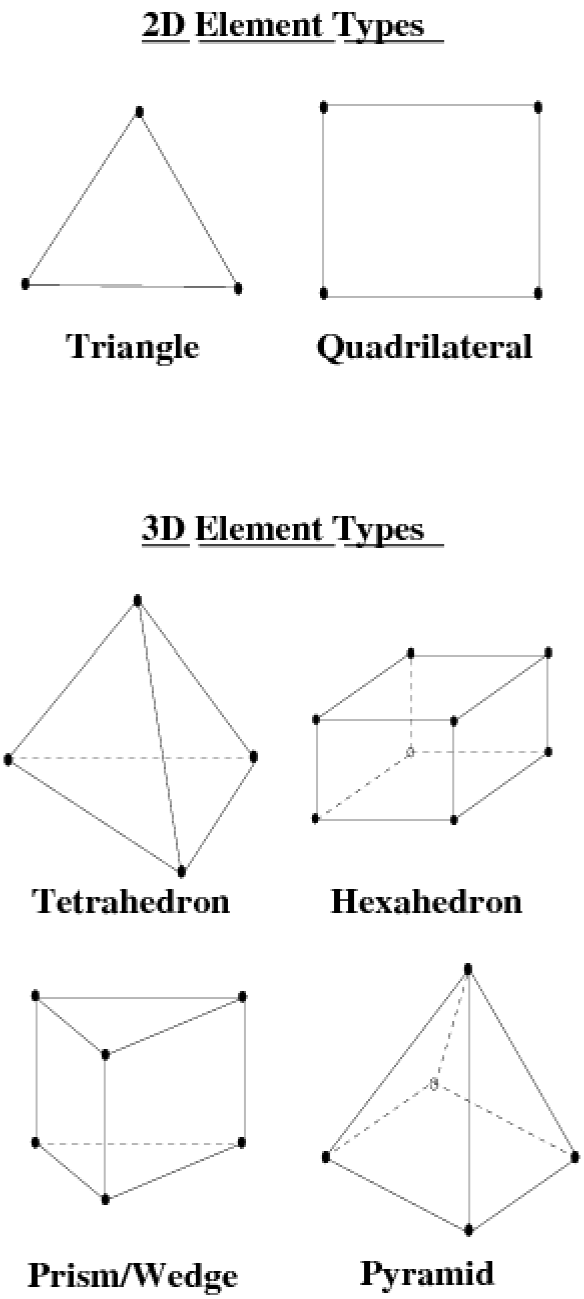
This chapter contains information about creating subdomains and boundary sets for ANSYS POLYFLOW using several different mesh generators, as well as information about interpolating, combining, and examining meshes.

7.1. Mesh Topologies

Since ANSYS POLYFLOW is an unstructured solver, it uses internal data structures to assign an order to the elements, faces, and mesh points in a mesh and to maintain contact between adjacent elements. It does not, therefore, require i, j, k indexing to locate neighboring elements. This gives you the flexibility to use the mesh topology that is best for your problem, since the solver does not force an overall structure or topology on the mesh.

In 2D, quadrilateral and triangular elements are accepted, and in 3D, hexahedral, tetrahedral, pyramid, and wedge elements can be used. (*Figure 7.1* (p. 134) depicts each of these element types.) Hybrid meshes containing quadrilateral and triangular elements or hexahedral, tetrahedral, pyramid, and wedge elements are also acceptable.

Figure 7.1 Element Types



Some examples of meshes that are valid for ANSYS POLYFLOW are presented in *Examples of Acceptable Mesh Topologies* (p. 135). *Choosing the Appropriate Mesh Type* (p. 137) explains how to choose the mesh type that is best suited for your problem.

7.1.1. PMeshes

PMeshes are topological groups of element segments (in 2D) and element segments or faces (in 3D) located inside a mesh and are used to facilitate the definition of the problem at a later stage. The name for such groups derives from the phrase, "pointer to mesh". PMeshes are used as a replacement for boundary sets when a model needs to be defined on an internal part of a mesh in a dimension lower than the dimension of the subdomains (e.g., in 1D or 2D for a 3D subdomain). These PMeshes can be either internal to the domain (located on a boundary or not) or external in a non-manifold topology.

PMeshes can be created in GAMBIT, ANSYS ICEM CFD, ANSYS Meshing, POLYMESH, PATRAN, and I-deas, as described in *Defining PMeshes in GAMBIT* (p. 139), `.poly` *Meshes Created with ANSYS ICEM CFD or ANSYS Meshing* (p. 139), *Elements of Mixed Dimension and PMesh Generation* (p. 142), *Defining PMeshes in 2D* (p. 143), and *Defining PMeshes in 3D* (p. 143). See *Specifying Conditions Using Sub-Models* (p. 192) for information about defining conditions on a PMesh using sub-models. PMeshes are typically used for bubbling in glass applications and for modeling internal radiation, as described in *Bubblers* (p. 490) and *User Inputs for Internal Radiation Model* (p. 297), respectively.

7.1.2. Examples of Acceptable Mesh Topologies

As mentioned above in *Mesh Topologies* (p. 133), ANSYS POLYFLOW can solve problems using a wide variety of meshes. *Figure 7.2* (p. 136) and *Figure 7.3* (p. 137) show examples of meshes that are valid for ANSYS POLYFLOW.

Meshes with zero-thickness walls (shell elements), triangular, tetrahedral, quadrilateral, and hexahedral meshes can all be used.

Figure 7.2 Hybrid Mesh with Tetrahedra and Hexahedra

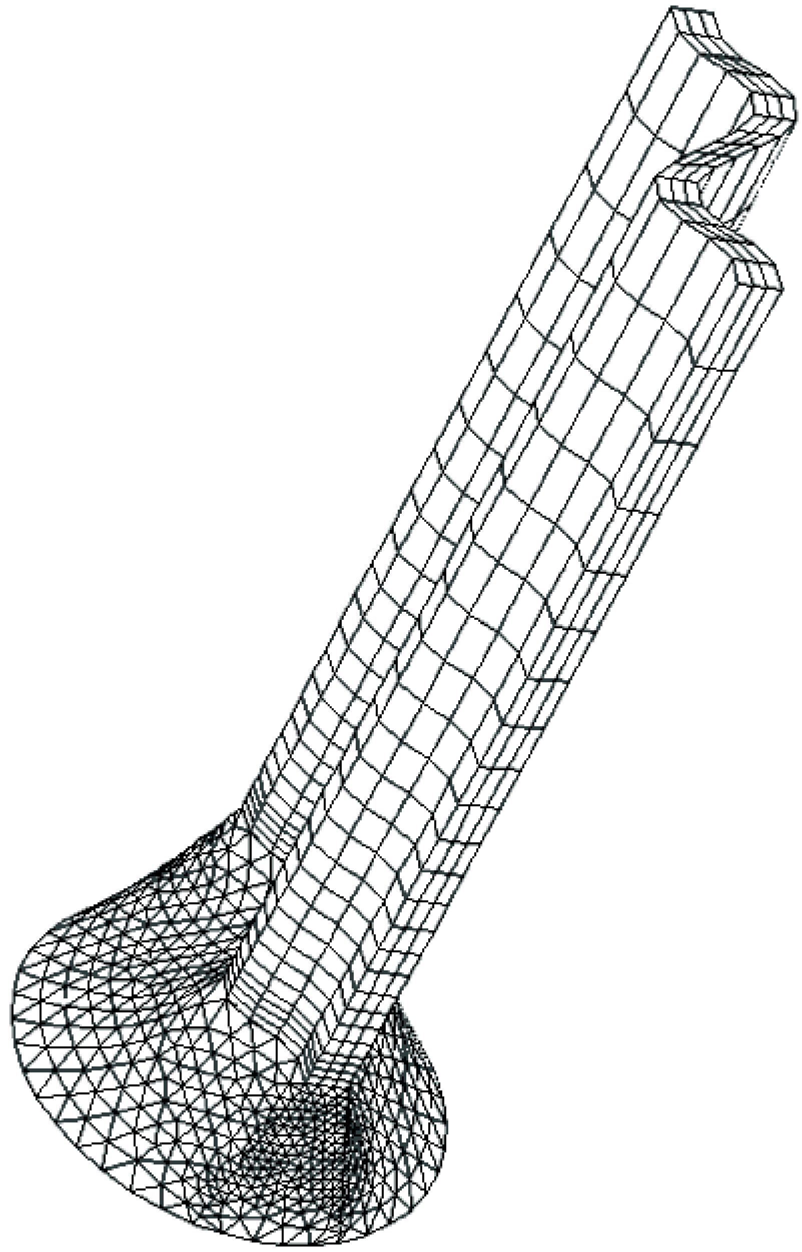
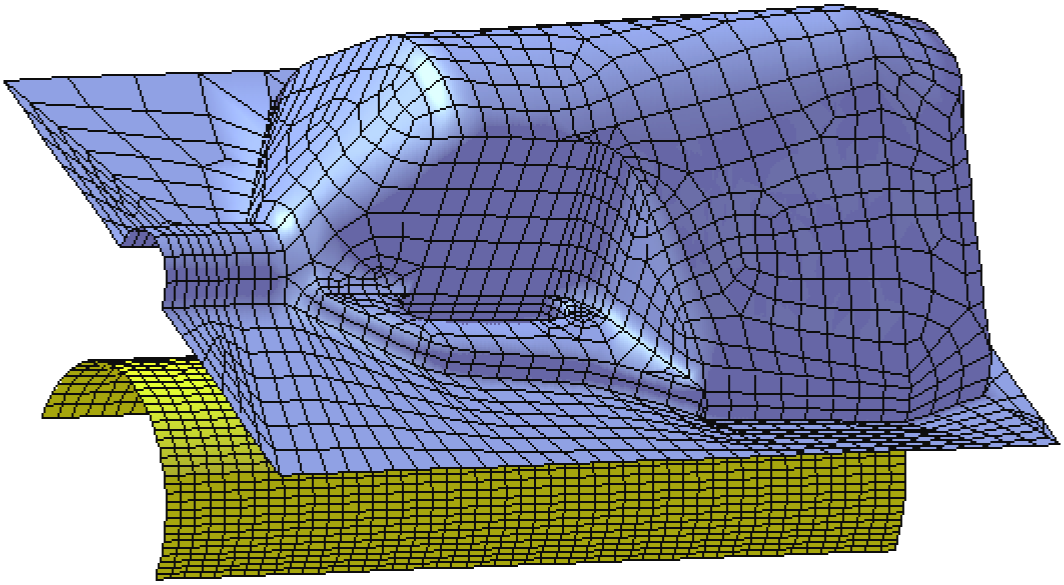


Figure 7.3 Mesh with Shell Elements



7.1.3. Choosing the Appropriate Mesh Type

ANSYS POLYFLOW can use meshes comprised of triangular or quadrilateral elements (or a combination of the two) in 2D, and tetrahedral, hexahedral, pyramid, or wedge elements (or a combination of these) in 3D. The choice of which mesh type to use will depend on your application. When choosing your mesh type, you should consider the following issues:

- Setup time
- Computational expense
- Application being modeled

To clarify the trade-offs inherent in your choice of mesh type, these issues are discussed further.

7.1.3.1. Setup Time

Many flow problems solved in engineering practice involve complex geometries. The creation of structured or block-structured meshes (consisting of quadrilateral or hexahedral elements) for such problems can be extremely time-consuming, if not impossible. Setup time for complex geometries is, therefore, the major motivation for using unstructured meshes employing triangular or tetrahedral elements. If your geometry is relatively simple, however, there may be no clear saving in setup time with either approach.

7.1.3.2. Computational Expense

For many typical geometries, the use of triangular or tetrahedral elements leads to a significantly higher number of elements than for quadrilateral or hexahedral elements to obtain similar accuracy on the same geometry. This higher number of elements results in a higher CPU cost. In this respect, a mesh made of triangles or tetrahedra is often less than optimal in terms of CPU cost.

The most efficient mesh is one that includes quadrilaterals or hexahedra in regions where the geometry is not too complex, and triangles or tetrahedra in regions where it is difficult to place quadrilateral or hexahedral elements.

7.1.3.3. Application Being Modeled

Some 3D remeshing techniques such as Optimesh, streamwise, and planar require the nodes to be defined within a plane. In addition, due to the nature of the remeshing technique, the node will be relocated in the plane orthogonal to the extrusion direction only. Therefore, such remeshing techniques require the use of brick (hexahedral) or prism elements in the extrudate (although tetrahedra and pyramids can be used in the region that is not being remeshed).

7.2. Meshes Created with GAMBIT

GAMBIT is the standard geometry and mesh generator for ANSYS POLYFLOW. When you use it to create a mesh, you will need to specify that the mesh is for ANSYS POLYFLOW *before* you define boundary sets, so that GAMBIT will write the boundary information properly to the neutral file. Some information about this process is provided here; see the GAMBIT Modeling Guide for details. See *Mesh Files Created by Other Programs* (p. 106) for information about converting GAMBIT neutral files.

7.2.1. Subdomains in GAMBIT

In GAMBIT, you must group elements belonging to the same subdomain using the **Continuum** icon. You will select a geometry, and all the elements belonging to this geometry will be transferred to this new group. The name of the group is arbitrary, but you will need to include in it a number that will become the ANSYS POLYFLOW subdomain index. For example, the following names are valid:

- SD2 (maps to SD2)
- Fluid Region 3 (maps to SD3)
- Region 4 Fluid (maps to SD4)

When the GAMBIT neutral file is converted to an ANSYS POLYFLOW mesh (as described in *Mesh Files Created by Other Programs* (p. 106)), these names will be mapped to ANSYS POLYFLOW subdomain indices. Any continuum with an invalid name will be mapped to the “additional subdomain". At the end of the conversion, the mapping between the GAMBIT names and the ANSYS POLYFLOW names will be reported. If any volumes have not been properly assigned to a continuum, they will be mapped to an additional subdomain.

7.2.2. Boundary Sets in GAMBIT

To define boundary sets in GAMBIT, you will select the segments (in 2D) or faces (in 3D) that belong to a particular boundary set using the **Boundary Condition** icon. The name of the group is arbitrary, but you will need to include in it a number that will become the ANSYS POLYFLOW boundary set index. For example, the following names are valid:

- BS2 (maps to BS2)
- Wall 3 (maps to BS3)
- Region 4 Wall (maps to BS4)

When the GAMBIT neutral file is converted to an ANSYS POLYFLOW mesh (*Mesh Files Created by Other Programs* (p. 106)), these names will be mapped to ANSYS POLYFLOW boundary set indices. Any group with an invalid name will be mapped to the “additional boundary set". After the conversion, the mapping between the GAMBIT names and the ANSYS POLYFLOW names will be reported.

If you select a segment (2D) or a face (3D) in GAMBIT that does not belong to the boundary of the domain, the segment or face will be ignored by the conversion. If any segments or faces on the boundary have not been assigned to a boundary set, they will be mapped to the additional boundary set.

The creation of the additional boundary set is useful in two ways:

- It is a convenient way for you to check if you have forgotten to define any nodal loads.
- If selecting all the nodes that belong to a particular boundary set requires a significant amount of effort in GAMBIT, you can rely on the additional boundary set to collect these unassigned nodes. In this case, however, you are strongly encouraged to verify the number of segments or faces in all boundary sets.

7.2.3. Defining PMeshes in GAMBIT

PMeshes are topological groups of element segments (in 2D) and element segments or faces (in 3D) located inside a mesh, as described in *PMeshes* (p. 134). In 2D, all PMeshes are one-dimensional (i.e., segments in GAMBIT). In 3D, ANSYS POLYFLOW uses 1D and 2D PMeshes while GAMBIT creates only 2D PMeshes (i.e., faces in GAMBIT).

For internal PMeshes, use the **Boundary Condition** icon in GAMBIT; for external PMeshes arranged in some non-manifold topology, use the **Continuum** icon. Assign a new PMesh index to the region that you want to identify as a PMesh. Names must start with PM or pm, or they will be ignored. When you read the GAMBIT mesh into ANSYS POLYDATA, the conversion filter will transform them into PMeshes. Both internal and external PMeshes can be included in the same mesh, provided that different indices are used.

Note that you should not have non-conformal elements (as generated by the Hex Core meshing scheme in GAMBIT) in the GAMBIT neutral file.

7.3. .poly Meshes Created with ANSYS ICEM CFD or ANSYS Meshing

When you generate a mesh using ANSYS ICEM CFD or export a mesh from ANSYS Meshing in the POLYFLOW format, the mesh information is written to a .poly file. You can convert such a .poly file to the ANSYS POLYFLOW format using ANSYS POLYDATA or ANSYS POLYMAN, as described in *Mesh Files Created by Other Programs* (p. 106) and *Importing a .poly File* (p. 38), respectively.

ANSYS Meshing is a mesh generator, whereas ANSYS ICEM CFD can generate both a geometry and a mesh. Both applications allow the generation of surface (boundary set) and volume (subdomain) topologies. Moreover, for 2D geometries you can define 1D internal and external PMeshes, while for 3D geometries you can define 2D external PMeshes and 1D and 2D internal PMeshes. Note that the PMeshes cannot be defined on the border. See *PMeshes* (p. 134) for details about PMeshes.

You do not need to generate midsegment or midface nodes in the mesh generator in order to use these interpolation schemes in ANSYS POLYFLOW. Midside nodes will be added by ANSYS POLYFLOW automatically, as needed.

Note the following with regard to meshes generated by the CutCell method in ANSYS Meshing:

- You must not combine (e.g., using ANSYS POLYFUSE) a mesh that was generated by the CutCell method with another type of mesh, if you intend to use it in an ANSYS POLYFLOW simulation; ANSYS POLYFLOW requires that the mesh you read in consists of a domain in which either every part or no part is a CutCell mesh. Consequently, you cannot use an unaltered CutCell mesh with moving boundaries (e.g., in a free jet region outside of an extrusion die), as the remeshing algorithms require a sliceable mesh, which is typically a swept mesh. To overcome this limitation, you can use POLYDATA to convert a portion of your CutCell mesh into a sliceable mesh, as described in *Generating a Sliceable Free Jet Mesh* (p. 164).

- When generating a CutCell mesh for flow applications, you must carefully check the mesh in order to avoid thin regions in which only one element exists between opposite walls. If such situation occurs, all velocity nodes of such elements have fixed (and generally) null values (i.e., a fixed wall condition): no fluid will cross these elements, leading to artificial obstacles in the flow.
- For CutCell meshes, the interpolation for the velocity field is limited: for a pure CutCell mesh, it must be the linear element; for a portion of a CutCell mesh that has been converted into a sliceable mesh, it can be either the linear element or the mini-element.
- The following tasks are not supported for CutCell meshes:
  - **MIXING task**
  - **Volume Of Fluid problem(s)**
- The following sub-tasks are not supported for CutCell meshes:
  - **Differential viscoelastic isothermal flow problem**
  - **Differential viscoelastic non-isothermal flow problem**
  - **Integral viscoelastic isothermal flow problem**
  - **Integral viscoelastic non-isothermal flow problem**
  - **Film model : Gen. Newtonian isothermal**
  - **Film model : Gen. Newtonian non-isothermal**
  - **Film model : Viscoelastic isothermal**
  - **Film model : Viscoelastic non-isothermal**
  - **Shell model : Gen. Newtonian isothermal**
  - **Shell model : Gen. Newtonian non-isothermal**
  - **Shell model : Viscoelastic isothermal**
  - **Shell model : Viscoelastic non-isothermal**
  - **Isothermal crystallisation**
  - **Non-isothermal crystallisation**
  - **Residual stresses and deformations (Narayanaswamy)**
  - **Internal radiation**
- The adaptive meshing technique is not supported for CutCell meshes.

7.4. Meshes Created with PATRAN or I-deas

Meshes for ANSYS POLYFLOW can also be created using PATRAN or I-deas, which are third-party CAD/CAE packages. PATRAN neutral files and I-deas universal files can be converted by ANSYS POLYDATA, as described in *Mesh Files Created by Other Programs* (p. 106). To ensure proper conversion of the mesh information, you will need to follow the guidelines presented below.

7.4.1. Element and Node Numbering

ANSYS POLYDATA's conversion utility does not modify the PATRAN or I-deas element or node numbering. Since element and node numbering for ANSYS POLYFLOW must start at 1 and must be continuous, you must be sure to number the elements and nodes in this way in PATRAN or I-deas (even though they do not require this themselves). Some numbering errors will be detected and reported by ANSYS POLYDATA during the conversion, but others will only be detected by ANSYS POLYFLOW itself. You should verify that the number of elements and nodes reported after the conversion by ANSYS POLYDATA is identical to the number of elements and nodes reported in the `TOPO` section of the ANSYS POLYFLOW listing file (described in *Contents of a Sample Listing File* (p. 111)). Specifically, the number of `Ele` and `Vert` reported by ANSYS POLYDATA after the conversion should match the number of `bricks` and `nodes` reported for the root mesh in the `TOPO` section.

By default, ANSYS POLYFLOW will optimize the element numbering, which affects the CPU time required for the calculation, before it begins the calculation. If you prefer to perform the optimization yourself in PATRAN or I-deas, and have ANSYS POLYFLOW use that optimization instead of optimizing the mesh itself, you will need to start ANSYS POLYFLOW with the `-opt msh` command line option, as described in *Using Optimized or Decomposed Mesh* (p. 149).

If you perform the optimization in PATRAN, use the Cuthill-McKee algorithm in the `optimize` section. In I-deas, reduce the element wavefront before modifying the element numbering. The reduction will not take place if the element is not modified. Optimizing the average wavefront minimizes the CPU time, and optimizing the maximum wavefront reduces the memory requirements.

7.4.2. Jacobians

ANSYS POLYFLOW requires all element Jacobians to have the same sign. If not, ANSYS POLYDATA will try to reorient the incorrectly-oriented elements geometrically (i.e., on the basis of the coordinates) or topologically (i.e., on the basis of the connectivity numbering) when it converts the PATRAN or I-deas mesh file. If any improperly-oriented elements cannot be corrected, ANSYS POLYDATA will report that the mesh connectivity is incorrect, and you will have to return to PATRAN or I-deas to reorient these elements. ANSYS POLYDATA corrects the orientation for *all* imported meshes, not just for PATRAN and I-deas.

7.4.3. Subdomains

Neither PATRAN nor I-deas understands ANSYS POLYFLOW's concept of subdomains. The subdomain information is converted from the material property ID, which is an element property. If you want to generate more than one subdomain, you should select appropriate elements and change their material property ID. For subdomain 2, for example, change the material property ID to 2. This is done through the PMAT directive in PATRAN, and through modification of the material ID property in I-deas. The subdomain numbering will be reported by ANSYS POLYDATA after the conversion is complete.

7.4.4. Boundary Sets

ANSYS POLYDATA will obtain boundary set information by converting structural nodal loads. The value of the nodal load itself is not important, with the exception that it cannot be zero (because all zero nodal loads will be discarded from the neutral or universal file). The load set index is used as a pointer to the ANSYS POLYFLOW boundary set. You must change the load set index whenever you need to generate a different boundary set. In general, load set 1 will become boundary set 1, and so on. In ANSYS POLYFLOW, nodes located at the intersection between two (or more) boundary sets belong to all of those boundary sets. The same rule must be respected in PATRAN or I-deas: you must include these nodes in *each* load set.

For example, in 2D, a node located at the intersection between BS1 and BS2 must be assigned a (nonzero) nodal load in load set 1, and another (nonzero) nodal load in load set 2. This extends to 3D meshes as well.

Nodal loads defined on internal nodes (i.e., nodes that are not located on the domain boundary) are not taken into account by the conversion utility. Also, if there are segments (in 2D) or faces (in 3D) located on the domain boundary that are not assigned a boundary set index by the end of the conversion, ANSYS POLYDATA will create a new boundary set (which is not linked to any load set) to collect these unassigned segments or faces. This new boundary set is called the "additional boundary set".

The creation of the additional boundary set is useful in two ways:

- It is a convenient way for you to check if you have forgotten to define any nodal loads.
- If selecting all the nodes that belong to a particular boundary set requires a significant amount of effort in PATRAN or I-deas, you can rely on the additional boundary set to collect these unassigned nodes. In this case, however, you are strongly encouraged to verify the number of segments or faces in all boundary sets.

7.4.5. Elements of Mixed Dimension and PMesh Generation

Elements of mixed dimension can be defined in PATRAN and I-deas. ANSYS POLYDATA will convert them to ANSYS POLYFLOW PMeshes (as described in *PMeshes* (p. 134)). Elements of the highest spatial dimension (e.g., 3 in 3D) will first be considered; they will determine the connectivity of the mesh and the number of ANSYS POLYFLOW "elements".

Elements of lower spatial dimension will then be considered (typically, face or line elements in 3D). These elements, when properly connected through nodes to elements of higher dimension, will generate pointers to existing faces or segments. The list of these pointers will become a PMesh, which will be listed in the translator output report (with numbers starting at 1 for each spatial dimension).

In order to distinguish between several PMeshes of the same dimension, the material property ID (used for subdomains, as described in *Subdomains* (p. 141)) is used to collect faces and segments into separate lists. Whenever a candidate for a PMesh is found, the converter checks for the existence of a PMesh of the same dimension with the same material property ID. If this list exists, the element is added to it.

If not, the list is first created and then the element is added to it. PMeshes are numbered in ascending order of the material property indices, so they can be recognized easily.

7.4.6. Notes for I-deas Master Series Users

I-deas Master Series always saves the coordinates of the FEM mesh to the universal file in SI units. You should keep this in mind when setting up, solving, and postprocessing your model, to prevent unexpected conversions and incorrect results.

If, for example, the geometry for the problem you want to solve in ANSYS POLYFLOW is 10 cm long, you should specify a length of 10 m for your mesh in I-deas. When you convert the mesh in ANSYS POLYDATA, it will become a dimensionless length of 10.

As described in *Unit Systems* (p. 129), you can consider this to be the length unit of your choice, as long as you are consistent. Similarly, if the width of the 10-cm-long mesh is 55 mm, think of this as 5.5 cm and specify a width of 5.5 m in I-deas.

If you find that your mesh has incorrect dimensions, you can use ANSYS POLYFUSE to scale it. See *Translating, Rotating, and Scaling a Mesh* (p. 154) for details.

7.5. Meshes Created with POLYMESH

POLYMESH is a mesh generator that was used to create meshes for ANSYS POLYFLOW before the introduction of GAMBIT. Mesh files created with POLYMESH can be read directly into ANSYS POLYDATA, as described in *Mesh Files Created by POLYMESH or Converted by ANSYS POLYDATA* (p. 105).

7.5.1. 2D Meshes

In POLYMESH 2D, subdomains are defined on a macroelement basis. This allows you to specify an index for each macroelement. Boundary sets can be defined at the level of the macrosegment (default) or at the level of the individual segments of the mesh.

7.5.1.1. Defining PMeshes in 2D

In 2D, all PMeshes (described in *PMeshes* (p. 134)) are one-dimensional, and a series of PMeshes identified by their numbers can be defined. In order to define a new PMesh, you will specify a series of macrovertices connected by macrosegments. All mesh segments located on this line will belong to the PMesh when the mesh is saved.

7.5.2. 3D Meshes

In POLYMESH 3D, subdomains are defined on a macroelement basis. This allows you to specify an index for each macroelement. Boundary sets are defined at the level of the macroface. For each macroface that belongs to the domain boundary, POLYMESH prompts you for a boundary set index. Macrofaces can be identified by the list of their respective macrovertices.

7.5.2.1. Defining PMeshes in 3D

In 3D, PMeshes (described in *PMeshes* (p. 134)) can be one- or two-dimensional. For each dimension, a series of PMeshes uniquely identified by their numbers can be defined. When defining a 1D PMesh, you will specify a series of macrovertices connected by macrosegments. All mesh segments located on this line will belong to the PMesh when the mesh is saved. When defining a 2D PMesh, you will specify a series of faces (defined by three or four macrovertices). All faces must be connected by at least one segment. All mesh faces generated on these macrofaces will eventually belong to the PMesh.

7.5.3. Element Numbering

When you save a mesh from POLYMESH, it will automatically optimize the element numbering, which affects the CPU time required for the calculation. By default, ANSYS POLYFLOW will reoptimize the element numbering for the mesh before it begins the calculation. If you prefer to have ANSYS POLYFLOW use POLYMESH's optimization instead of optimizing the mesh itself, you will need to start ANSYS POLYFLOW with the `-opt msh` command line option, as described in *Using Optimized or Decomposed Mesh* (p. 149).

7.6. Meshes Created with HYPERMESH

You can create meshes for ANSYS POLYFLOW using HYPERMESH, which is a third-party **CAD/CAE** application (preprocessor for HyperWorks). You can convert HYPERMESH `*.hma` or `*.hmascii` files using ANSYS POLYDATA as described in *Mesh Files Created by Other Programs* (p. 106). To ensure proper conversion of the mesh information, you need to follow the guidelines presented in the following sections.

7.6.1. Element and Node Numbering

The conversion utility of ANSYS POLYDATA does not modify the HYPERMESH element or node numbering. Since element and node numbering for ANSYS POLYFLOW must start at one and must be continuous, ensure to number the elements and nodes in this way in HYPERMESH.

ANSYS POLYDATA will detect and report some numbering errors during the conversion, but the other errors will only be detected by ANSYS POLYFLOW. Verify that the number of elements and nodes reported after the conversion by ANSYS POLYDATA is identical to the number of elements and nodes reported in the **TOPO** section of the ANSYS POLYFLOW listing file (described in *Contents of a Sample Listing File* (p. 111)).

Note

Ensure that the number of `Ele` and `Vert` reported by ANSYS POLYDATA after the conversion matches the number of bricks and nodes reported for the root mesh in the `TOPO` section.

7.6.1.1. Meshes

By default, ANSYS POLYFLOW will optimize the element numbering, which affects the CPU time required for the calculation, before it begins the calculation. If you prefer to perform the optimization yourself in HYPERMESH, and have ANSYS POLYFLOW use that optimization instead of optimizing the mesh itself, then start ANSYS POLYFLOW with the `-opt msh` command line option, as described in *Using Optimized or Decomposed Mesh* (p. 149). If you perform the optimization in HYPERMESH (if any), use a method of element renumbering that minimizes average wavefront to reduce the CPU time of your simulation, and that minimizes the maximum wavefront to reduce the memory requirements.

7.6.2. Jacobians

ANSYS POLYFLOW requires all element Jacobians to have the same sign. If not, when converting the HYPERMESH mesh file, ANSYS POLYDATA will try to reorient the incorrectly-oriented elements:

- Geometrically (i.e., on the basis of the coordinates)
- Topologically (i.e., on the basis of the connectivity numbering).

If any improperly-oriented elements cannot be corrected, ANSYS POLYDATA will report that the mesh connectivity is incorrect, and you will have to return to HYPERMESH to reorient these elements.

Note

ANSYS POLYDATA corrects the orientation for all imported meshes, not just for HYPERMESH.

7.6.3. Subdomains and Boundary Sets

HYPERMESH does not understand the concept of subdomains or boundaries of ANSYS POLYFLOW. So, in HYPERMESH you have to specify the components. Each component is a tag (name) and contains a set of elements. The translator will not allow you to mix elements of different dimensions, example, triangle and hexagons, in the same component.

- For 2D meshes, components constituted with segments (`bar2`, in the naming of HYPERMESH) will be considered as boundaries, while components constituted with triangles and quads (`tri3` and `quad4` in the naming of HYPERMESH) will be considered as subdomains.
- For 3D meshes, components constituted with triangles and quads (`tri3` and `quad4`, in the naming of HYPERMESH) will be considered as boundaries, while components constituted with tets, wedges, pyramids and hexes (`tetra4`, `pyramid5`, `penta6` and `hexa8` in the naming of HYPERMESH) will be considered as subdomains.

It is not allowed to have elements owned by two components, or to define internal boundaries. The names given in HYPERMESH will be replaced during conversion by a subdomain id or boundary id, in the order with which you find the components in the file. The subdomain and boundary numbering will be reported by ANSYS POLYDATA after the conversion is complete. For more information about creating components, refer to HYPERMESH user's manual.

7.6.4. Restrictions

It is not possible to import `PMeshes` or internal boundaries using this version of the converter. It is recommended not to define internal boundaries because you can create an internal boundary as the intersection of two subdomains. Moreover, only the linear elements, `bar2`, `tri3`, `quad4`, `tetra4`, `pyramid5`, `penta6`, and `hexa8` are recognized. Hence quadratic or serendipity elements (e.g., `hexa20`, `quad8` etc.) are not accepted.

7.7. ANSYS FLUENT Meshes Created with ANSYS Meshing, TGrid, and GAMBIT

When you generate a mesh using TGrid and GAMBIT or export a mesh from ANSYS Meshing in the FLUENT format, the mesh information is written as an ANSYS FLUENT mesh (i.e., a `.msh` file). You can convert such a `.msh` file to the ANSYS POLYFLOW format using ANSYS POLYDATA, as described in *Mesh Files Created by Other Programs* (p. 106).

ANSYS Meshing, TGrid, and GAMBIT allow the generation of surface (boundary set) and volume (subdomain) topologies. The names of the different topological entities (subdomains and boundaries) will transfer to the ANSYS POLYFLOW mesh file. It is a good practice to provide mnemonic names that help to easily identify the different geometric entities. Note that the names are limited to a maximum of 12 characters.

When working with ANSYS FLUENT meshes in ANSYS POLYFLOW, note the following limitations:

- Contrary to ANSYS FLUENT, ANSYS POLYFLOW applications cannot handle polyhedral elements. If such elements are found, an error message will inform you that the translation has been aborted.
- Only the external boundaries and subdomains will be saved in the ANSYS POLYFLOW mesh file; internal zones and PMeshes will be skipped. If you wish to retain the PMeshes, you should instead use GAMBIT to generate a GAMBIT neutral file or export a mesh from ANSYS Meshing in the POLYFLOW format (i.e., a `.poly` file). For information about these kinds of files, see *Meshes Created with GAMBIT* (p. 138) and *`.poly` Meshes Created with ANSYS ICEM CFD or ANSYS Meshing* (p. 139), respectively.
- Note that POLYDATA cannot read a `.msh` file that has non-conformal elements (as generated by the CutCell method in ANSYS Meshing). If you wish to use a CutCell mesh in your POLYFLOW simulation, you should instead export the mesh from ANSYS Meshing in the POLYFLOW format (i.e., generate a `.poly` file). See *`.poly` Meshes Created with ANSYS ICEM CFD or ANSYS Meshing* (p. 139) for more information.

7.8. Mesh Decomposition and Optimization

Optimization of the element numbering is critical in order to obtain the best performance from the ANSYS POLYFLOW solver. This optimization is performed automatically by ANSYS POLYFLOW before it begins a calculation. It is also possible to optimize the mesh in some mesh generators (I-deas, PATRAN, and POLYMESH), or in ANSYS POLYDATA, and have ANSYS POLYFLOW use that optimized mesh instead of performing the optimization itself.

For large 3D meshes, decomposition of the mesh into sub-parts is another method used by ANSYS POLYFLOW to improve the solver performance. Typically, on a 3D mesh, you can expect the CPU time and memory usage to be reduced by a factor of 2 to 10, depending on the mesh size and topology.

ANSYS POLYFLOW takes care of mesh optimization and decomposition for you, but you also have the option to perform them yourself, either in your mesh generator or in ANSYS POLYDATA.

- See *About the Optimization of Element Numbering* (p. 146) and *Decomposing and Optimizing the Mesh* (p. 147) for information about optimizing and decomposing a mesh with ANSYS POLYDATA.
- *Using Optimized or Decomposed Mesh* (p. 149) describes how to tell ANSYS POLYFLOW to use your optimized or decomposed mesh, rather than performing the optimization or decomposition itself.
- *Specifying the Number of Sub-Parts for Decomposition* (p. 149) describes how to specify the number of sub-parts into which ANSYS POLYFLOW should decompose the mesh (leaving the rest of the decomposition to be handled automatically by ANSYS POLYFLOW).

7.8.1. About the Optimization of Element Numbering

Element numbering optimization is an NP-complete problem. This means that it is impossible to prove that an element numbering scheme is optimal without evaluating all possible permutations. Since the evaluation of all possible permutations is impractical, a heuristic method is used instead. ANSYS POLYDATA tests several possibilities, and then selects the best numbering solution and writes it to the specified mesh file.

Two optimization methods are available: the reverse Cuthill-McKee algorithm and the Fiedler vector technique implemented in the Chaco algorithm (developed and licensed by Sandia National Laboratories).

The reverse Cuthill-McKee algorithm is provided with the standard release of ANSYS POLYFLOW, and Chaco is available as an additional licensing option. The reverse Cuthill-McKee algorithm generally yields adequate results for all but the most complex meshes. For very complex meshes with a large number of elements, the Fiedler vector method in Chaco is recommended. The Fiedler vector method is also recommended if you are using the Chaco decomposition method.

7.8.2. About Domain Decomposition

If you are using the parallel version of the ANSYS POLYFLOW solver, you will need to divide the computational domain into sub-parts. This decomposition into sub-parts can also be used to reduce the CPU cost of a calculation using the standard serial solver. Two decomposition methods are available: Metis (developed at the University of Minnesota) and Chaco (developed and licensed by Sandia National Laboratories). Metis is provided with the standard release of ANSYS POLYFLOW, and Chaco is available as an additional option. They both yield good results, but Chaco may provide better results in some cases.

The number of sub-parts must be a power of 2. Note that the number of sub-parts can be different from the number of processors used by the parallel solver. The number of sub-parts must be higher than the number of active processors, which is in turn arbitrary and not constrained to be a power of 2. It is not uncommon to solve a large 3D problem with 512 sub-parts on, say, 4 or 8 processors or even on a single-processor machine.

7.8.3. Decomposing and Optimizing the Mesh

The decomposition and optimization is generally (but not necessarily) performed *before* you read the mesh into ANSYS POLYDATA. Once the decomposed and optimized mesh has been saved, you can read it in and proceed with the problem setup. The steps for optimizing the element numbering and decomposing the mesh are as follows:

1. Select the **Mesh decomposition and optimisation** menu item.

   Mesh decomposition and optimisation

2. Define the decomposition:

   a. If both decomposition methods are available, select the one you want to use by choosing the appropriate menu item:

      Select Metis decomposer

      or

      Select Chaco decomposer

      To disable decomposition, select the **Disable decomposition into sub-parts** menu item. To reenable it, you can choose **Enable decomposition into sub-parts**.

      Disabling decomposition is not recommended for large 3D meshes. For simple 2D meshes, however, decomposition will not result in significant performance improvements, so you can disable it; you should, however, still optimize the element numbering, which significantly affects the solver performance, even in 2D.

   b. Modify the number of sub-parts, if you want to change from the default of 16 sub-parts.

      Modify the number of sub-parts

      Typically, optimal performance is obtained when the number of elements in each sub-part is between 250 and 500. For example, on a mesh with 20,000 elements, the fastest solution is obtained with 16 or 32 sub-parts. Note that the optimal number of sub-parts does not depend on the number of variables being solved in each element. While switching from a generalized Newtonian flow problem to a viscoelastic flow problem increases the number of solution variables significantly, the optimal number of sub-parts will remain the same.

3. Define the optimization:

   a. If both optimization methods are available, select the one you want to use by choosing the appropriate menu item:

      Select Cuthill McKee optimiser

or

Select Chaco optimiser

If you are using the Chaco decomposition method, the Chaco optimization method is recommended.

To disable optimization altogether, select the **Disable numbering optimisation** menu item. To reenable it, you can choose **Enable numbering optimisation**. Disabling the numbering optimization is *not* recommended.

b. Indicate whether the average or the maximum frontal width should be optimized by selecting the appropriate menu item:

Optimize the maximum frontal width (memory size)

or

Optimize the average frontal width (CPU time)

Optimizing the average wavefront minimizes the CPU time, and optimizing the maximum wavefront reduces the memory requirements. In most cases, however, optimizing for one criterion will result in optimization for the other as well.

c. If you are using the Chaco optimization method, modify the optimization algorithm parameters, if necessary. In most cases, there will be no need to change these parameters.

Options for optimisation

(This menu item is not available when the Cuthill-McKee algorithm is used.)

- **Modify the tolerance factor** sets the tolerance parameter that the Chaco algorithm uses for optimizing the numbering.
- **Modify the weight on elements** controls the relative importance of variables located within sub-parts with respect to variables located at the interface between sub-parts. The default value reduces the length of the interfaces as much as possible.
- **Modify the weight on Schur variables** controls the relative importance of variables located at the interface between sub-parts with respect to variables within sub-parts. The default value reduces the length of the interfaces as much as possible.

4. Specify the cost evaluation parameters.

Options for cost evaluation

The cost evaluation is a qualitative evaluation of the relative acceleration a given mesh decomposition is likely to yield, based on an operations count for a single linear scalar variable. The comparison of the value obtained by not taking decomposition into account to the value obtained with decomposition is an order of magnitude representation of the estimated acceleration.

By default, ANSYS POLYDATA will evaluate and report the cost of the new mesh numbering, taking into account the decomposition into sub-parts. To report the cost evaluation based on the non-decomposed mesh, select **Do not take sub-domains decomposition into account**. To report the cost evaluation for the original mesh numbering, select **Do not take numbering optimisation into account**.

You can turn off the cost evaluation report by selecting the **Disable cost evaluation** menu item instead of the **Options for cost evaluation** menu item. The cost evaluation will still be performed, but the results will not be reported.

5. Specify the name of the original mesh file.

   Enter the filename of the initial mesh

6. Specify the name of the file where the optimized and/or decomposed mesh should be saved.

   Enter the filename of the new mesh

   You can overwrite the original mesh file by specifying the same name, but it may be safer to save the new mesh to a different file.

7. Check the settings listed at the top of the **Mesh decomposition and optimisation** menu to be sure that they are correct, and repeat the appropriate step(s) above to make any corrections.

8. Perform the decomposition and/or optimization.

   Run computation...

You can now read the newly saved mesh file into ANSYS POLYDATA (as described in *Mesh Files Created by POLYMESH or Converted by ANSYS POLYDATA* (p. 105)) and proceed with the problem definition.

7.8.4. Using Optimized or Decomposed Mesh

By default, ANSYS POLYFLOW will optimize and (if necessary) decompose the mesh before it begins the calculation. If you want ANSYS POLYFLOW to use a mesh that you have already optimized and/or decomposed (using either your mesh generator or ANSYS POLYDATA), you will need to start ANSYS POLYFLOW with the `-opt msh` command line option:

```
polyflow -opt msh < my.dat
```

or

```
polyflow -opt msh < my.dat > my.lst
```

When ANSYS POLYFLOW is started with this option, it will not perform any optimization or decomposition of the mesh. Instead, it will use the mesh file identified in the data file without making any changes to it.

7.8.5. Specifying the Number of Sub-Parts for Decomposition

If you want ANSYS POLYFLOW to perform the decomposition before it begins the calculation, but you want to specify the number of sub-parts yourself, you will need to start ANSYS POLYFLOW with the `-opt` *number_of_sub-parts* command line option (where *number_of_sub-parts* is the desired number of sub-parts, such as 8):

```
polyflow -opt 8 < my.dat
```

or

```
polyflow -opt 8 < my.dat > my.lst
```

See *About Domain Decomposition* (p. 146) for more information about sub-parts.

7.9. Results Interpolation Onto Another Mesh

Results interpolation onto another mesh, or mesh-to-mesh interpolation, allows transferring results obtained on one mesh to a second mesh that is unrelated to the first one except through the geometry. Usually, the second mesh will be more refined than the first one. There is no assumption that the nodes or element numbering correspond, however a reasonable geometric match is required. If the two geometries do not match, the error will be of the order of the geometrical difference. All calculated fields, including velocity, temperature, pressure and species, will be transferred from the first mesh to the second mesh.

Note that the procedure described below is reasonable as long as it is not applied frequently on a given case. For a recurrent and automatic transfer of results onto new meshes, you should consider using adaptive meshing tools as described in *Adaptive Meshing* (p. 366).

7.9.1. The CSV File

The interpolation of results from one mesh to the other makes use of a “neutral" results file in a format that is based on geometrical locations only. (Ordinary finite-element formats make use of element topological relationships.) This is called a CSV (comma separated variables) file in ANSYS POLYDATA; it is common for tabulated data and can also be read into spreadsheet programs such as Excel.

A CSV file contains a list of data points and a list of values. Fields are recognized by their (case-sensitive) names. When you save ANSYS POLYFLOW results in a CSV format, a data entry is written for each nodal location at which the quantity is defined.

7.9.2. Using Mesh-to-Mesh Interpolation

The procedure for interpolating results from one mesh to another is as follows:

1. Create the data file for the first mesh in ANSYS POLYDATA, and request that results be output to a CSV file. Select the **Enable CSV (Excel) output** menu item in the **Outputs** menu in ANSYS POLYDATA.

   Enable CSV (Excel) output

2. Calculate a solution on the first mesh. Upon completion of the calculation, ANSYS POLYFLOW will save the results to a CSV file.

3. Create the data file for the second mesh in ANSYS POLYDATA, and specify that the calculation should be started from the CSV file. In the **Numerical parameters** menu, select the **Start from an old CSV (Excel) file** menu item.

   Start from an old CSV (Excel) file

4. Calculate a solution on the second mesh. All variables currently defined will be interpolated and initialized on the basis of the CSV file before ANSYS POLYFLOW launches the calculation.

Important

Note that if you already have results from an ANSYS POLYFLOW simulation, but you did not request a CSV file, you can convert the ANSYS POLYFLOW results (`res`) file to a CSV file within ANSYS POLYDATA:

1. Select **Convert old results file**.

   Convert old results file
2. Select the results file to be converted.

   Enter the name of the res file
3. Request output to a CSV (Excel) file.

   Select outputs
4. Return to the **Convert old results file** menu and initiate the conversion.

   Convert
5. When prompted, modify or accept the data fields to be saved, and then modify or accept the name for the CSV file (`csv`, by default). ANSYS POLYDATA will launch ANSYS POLYFLOW to generate the CSV file.
6. Click **Return** in the ANSYS POLYFLOW window to return to ANSYS POLYDATA.

You can then follow the steps above to interpolate the results to a new mesh.

7.9.3. Using the CSV File to Initialize Solution Variables

A CSV file can also be used to initialize a field such as temperature or thickness, using experimental data, for example. Generate a template CSV file (following the procedure in *Using Mesh-to-Mesh Interpolation* (p. 150)), edit the values in your favorite spreadsheet, and save the file. Then follow the procedure outlined in *Using Mesh-to-Mesh Interpolation* (p. 150) to start the calculation from the CSV file.

There is no minimum or maximum number of points in a CSV file; the only requirement is that at least one point must be specified in order for the initialization to proceed. Note that if a single value is provided for the field, then all nodal values will be initialized to this value. If more than one value is provided, data will be interpolated between these values.

7.10. Combining Meshes with ANSYS POLYFUSE

7.10.1. Introduction

In some cases, you may need to combine several disconnected parts in order to simulate a given process. For example, blow molding and thermoforming simulations involve a parison and molds, and extruders and batch mixers that use the mesh superposition technique (see *Flows with Internal Moving Parts* (p. 457)) will also involve multiple parts. In such cases, you can create a separate mesh file for each part, combine them using ANSYS POLYFUSE, and then perform the simulation on the entire model.

In cases such as blow molding, for example, the number of elements defined in the mold does not significantly affect the computational time. You can therefore generate a very fine mesh for these solid parts in order to get an accurate definition of their shape. Unfortunately, writing a mesh with a large number of elements for a complex geometry may take a significant amount of time in your mesh generator. Each time you want to modify the parison shape or discretization, you will need to spend a lot of time generating the mesh for the mold as well. If, however, you create the mesh for the mold only once and save it to a separate file, you can then make modifications to the parison mesh without regenerating the mold mesh. Similarly, for an extruder modeled with the mesh superposition technique, you can generate the mesh for the screw elements once, and then modify the mesh for the barrel as often as you need.

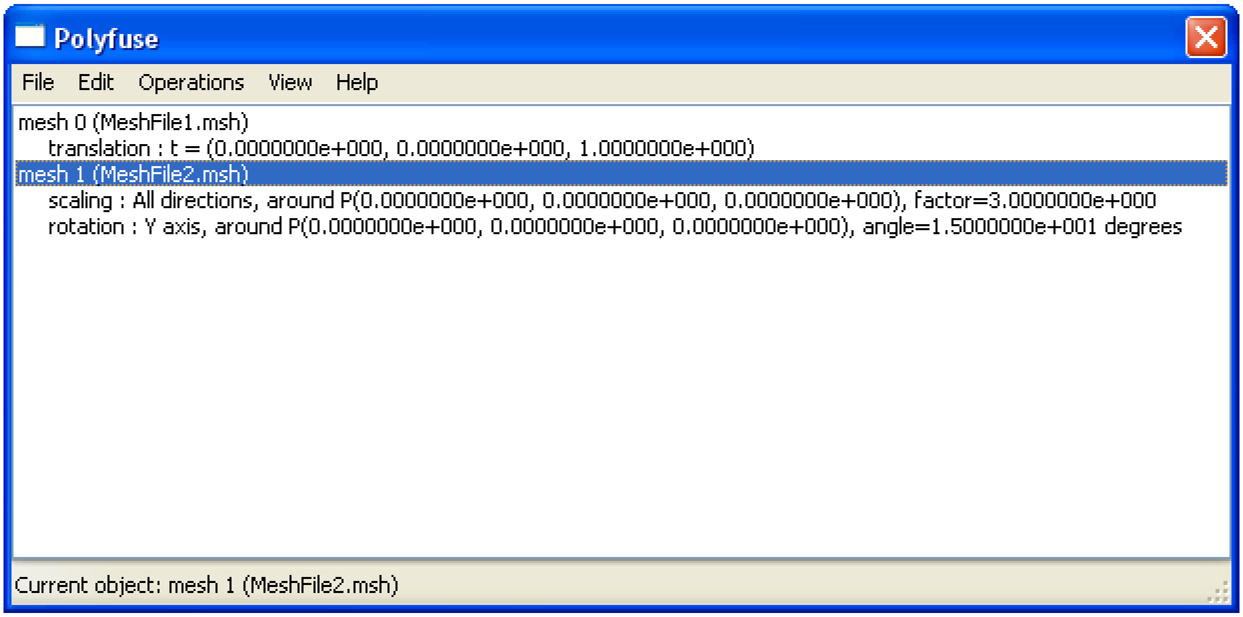
7.10.2. Starting ANSYS POLYFUSE

To start ANSYS POLYFUSE, select the **Combine mesh files** menu item in the top-level ANSYS POLYDATA menu. (You need not read the mesh files first; you will read them directly into ANSYS POLYFUSE.) The ANSYS POLYFUSE window (*Figure 7.4* (p. 152)) will open.

Combine mesh files

There are two other ways to launch ANSYS POLYFUSE: you can start ANSYS POLYFUSE from ANSYS POLYMAN, as described in *Starting the Programs* (p. 40); and you can start POLYFUSE from a POLYFLOW component system in ANSYS Workbench (via the context menu that is opened by right-clicking on the **Setup** cell), as described in the separate ANSYS POLYFLOW in ANSYS Workbench User's Guide.

Figure 7.4 The ANSYS POLYFUSE Window



```
mesh 0 (MeshFile1.msh)
    translation : t = (0.0000000e+000, 0.0000000e+000, 1.0000000e+000)
mesh 1 (MeshFile2.msh)
    scaling : All directions, around P(0.0000000e+000, 0.0000000e+000, 0.0000000e+000), factor=3.0000000e+000
    rotation : Y axis, around P(0.0000000e+000, 0.0000000e+000, 0.0000000e+000), angle=1.5000000e+001 degrees
```

Current object: mesh 1 (MeshFile2.msh)

There are three main components of the ANSYS POLYFUSE window:

- tool bar with menu items (**File**, **Edit**, etc.)
- list of existing objects
- **current object** field, which shows the object selected for manipulation or modification

Note that an object is either a mesh or a scaling, translation, or rotation operation applied to a mesh.

7.10.3. Steps for Combining Meshes

The basic steps for combining multiple meshes with ANSYS POLYFUSE are as follows:

1. Select the meshes to be combined.

   File → **Select Meshes**

2. Perform any manipulations on the meshes using the **Translate**, **Rotate**, and **Scaling** menu items and the items in the **Edit** submenu.

3. If you want, examine the mesh using the menu items in the **View** menu, and report information about the mesh using the **Info** menu item.

4. Save the combined meshes to a single file.

   File → **Write Mesh**

5. Exit from ANSYS POLYFUSE and return to ANSYS POLYDATA.

   File → **Exit**

Back in ANSYS POLYDATA, you can read the new mesh file and proceed with the problem definition.

7.10.4. Reading and Writing Files

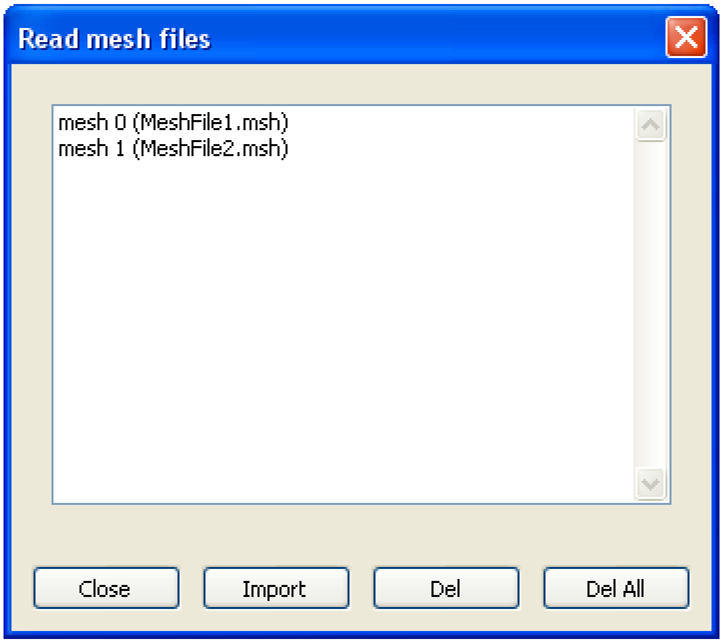
You can read meshes into ANSYS POLYFUSE, save the combined mesh, and save and reopen an ANSYS POLYFUSE session.

7.10.4.1. Selecting the Mesh Files

You can select the mesh files to be merged using the **Read mesh files** panel (*Figure 7.5* (p. 153)).

File → **Select Meshes**

Figure 7.5 The Read mesh files Panel


Read mesh files

mesh 0 (MeshFile1.msh)
mesh 1 (MeshFile2.msh)

To add a mesh to the list, click **Import** and specify the mesh file to be read. The file's name will be added to the list.

If you import the wrong mesh, you can select it in the list and click **Del** to remove it. Click **Del All** to remove all meshes from the list and start over.

7.10.4.2. Saving the Combined Mesh File

When you are ready to merge the files into a single mesh file, choose the **File/Write Mesh** menu item and specify the new file name.

File → **Write Mesh**

7.10.4.3. Reading and Writing an ANSYS POLYFUSE Session File

As described in *Reading and Writing ANSYS POLYDATA Session Files* (p. 108) for ANSYS POLYDATA, you can also read and write session files for ANSYS POLYFUSE.

- To save an ANSYS POLYFUSE session file, select the **File/Save** menu item and specify a name for the session file.

  File → **Save**

  The session file will contain information about which meshes have been read and which operators have been applied to them.

- To read the session file back into ANSYS POLYFUSE, select the **File/Open** menu item and specify its name.

  File → **Open**

  ANSYS POLYFUSE will load the session file, read the specified meshes, and apply the specified operators.

7.10.5. Translating, Rotating, and Scaling a Mesh

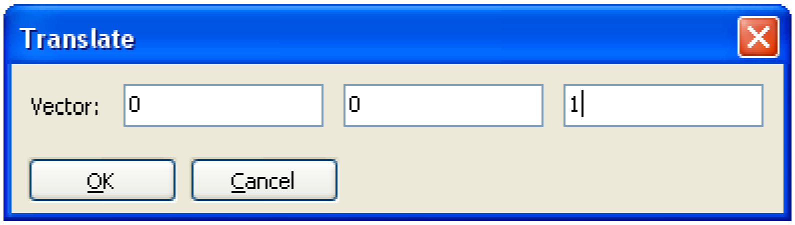
7.10.5.1. Translating a Mesh

To translate a mesh, follow these steps:

1. In the ANSYS POLYFUSE main window, select the mesh to be translated.
2. Select the **Translate** menu item.
3. In the resulting **Translate** panel (*Figure 7.6* (p. 155)), specify the translation **vector**: for 2D, $\mathbf{t} = (tx\,, ty)$, and for 3D, $\mathbf{t} = (tx\,, ty\,, tz)$. For each node of the selected mesh, the new coordinates will be calculated as $x_{new} = x_{old} + \mathbf{t}$.
4. Click **OK**.

The translation information will be added as an object below the selected mesh in the main window (see *Figure 7.4* (p. 152)). If you need to make changes to the translation vector, you can use the items in the **Edit** menu, as described later in this section.

Figure 7.6 The Translate Panel



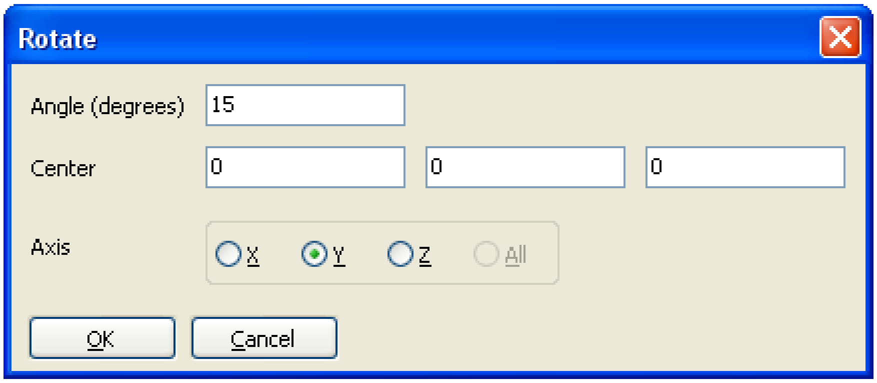
Vector: 0 0 1

7.10.5.2. Rotating a Mesh

To rotate a mesh, follow these steps:

1. In the ANSYS POLYFUSE main window, select the mesh to be rotated.
2. Select the **Rotate** menu item.

   Figure 7.7 The Rotate Panel

   Angle (degrees) 15

   Center 0 0 0

   Axis ○ X ● Y ○ Z ○ All

3. In the resulting **Rotate** panel (*Figure 7.7* (p. 155)), define the rotation:

   a. Specify the direction of the **axis** as parallel to the **X**, **Y**, or **Z** axis.

   b. Specify the origin (**center**) about which the rotation will be performed: for 2D, $\mathbf{p} = (px, py)$, and for 3D, $\mathbf{p} = (px, py, pz)$.

   c. Specify the **angle** of rotation (α) in degrees.

4. Click **OK**.

The rotation information will be added as an object below the selected mesh in the main window (see *Figure 7.4* (p. 152)). If you need to make changes, you can use the items in the **Edit** menu, as described later in this section.

For each node of the selected mesh, the new coordinates will be calculated as $\mathbf{x}_{new} = \mathbf{p} + \mathbf{R} \cdot (\mathbf{x}_{old} - \mathbf{p})$, where $\mathbf{R}$ is the rotation matrix.

If the rotation axis is parallel to the x axis,

$$\mathbf{R} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \quad (7–1)$$

If the rotation axis is parallel to the y axis,

$$\mathbf{R} = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix} \quad (7–2)$$

If the rotation axis is parallel to the z axis,

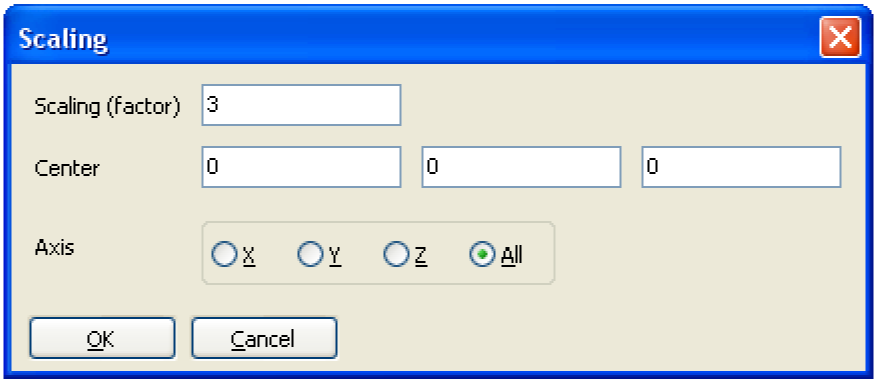
$$\mathbf{R} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (7–3)$$

7.10.5.3. Scaling a Mesh

To scale a mesh, follow these steps:

1. In the ANSYS POLYFUSE main window, select the mesh to be scaled.
2. Select the **Scaling** menu item.

Figure 7.8 The Scaling Panel



Scaling

Scaling (factor): 3

Center: 0 0 0

Axis: ○ X ○ Y ○ Z ● All

3. In the resulting **Scaling** panel (*Figure 7.8* (p. 156)), define the scaling:
   a. Specify the scaling direction (**axis**) to be in the **X**, **Y**, or **Z** direction, or in **all** directions.
   b. Specify the **center** point for the scaling: for 2D, $\mathbf{p} = (px, py)$, and for 3D, $\mathbf{p} = (px, py, pz)$.
   c. Specify the **scaling factor** (f).
4. Click **OK**.

The scaling information will be added as an object below the selected mesh in the main window (see *Figure 7.4* (p. 152)). If you need to make changes, you can use the items in the **Edit** menu, as described later in this section.

For each node of the selected mesh, the new coordinates will be calculated as $\mathbf{x}_{new} = \mathbf{p} + \mathbf{A} \cdot (\mathbf{x}_{old} - \mathbf{p})$, where $\mathbf{A}$ is the scaling matrix.

If the scaling is only in the x direction,

$$\mathbf{A} = \begin{pmatrix} f & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \textbf{(7–4)}$$

If the scaling is only in the y direction,

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \textbf{(7–5)}$$

If the scaling is only in the z direction,

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & f \end{pmatrix} \qquad \textbf{(7–6)}$$

If the scaling is in all directions,

$$\mathbf{A} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & f \end{pmatrix} \qquad \textbf{(7–7)}$$

7.10.5.4. Manipulating Translation, Rotation, and Scaling Operations

Once you have defined a translation, rotation, or scaling operation, you can modify it, change the order of operations, delete it, or copy it to another mesh. The procedure is as follows:

1. Select the operation object in the main window.

   Important

   The **Edit** menu items apply only to operation objects; they will have no effect if you select a mesh object.

2. Select the desired option in the **Edit** menu:

- **Move Up** and **Move down** are used to modify the order in which the operations on a particular mesh are performed. For example, a translation operation followed by a rotation operation will yield different results than the same rotation operation followed by the same translation operation.
- **Modify** opens the **Translate**, **Rotate**, or **Scaling** panel for the selected operation so that you can make changes to it. When you click **OK**, the new definition will replace the old one. If you click **Cancel**, your changes will be ignored and the original definition for the operation will be retained.
- **Delete** deletes the selected operation.
- **Copy** and **Paste** are used to copy an operation to another mesh or to repeat it for the same mesh. After you have chosen the operation object and selected the **Copy** menu item, select the mesh for which you want to apply the copied operation, and select the **Paste** menu item. The operation will be placed at the end of the list for the mesh.

7.10.6. Viewing the Mesh

To see what your meshes look like, first select the **Update Coord.** menu item to ensure that the mesh display will include the final results of all operations performed on the meshes.

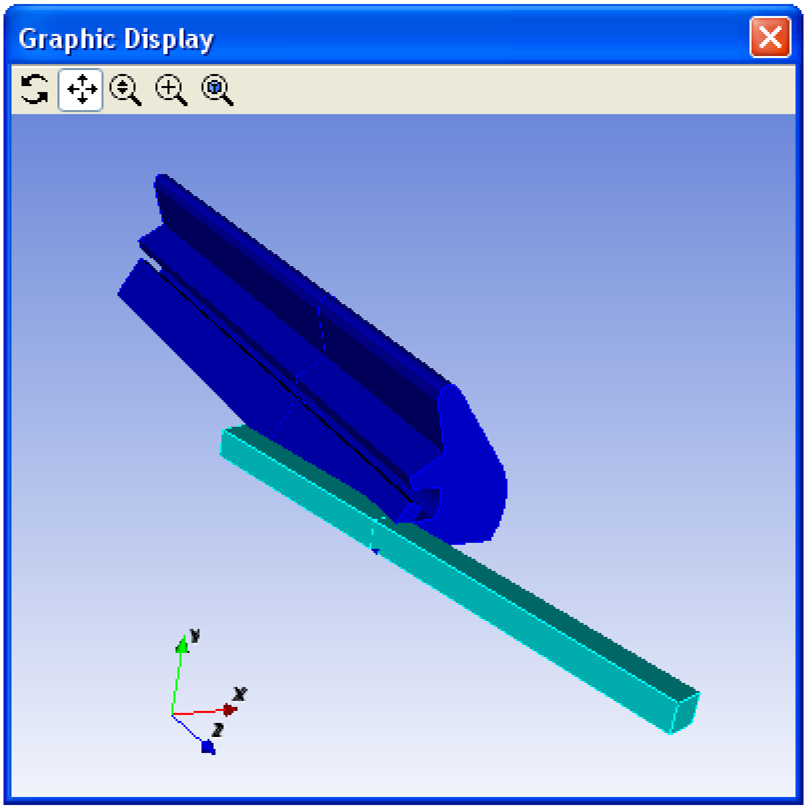
View → **Update Coord.**

Then select the **Open View** menu item.

View → **Open View**

ANSYS POLYFUSE will open the **Graphic Display** window (*Figure 7.9* (p. 159)), which contains the display of the meshes. The **Graphic Display** window incorporates a graphics toolbar that can be used to manipulate the view, in the same manner as the POLYDATA graphics toolbar described in *Graphics Toolbar* (p. 26). You can also manipulate the display using the shortcut keys described for the POLYDATA **Graphic Display** window (see *Table 7.1: Graphics Display Window Shortcut Keys* (p. 160)).

Figure 7.9 The Graphic Display Window



To close the **Graphic Display** window, select the **Close View** menu item.

View → **Close View**

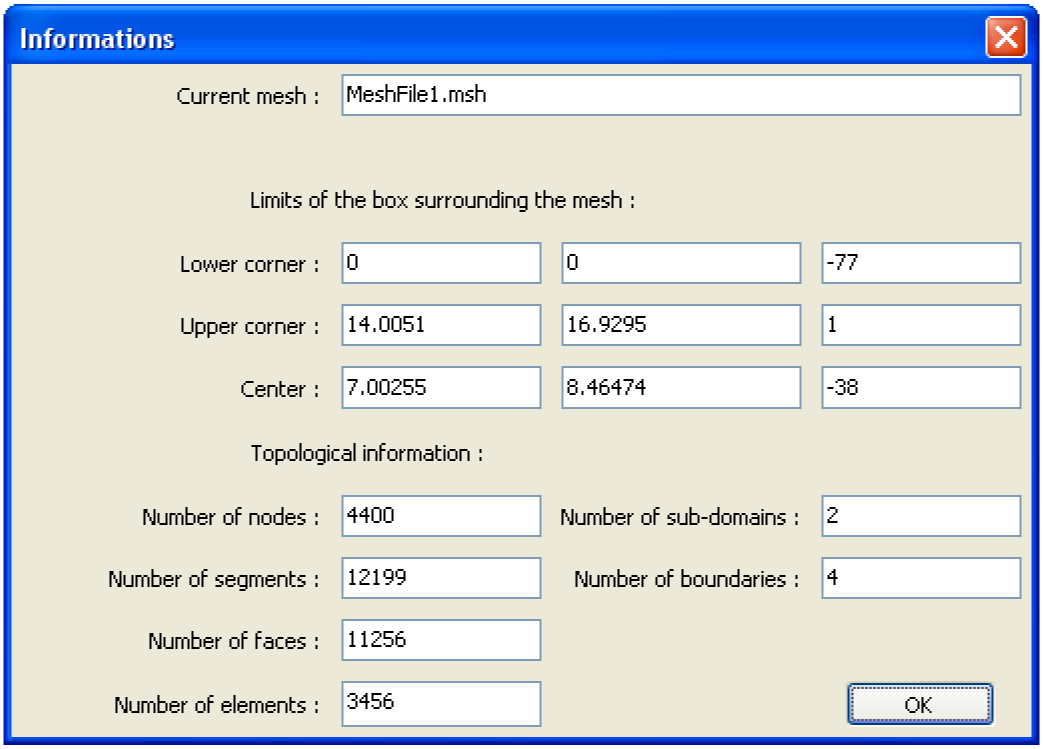
If you make changes to the mesh operations while the mesh is displayed, select the **Update Coord.** menu item again to update the display.

View → **Update Coord.**

7.10.7. Reporting Information about the Mesh

You can check the number of nodes, segments, faces, elements, subdomains, and boundaries in the selected mesh (shown in the **current object** field in the main window) using the **Info** menu item, which opens the **Informations** panel (*Figure 7.10* (p. 160)).

Figure 7.10 The Informations Panel



Informations

Current mesh : MeshFile1.msh

Limits of the box surrounding the mesh :

Lower corner :	0	0	-77
Upper corner :	14.0051	16.9295	1
Center :	7.00255	8.46474	-38

Topological information :

Number of nodes :	4400	Number of sub-domains :	2
Number of segments :	12199	Number of boundaries :	4
Number of faces :	11256		
Number of elements :	3456		

The limits of the box surrounding the mesh are also reported. The box is calculated based on the current coordinates of the mesh nodes (after the **View/Update Coord.** menu item has been selected). This information helps you determine if the mesh is correctly positioned (e.g., if translation operations have been properly performed).

7.11. Examining the Mesh in ANSYS POLYDATA

Once you have read the mesh into ANSYS POLYDATA, it will be displayed in the **Graphics Display** window (as described in *Graphics Display Window* (p. 25)). There are several ways in which you can modify the display of the mesh, including translating, rotating, and zooming the view, specifying which portions of the mesh you want to include in the display, changing the colors in the display, requesting display of the outline or grid lines or a filled-in display, and saving the display to a PostScript file. See the sections that follow for complete details.

Most of the display modifications can be performed using shortcut keys, which are summarized in *Table 7.1: Graphics Display Window Shortcut Keys* (p. 160). Click the left mouse button in the **Graphics Display** window and type the appropriate letter or key combination.

Table 7.1 Graphics Display Window Shortcut Keys

	Action	Shortcut
Appearance	Display the geometry in shaded mode.	**s**
	Display the geometry as a wireframe. This can be useful when one mesh is hidden by another one.	**w**
Axes	Remove the axes from the display.	**a**
	Add axes to the display.	**A**
Background Color	Set the background color to light blue.	**b**

	Action	Shortcut
	Set the background color to black.	**B**
	Set the background color to white.	**W**
	Set the background color to grey.	**G**
Size	Size the view to fit in the window.	**r** or **R**
	Double the x, y, or z dimension.	**X**, **Y**, or **Z**, respectively
	Reduce the x, y, or z dimension by half.	**Ctrl-x**, **Ctrl-y**, or **Ctrl-z**, respectively
View	Change the view to be perpendicular to the yz plane from the positive x coordinate.	**x**
	Change the view to be perpendicular to the xz plane from the positive y coordinate.	**y**
	Change the view to be perpendicular to the xy plane from the positive z coordinate.	**z**
	Display an orthographic view.	**o**
	Display a perspective view.	**p**

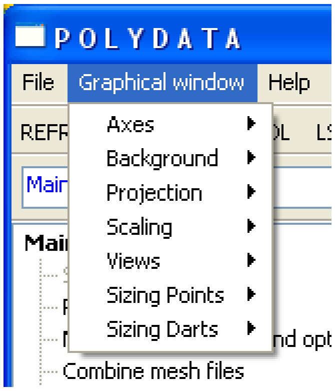
7.11.1. Translating, Rotating, and Zooming with the Mouse

You can translate, rotate, and zoom the view by clicking a button in the graphics toolbar and then clicking and dragging the mouse in the **Graphics Display** window. See *Graphics Toolbar* (p. 26) for details.

7.11.2. Changing the View Axis

The default view is perpendicular to the xy plane from a positive z coordinate. To change to a view perpendicular to the yz plane from a positive x coordinate, click the left mouse button in the **Graphics Display** window and type x. To change to a view perpendicular to the xz plane from a positive y coordinate, type y. To return to the default view, type z. You can also select the view axis in the **Views** menu in the **Graphical window menu** (*Figure 7.11* (p. 161)).

Figure 7.11 The Graphical window Menu



7.11.3. Scaling and Resizing the View

When you zoom in and out (as described in *Graphics Toolbar* (p. 26)), the view is scaled by the same amount in all coordinate directions. It is also possible to shrink or expand the view in a specified coordinate direction. Click the left mouse button in the **Graphics Display** window and perform the following:

- To double the x dimension, type X; to reduce it by half, type **Ctrl**-**x**.
- To double the y dimension, type Y; to reduce it by half, type **Ctrl**-**y**.
- To double the z dimension, type Z; to reduce it by half, type **Ctrl**-**z**.

You can also perform this scaling in the **Scaling** menu in the **Graphical Window** menu. To reset all scaling factors to 1 (the default value) and fit the display in the window, click the left mouse button in the **Graphics Display** window and type `r`. To fit the display in the window but keep the current scaling factors, type R.

7.11.4. Displaying Perspective and Orthographic Views

You may choose to display either a perspective view (the default) or an orthographic view. To display a perspective view, click the left mouse button in the **Graphics Display** window and type p. To display an orthographic view, type o. You can also select these views in the **Projection** menu in the **Graphical Window** menu.

7.11.5. Modifying the Background Color for the Display

By default, the background color for the mesh display in the **Graphics Display** window is light blue. To change it to black, white, or gray, click the left mouse button in the window and type B, W, or G, respectively. You can then type b to return to light blue. You can also select the background color in the **Background** menu in the **Graphical Window** menu.

7.11.6. Including the Coordinate Axes in the Display

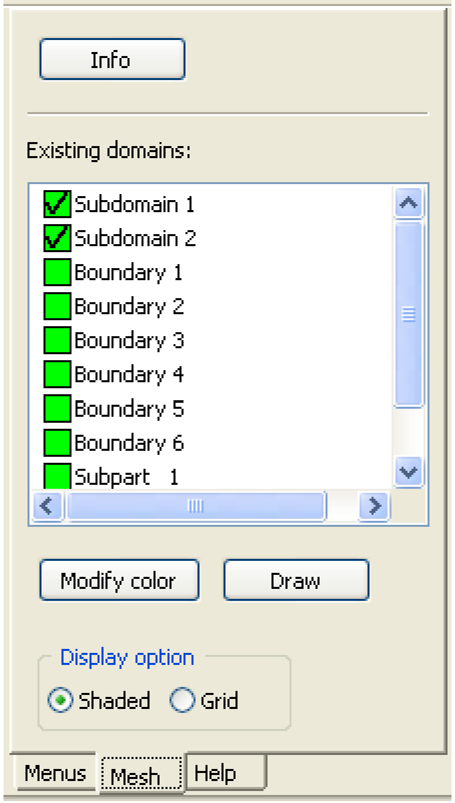
By default, the coordinate axes are not included in the **Graphics Display** window. To add them, you can click the left mouse button in the **Graphics Display** window and type A. To remove the axes, type a. You can also add and remove the axes in the **Axes** menu in the **Graphical Window** menu.

7.11.7. Selecting Portions of the Mesh for Display

By default, the entire mesh is displayed in the **Graphics Display** window. If you want to select certain portions of the mesh for display, follow the steps below:

1. Select the **MESH** tab at the bottom of the ANSYS POLYDATA window. (*Figure 7.12* (p. 163)), the **Existing Domains** list will show the portions of the mesh that are currently selected for display.

Figure 7.12 The Mesh Topology Panel



2. To delete one of the items selected for display, deselect it in the **Existing Domains** list.
3. To add a new item (boundary, subdomain, etc.) to the display, simply select it in the **Existing domains** list.
4. Click the **Draw** button to update the **Graphics Display** window.

7.11.8. Modifying the Color of the Mesh Display

To modify the color of the mesh display, follow the steps below:

1. Select the **MESH** tab at the bottom of the ANSYS POLYDATA window to open the **Mesh Topology** panel (*Figure 7.12* (p. 163)).
2. Select the item for which you want to change the color in the **Existing Domains** list.
3. Click the **Modify Color** field and select the desired color.
4. Click the **Draw** button to update the **Graphics Display** window.

7.11.9. Specifying Outline, Wireframe, and Shaded Displays

There are several ways to display the mesh: the outline, the grid lines within the mesh (wireframe), and a shaded (filled-in) view of the mesh. A shaded view is the default. To change to a wireframe view, you can click the left mouse button in the **Graphics Display** window and type `w`. To return to a shaded view, type `s`.

You can also specify a shaded or wireframe view (as well as an outline view) using the **Mesh Topology** panel. The procedure is as follows:

1. Select the **MESH** tab at the bottom of the ANSYS POLYDATA window to open the **Mesh Topology** panel (*Figure 7.12* (p. 163)).
2. Follow the instructions for the type of display you want:
   - To display the grid outline, deselect all items from the **Existing Domains** list and select the **Skeleton** item. See *Selecting Portions of the Mesh for Display* (p. 162) for details.
   - To display the wireframe grid lines within the mesh (or the portions of the mesh selected for display), select the **Grid** option at the bottom of the panel.
   - To display a filled-in view of the mesh (or the portions of the mesh selected for display), select the **Shaded** option at the bottom of the panel.
3. Click the **Draw** button to update the **Graphics Display** window.

7.11.10. Reporting Information about the Mesh

You can check the number of nodes, segments, faces, elements, subdomains, boundaries, and PMeshes in the mesh using the **Info** button. Information about the mesh decomposition is also available. The limits of the box surrounding the mesh are also reported. The box is calculated based on the current coordinates of the mesh nodes.

7.12. Generating a Sliceable Free Jet Mesh

The most commonly used remeshing techniques available in POLYFLOW require that the free jet domain is "sliceable". To understand what is meant by sliceable, imagine that a free jet domain is divided into a series of parallel slices that are perpendicular to the longitudinal axis: the first slice is the starting section of the remeshing domain, the last slice is the ending section of the remeshing domain, and the slices in between exist where there are nodes. The mesh is considered sliceable if the topology (i.e., the number of nodes, segments, and faces) is identical in each slice.

A free jet domain that is *not* sliceable can be generated in various ways: for example, if you use the CutCell method in ANSYS Meshing (in such circumstances, a non-conformal mesh is generated throughout the full mesh, and thus also in the free jet), or if you generate a mesh containing tetrahedral or pyramids in the free jet.

For the times when you want to use a remeshing technique that requires a sliceable mesh, POLYDATA provides a tool that allows you to replace the mesh of the free jet region with a new sliceable mesh; as a result, you do not need to recreate a sliceable mesh in ANSYS Meshing (a task that can be quite difficult). This tool can also be used to modify aspects of an existing sliceable free jet mesh, such as the level of refinement or the length of the free jet. Such a modification can be performed quickly and easily without having to use ANSYS DesignModeler or ANSYS Meshing.

> **Note**
>
> For complex simulations (e.g., inverse prediction) in which the adaptive and/or constant sections are not sliceable, POLYDATA has remeshing techniques that do not require the domain to be sliceable. Such techniques are specified using the **Adaptive section for prediction (no slice)** and the **Constant section for prediction (no slice)** in the **Remeshing technique** menu (see *Inverse Extrusion and Die Design* (p. 331) for details).

To convert a mesh into a sliceable mesh or to modify an existing sliceable mesh, perform the following steps:

1. Read the mesh.
2. Select the **Generate a sliceable free jet** menu item in the main menu of POLYDATA.

   Generate a sliceable free jet

   The **Generate a sliceable free jet** menu will open.
3. Specify the domain of the free jet.

   Enter the free jet domain

   When defining the domain, note that the following conditions must be met in order for the free jet topology to be acceptable: each sub-domain of the free jet must touch both the starting and ending sections of the free jet; furthermore, such sub-domains cannot lie along boundaries of the extrusion die.
4. Specify which boundaries make up the starting and ending sections of the free jet domain.

   Enter the free jet boundary conditions

   The starting section can each be composed of several boundaries, but all of the boundaries must lie on a single plane that has a normal aligned with a coordinate axis (X, Y, or Z). Each boundary of the starting section must have a corresponding boundary in the ending section. The ending section must be planar and parallel to the starting section. The resulting mesh will be "extruded" from the starting section in the normal direction to the ending section.

   > **Note**
   >
   > Several starting boundaries can be extruded to the same boundary in the ending section. In fact, the ending section is usually a single boundary.

   When defining the starting and ending sections, note that the following conditions must be met in order for the free jet topology to be acceptable: each boundary of the free jet that is not part of the starting and ending sections must connect to both the starting and ending sections of the free jet.
5. Specify the length of the free jet.

   Modify length of the free jet

   Start with a reasonable estimate of the length. You can always use this menu item in successive simulations to fine-tune the length: you can generate a longer extrudate if the resulting velocity profile is not fully uniform in the ending section, or a shorter extrudate if you do not observe any additional deformation of the free jet beyond a given distance from the die lip.
6. You can change the method by which the mesh is generated, as displayed in the text at the top of the **Generate a sliceable free jet** menu, by repeatedly clicking the **Modify method of generation** menu item.

   Modify method of generation

   The selected method determines how the thickness of the elements evolves along the extrusion direction, using parameters that are defined in later steps. Three methods are available:

- uniform

  For this method, the thickness of every element in the direction of extrusion will be set to the starting element thickness.

- linear

  For this method, the thickness of the elements will evolve linearly in the direction of extrusion from the starting element thickness, according to the following equation:

  $$t_l=t_i(1+(G-1)(l-1)) \tag{7–8}$$

  where t_l is the thickness of the layer number l, t_i is the starting element thickness, and G is the growth factor.

- geometric

  For this method, the thickness of the elements will evolve geometrically in the direction of extrusion from the starting element thickness, according to the following equation:

  $$t_l=t_{\mathrm{i}}\left(G^{(l-1)}\right) \tag{7–9}$$

  where t_l is the thickness of the layer number l, t_i is the starting element thickness, and G is the growth factor.

7. Specify the thickness of the first layer of elements in the free jet (adjacent to the starting section).

   Modify starting element thickness

8. If you selected the linear or geometric method of generation in a previous step, specify the growth factor used to determine how the thickness of the elements evolves along the extrusion direction.

   Modify growth factor

9. Specify the file name of the new mesh that will be generated by the solver.

   Modify new mesh file name

   By default, the file name of the new mesh is `newfrj.msh`.

10. Save your settings and generate the sliceable mesh.

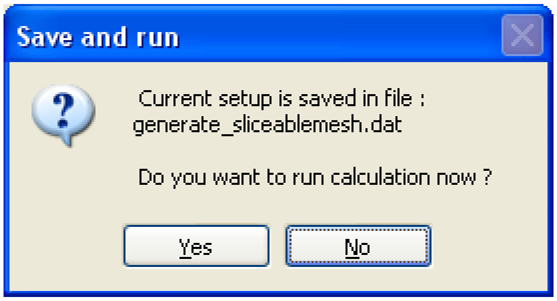
    If you launched POLYDATA from POLYMAN or from ANSYS Workbench, click the **Save and exit** menu item and then run the solver.

    Save and exit

    If you are using the stand-alone version of POLYDATA, click the **Generate and exit** menu item, which will open the **Save and run** dialog box (see *Figure 7.13* (p. 167)).

    Generate and exit

Figure 7.13 The Save and run Dialog Box



Current setup is saved in file :
generate_sliceablemesh.dat

Do you want to run calculation now ?

Click **Yes** to run POLYFLOW immediately. In any case, a data file (named `generate_sliceablemesh.dat`) will be generated for further use.

In the listing file produced by the POLYFLOW solver, you can follow the generation of the new mesh:

```
**************************************
*                                    *
* Generation of a sliceable free jet *
*                                    *
**************************************

Domain to remesh : free jet support.


Starting sections :
 - (freejet*die).

Ending sections :
 - (freejet*outlet).

Geometric method of generation

Free jet length :  0.5000000E+02
Element thickness . :  0.1500000E+01 (in first layer)
Growth factor  :  0.1200000E+01

Nb. of layers  :     12
Generation in Z- direction

Saving subdomains :
  - die
  - freejet

Saving boundaries :
  - walls
  - inlet
  - outlet
  - part_solid_1

*** WARNING *** from [regenr]
   Distorted element has been detected.
   Element    6080 of subdomain    1
   The check of the element connectivity is skipped,
   but the conversion continues without guarantee of results.
   Warning while creating topology (part I) of the new mesh !!


Saving new mesh in file : newfrj.msh
```

Note that in the previous example, a warning is displayed: a distorted element has been found in the `subdomain 1` (corresponding in this example to the domain named `die`). This illustrates the fact that the elements that are outside of the free jet domain are not altered.

7.13. Converting a Shell Mesh and Results

At the end of a shell blowing / thermoforming process simulation, ANSYS POLYFLOW calculates the final thickness (and possibly temperature) distribution(s) through the blown / thermoformed product. It may also be useful to examine how the product cools and the deformations induced by such cooling. To do so, it is necessary to convert the shell mesh and result files into one of the following:

- a full 3D mesh and corresponding result file.
- a format that is suitable for a subsequent analysis with another software package. Currently, ANSYS POLYFLOW can convert shell mesh and result files into a file that can be used as part of an LS-DYNA simulation.

Within the shell approximation for a multilayer system, the layer ordering is not relevant. Therefore, for such a multilayer system the thickness of a 3D mesh created from a shell mesh will be equal to the sum of all individual thicknesses.

The steps that follow describe how to convert shell meshes and result files.

1. For time-dependent or evolution problems, make sure that output files were saved at all of the time / evolution steps, intervals, or time values that are of interest to you. See *Output for Time-Dependent and Evolution Calculations* (p. 126) for details.
2. Read the mesh file. If you did not use adaptive meshing, read the initial mesh file. If you used adaptive meshing for the shell blowing process, the last `adapt.msh` file generated by ANSYS POLYFLOW (or a previous one, if relevant) should be read; note that in such a case, the files named `adapt.msh` and `rst.msh` are the same.
3. Click **Convert shell mesh and results** in the main menu of ANSYS POLYDATA to open the **Convert shell mesh and results** menu.

   Convert shell mesh and results

   Note that the format of the output files is not specified by default in the **Convert shell mesh and results** menu.
4. Specify the name of the ANSYS POLYFLOW result file you would like to convert.

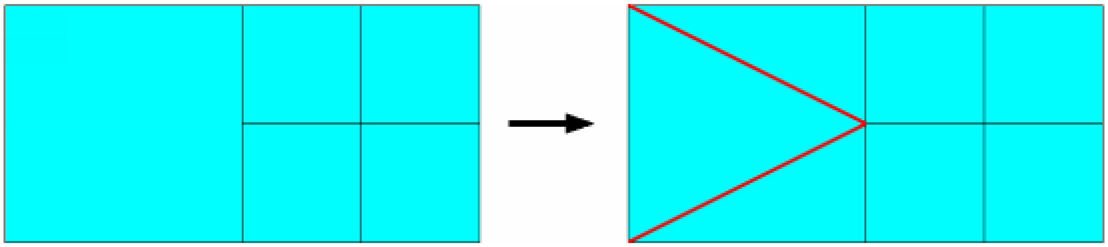
   Enter the input result file name

   Note that the result file should correspond with the mesh you read in step 2. Typically, you will specify the last `.res` file generated by ANSYS POLYFLOW during the simulation of the shell blowing process. If you read a mesh file at a previous time step, however, you should specify the result file that was saved at that time step.
5. Specify the domain to be extended into 3D.

   Enter the shell domain

   You have to select only the shell domain of your previous simulation. Be careful that you do not select the domains corresponding to the mold(s).
6. The next option concerns an optional refinement/unrefinement. By default, if adaptive meshing has been used in the previous simulation, a “conformalization" step will take place: elements adjacent to subdivided elements will be replaced in order to get a final mesh that is conformal (see *Figure 7.14* (p. 169)):

Figure 7.14 Example of Conformalization



However, it is also possible to restrain the levels of subdivision by specifying the minimum and maximum levels of refinement. To do so, select the **Activate refinement/unrefinement** menu item.

Activate refinement/unrefinement

When prompted, the **Modify min and max levels of refinement** item becomes available. Select this item:

Modify min and max levels of refinement

Then enter the desired range of refinement. All elements that have a level of refinement below the minimum will be refined, while those having a level higher than the maximum will be unrefined. If the minimum is not identical to the maximum level, a step of conformalization will be necessary and added automatically.

7. If you want to generate 3D mesh and result files, perform the following steps. Note that you can generate both 3D mesh and result files and an LS-DYNA output file at the same time.

    a. Click the **=> Enable 3D mesh and result files generation** menu item.

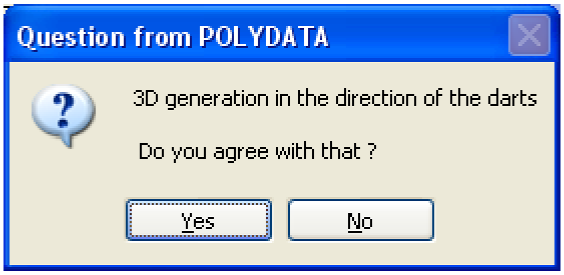
    => Enable 3D mesh and result files generation

    b. Specify the mesh and result file names, by selecting the **Enter the output mesh file name** and the **Enter the output result file name** menu items, respectively. By default, the mesh and result files will be named `output.msh` and `output.res`, respectively. With those files, it will be possible to define the simulation of cooling and thermoelastic deformations in another ANSYS POLYDATA session.

    Enter the output mesh file name

    Enter the output result file name

    c. Since it is possible to generate a 3D mesh by expansion of one side or the other side of the shell surface, you must specify the direction of generation. If you select the **Specify direction of 3D generation** menu item, the shell will be shown in the **Graphics Display** window with darts indicating the normal direction to the faces. By default, the mesh will be generated in that direction. A message appears also on your screen:



Question from POLYDATA

3D generation in the direction of the darts

Do you agree with that ?

By selecting **Yes** or **No**, you can accept or change the direction of generation. Note that your selection appears in the comments of the menu:

3D generation in the direction of the darts

or

3D generation in the opposite direction to the darts

d. If the simulation contains plane(s) of symmetry, it is necessary to define them. To do so, select the **Specify planes of symmetry** menu item.

☰ **Specify planes of symmetry**

A new menu will appear. By default, no constraints are applied on the directions of generation staying on boundaries of the shell domain. However, if planes of symmetry exist, the direction of generation must stay in such planes. To define such constraints, select each boundary B corresponding to a plane, then in the new menu **Specify plane of symmetry along boundary B**, switch to the **Plane of symmetry** option and enter the coefficients of the normal direction to the plane of symmetry.

e. Before running the conversion, you can check / modify some parameters used during the conversion. To do so, click the **Advanced Options** menu item.

☰ **Advanced Options**

A new menu will appear that presents a set of numerical parameters. The most important parameter is the **number of layers of elements**. Three layers of elements are generated by default. By increasing the number of layers, you can better capture any thermal boundary layer. The distribution of elements in the thickness follows a Chebichef law.

Other parameters are needed by the iterative scheme used to generate a “good" 3D mesh. Indeed, the generated 3D elements may be distorted: too bended, too stretched, nonconvex, or with negative Jacobian, depending on the local curvature of the shell and the local thickness attached to that element. That is why you have to specify minimum angle, maximum angle, maximum bend and maximum skew. While all 3D elements do not respect those criteria, new small local deformations of the 3D mesh occurs to render the mesh well shaped. Note that the more stringent the criteria are, the more iterations are needed (and maybe without finding a solution that respects those criteria). As a side effect of this procedure, the boundaries of the 3D mesh that are adjacent to the shell surface will probably not remain perpendicular to the shell surface.

In order to facilitate the generation of well shaped 3D elements, all quadrilateral faces of the shell surface that are too bended or that have a high internal angle will be divided into two triangles. The parameters that fix the limit are **maximum bend for faces** and **maximum angle for faces**, respectively.

For any 3D element, if its starting face does not respect the criteria mentioned previously, then the criteria applied on the element will be based on the limits of the starting face multiplied by (1+tolerance) or (1-tolerance). For example, if a given starting face has a minimum angle between two segments equal to 10° and the minimum angle specified is 20°, then the minimum angle that will effectively be applied for the criteria is $10*(1\text{-tolerance})^\circ$. Note that the small deformations applied on the 3D mesh are based on the local reorientation of the direction of generation. The starting shell surface is never deformed.

Another parameter that affects the speed of convergence of the iterative scheme is the mode of update of the local normals to the shell surface. The **delayed mode** seems to be faster than the **immediate mode**. The default value (**delayed mode**) should not be modified.

For debugging purpose, it is possible to write in the listing file more information by changing the verbosity (from minimum to maximum).

To accept the values of the numerical parameters, click the **Upper level menu** menu item.

Upper level menu

You will return to the **Convert shell mesh and results** menu.

8. If you want to generate an LS-DYNA output file, perform the following steps. Note that you can generate both an LS-DYNA output file and 3D mesh and result files at the same time.

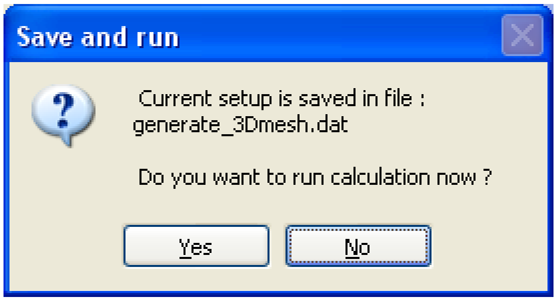
   a. Click the **=> Enable LS-DYNA output generation** menu item.

      =>Enable LS-DYNA output generation

   b. Specify the name for the LS-DYNA output file, by clicking the **Enter the output LS-DYNA file name** menu item and entering the name in the dialog box that opens. By default, the output file will be named `output.dyn`. You will use this file to define a simulation in LS-DYNA.

      Enter the output LS-DYNA file name

9. Click the **Convert and exit** or the **Save and exit** menu item to run an ANSYS POLYFLOW simulation that generates the 3D mesh and result files and/or LS-DYNA output file. The following message will appear to verify that you want to run the calculation:



Current setup is saved in file :
generate_3Dmesh.dat

Do you want to run calculation now ?

By clicking **Yes**, you will run ANSYS POLYFLOW immediately. In any case, a data file is generated (`generate_3Dmesh.dat`) for further use.

A listing file will be produced by ANSYS POLYFLOW, which provides information about the steps of the convergence (e.g., the number of distorted 3D elements decreasing with the number of iterations, information about the mesh conformalization, interpolation of thickness fields, etc.). The following is an example of the listing file produced when generating 3D mesh and result files.

```
 ****************************
 *                          *
 * Shell to 3D conversion   *
 *                          *
 ****************************

 Shell ...................... : shell support
 Thickness field(s) ......... : THICKNESS
 Isothermal shell
 3D generation along the normal to the faces
 Number of layers of elements :   3
 Maximum number of iterations : 500
 Update of normals .......... : delayed mode
 Information printed ........ : min.
 Minimum interior angle ..... : 0.2000000E+02
 Maximum interior angle ..... : 0.1600000E+03
 Maximum bend ............... : 0.8000000E+00
 Maximum skew ............... : 0.1000000E+02
 Maximum angle for faces .... : 0.1600000E+03
 Maximum bend for faces ..... : 0.1000000E+00
 Tolerance .................. : 0.1000000E+00

 + Planes of Symmetry

 Plane (S1*B3).
 of normal direction : 0.1000000E+01 0.0000000E+00 0.0000000E+00

 Iteration 1, # distorted elements = 795 / 18536
 Iteration 2, # distorted elements = 552 / 18536
 Iteration 3, # distorted elements = 565 / 18536
 Iteration 4, # distorted elements = 511 / 18536
 Iteration 5, # distorted elements = 485 / 18536
 ...
 Iteration 251, # distorted elements = 4 / 18536
 Iteration 252, # distorted elements = 3 / 18536
 Iteration 253, # distorted elements = 2 / 18536
 Iteration 254, # distorted elements = 1 / 18536
 Iteration 255, # distorted elements = 0 / 18536

 Quality of 3D elements : final check

>>  Number of distorted elements = 0 / 18536
```

Note

The divisor displayed for the number of distorted elements for each iteration (`18536` in the previous example) represents the number of starting faces (after all possible faces have been divided into triangles).

Chapter 8: Boundary Conditions

This chapter provides information about flow boundary conditions, as well as information about using a rotating reference frame and about specifying conditions on boundaries or within the domain using sub-models. See *Heat Transfer* (p. 283) for information about thermal boundary conditions.

Information in this chapter is presented in the following sections:

8.1. Overview of Boundary Conditions

Solving a problem in ANSYS POLYFLOW requires that you prescribe information about the flow along the boundary of the computational domain for each sub-task. Solving a time-dependent flow also requires that you prescribe initial conditions within the domain, as discussed in *Time-Dependent Flows* (p. 539). In addition, for viscoelastic flows, boundary conditions for the viscoelastic part of the constitutive equation are required at inlets. These represent the contribution to the constitutive equation from the past history of material particles entering the flow domain.

The domain of the sub-task is surrounded by closed curves (in 2D) or surfaces (in 3D) consisting of boundary sets and intersections with adjacent subdomains that are not included in the sub-task. Each of these curves or surfaces is referred to as a boundary section.

8.1.1. Available Boundary Conditions

The default condition defined by ANSYS POLYDATA on each section of the boundary is a zero wall velocity (zero normal velocity and zero tangential velocity), except for the axis of symmetry in an axisymmetric model. For an axisymmetric model, ANSYS POLYDATA automatically prescribes a symmetry condition along the line $r = 0$. You can keep the default condition where appropriate, and specify new conditions for the rest of the boundary sections. The boundary conditions available in ANSYS POLYDATA are as follows:

- zero wall velocity (vn=vs=0)
- slip conditions
- porous wall
- axis of symmetry
- inflow
- outflow
- free surface
- normal and tangential velocities imposed
- normal and tangential forces imposed
- normal velocity and tangential force imposed
- normal force and tangential velocity imposed
- global force imposed
- cylindrical velocities imposed
- velocity profile from CSV file
- interface
- interface with porous media
- interface with elastic solid
- inlet of periodic condition
- outlet of periodic condition
- source of connected condition
- target of connected condition

It is also possible to specify conditions on a boundary section using a submodel, as described in *Specifying Conditions Using Sub-Models* (p. 192). With sub-models, you can also specify conditions on an internal region of the domain.

8.1.2. Setting Boundary Conditions

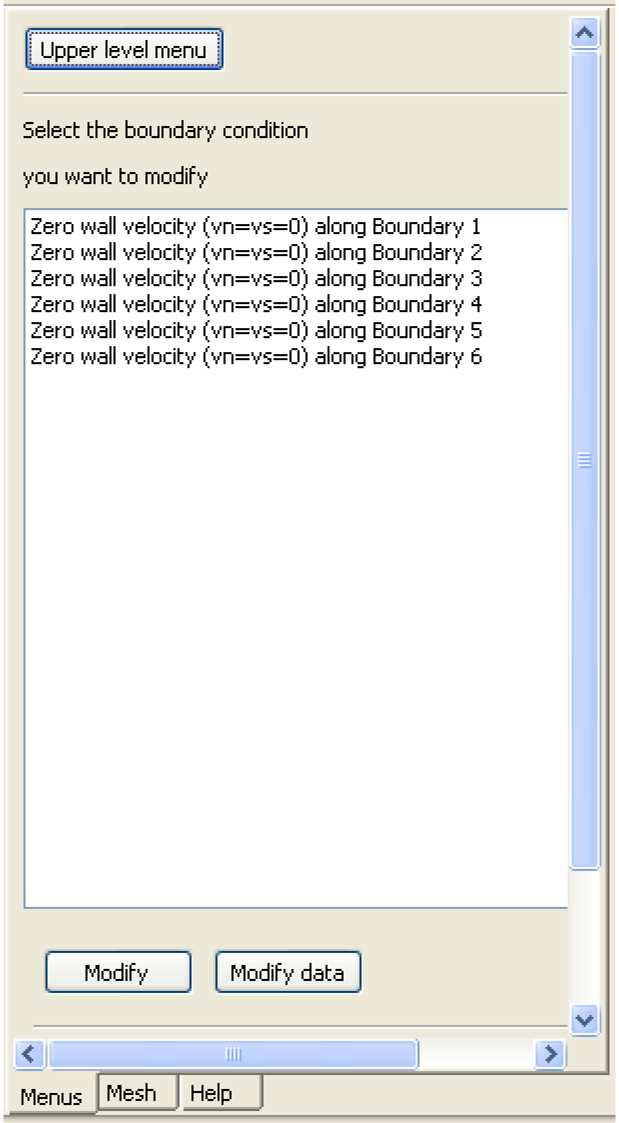
The general procedure for setting flow boundary conditions in ANSYS POLYDATA is presented below:

1. Select the **Flow boundary conditions** menu item in the sub-task menu.

   Flow boundary conditions

   ANSYS POLYDATA will update the display to show the default boundary conditions, as shown in *Figure 8.1* (p. 175).

Figure 8.1 The Flow boundary conditions Panel



Select the boundary condition

you want to modify

Zero wall velocity (vn=vs=0) along Boundary 1
Zero wall velocity (vn=vs=0) along Boundary 2
Zero wall velocity (vn=vs=0) along Boundary 3
Zero wall velocity (vn=vs=0) along Boundary 4
Zero wall velocity (vn=vs=0) along Boundary 5
Zero wall velocity (vn=vs=0) along Boundary 6

2. Select the boundary section (boundary set or intersection with an adjacent subdomain) for which you want to set the conditions.

3. Click **Modify**. ANSYS POLYDATA will display a menu of boundary condition choices that are appropriate for the selected boundary section.

4. Choose the boundary condition you want to impose, and set the related parameters as described in the corresponding section of this chapter.

8.2. Zero Wall Velocity Condition

The zero wall velocity condition allows you to set the normal and tangential velocity components on the boundary section to zero. To define the zero wall velocity condition on a boundary section, select the **Zero wall velocity (vn=vs=0)** menu in the list of boundary condition choices.

Zero wall velocity (vn=vs=0)

where **vn** = the normal velocity component

vs = the tangential velocity component

Note

In 3D, **vs** is a vector with two components.

8.3. Slip Condition

For the slip condition in ANSYS POLYFLOW, a zero normal velocity component is imposed simultaneously with one of three relationships between the shear force and the tangential relative velocity.

8.3.1. Shear Force Calculation

The default relationship is the generalized Navier's law:

$$f_s = F_{slip}\,(\mathbf{v}_{wall} - \mathbf{v}_s)\,\left|\mathbf{v}_s - \mathbf{v}_{wall}\right|^{e_{slip} - 1} \qquad \textbf{(8–1)}$$

where $\mathbf{v}_s$ is the tangential velocity of the fluid, $\mathbf{v}_{wall}$ is the tangential velocity of the wall, and F_{slip} and e_{slip} are material parameters. $\mathbf{v}_{wall}$ is assumed to be zero, by default.

Note that full slip is obtained when $F_{slip} = 0$. *Equation 8–1* (p. 176) is either linear, with $e_{slip} = 1$, or of the full power law type with $0 < e_{slip} < 1$.

For generalized Newtonian flow, $e_{slip} = 1$ preserves the linear character of the flow problem. When $e_{slip} < 1$, the problem becomes nonlinear and requires an iterative solution technique. In 2.5D and in 3D, the s subscript stands for both tangential directions, as it would be unphysical to allow different slip laws in different directions of the tangent subspace.

Another relationship is the threshold law:

$$f_s = \begin{cases} -F_{slip}\,(\mathbf{v}_s - \mathbf{v}_{wall})\ , & \mathbf{v}_s - \mathbf{v}_{wall} < y_c / F_{slip} \\ -y_c - F_{slip,2}\,(\mathbf{v}_s - \mathbf{v}_{wall} - \dfrac{y_c}{F_{slip}})\ , & \mathbf{v}_s - \mathbf{v}_{wall} \geq y_c / F_{slip} \end{cases} \qquad \textbf{(8–2)}$$

where F_{slip} and $F_{slip,2}$ are two different slip coefficients, and y_c is the critical force density at which slip starts to occur.

The third relationship is the asymptotic law:

$$f_s = -F_{slip}\left[1 - \exp\left(-\frac{\mathbf{v}_s - \mathbf{v}_{wall}}{v_c}\right)\right] \quad \textbf{(8–3)}$$

where v_c is a scaling factor with the dimensions of the velocity. v_c affects the slope of the slip-velocity curve.

8.3.2. User Inputs for the Slip Condition

To define slip conditions on a boundary section, do the following:

1. Select the **Slip conditions** menu item in the list of boundary condition choices.

   Slip conditions

2. Set the value for the velocity of the wall ($\mathbf{v}_{wall}$).

   Define v_wall, the velocity of the wall

   - To assign a translational velocity, specify the **translation velocity**.
   - To assign a rotational velocity in 2D, specify the **rotation center** and **angular velocity**.
   - To assign a rotational velocity in 3D, specify the **1st point of the axis**, **2nd point of the axis**, and **angular velocity**.

   In 2D, the rotation axis is perpendicular to the plane of the computational domain. In 3D, it is defined by two points.

3. Select the slip law you want to use and specify the related parameters:

   - To use *Equation 8–1* (p. 176) (the default), select **F(v) = Generalized Navier's law** and specify constant values for **k** (F_{slip}) and **e** (e_{slip}).

     F(v) = Generalized Navier's law

   - To use *Equation 8–2* (p. 176), select **F(v) = Threshold law** and specify constant values for **k** (F_{slip}), **k2** ($F_{slip,2}$), and **yc** (y_c).

     F(v) = Threshold law

   - To use *Equation 8–3* (p. 177), select **F(v) = Asymptotic law** and specify constant values for **k** (F_{slip}) and **vc** (v_c).

     F(v) = Asymptotic law

4. Assign a constraint on geometric normal vector.

   There can be a potential conflict of boundary conditions in some situations. You can have slipping boundary conditions on a boundary side and a free surface on an adjacent boundary side. The finite element contributions from both sides may result in a motion of the corresponding intersection point, which may receive an inappropriate velocity component.

For preventing possible difficulties, you may reinforce the constraint on the vanishing normal velocity component by canceling some component of the normal vector to the wall. This can be achieved by selecting:

Define constraints on wall normals

Specify the components of the geometric normal vector to the boundary to be canceled, to reinforce the constraint on vanishing velocity along the other directions.

Select one of the following as applicable to cancel the appropriate component of the geometric normal vector:

Cancel X-component of wall normals

Cancel Y-component of wall normals

Cancel Z-component of wall normals

The latter option is available in 3D flow. Such options have to be selected carefully, depending on the actual geometry. Also, at least one component should not vanish. For the above options, you can only cancel or restore the corresponding components of the wall normal vectors.

5. You can access advanced options for the slip condition definition by clicking **Advanced options** menu option.

   Advanced options

   - The slipping formulation utilizes a stabilization parameter referred to as “epsilon”, which can help resolve problems that arise from conflicting boundary conditions. You can modify the value of this parameter by clicking **Modify epsilon** and entering a new value in the dialog box that opens.

     Modify epsilon

     ### Important

     You should only modify the value of epsilon when recommended to do so by your support engineer.

   - Slipping involves the evaluation of an unknown force density field. Like other fields, this force density field must be interpolated. You can specify that the interpolation is: the same as that used for the velocity field (this is the default method); the same as that used for the pressure field; or constant at the level of the elements. To modify the method of interpolation for slipping, click **Modify interpolation** and enter a value that corresponds to your preferred method in the **Specify the interpolation rule** dialog box that opens.

     Modify interpolation

     In most cases, the default selection is suitable. When your calculation fails due to slipping, it is recommended that you try the interpolation that is constant at the level of the elements.

8.4. Porous Wall Condition

Porous wall conditions are used to model a thin “membrane" that has a known normal force on it. It is essentially a 1D simplification of the porous media model available with the Darcy flow sub-task. Examples of uses for the porous wall condition include screens and filters.

For the porous wall condition, a zero tangential velocity component is imposed simultaneously with one of three relationships between the normal force and the normal velocity.

8.5. Symmetry Condition

The symmetry condition is equivalent to imposing zero values for the normal velocity and tangential force. For axisymmetric models, ANSYS POLYDATA identifies the axis of symmetry ($r = 0$) automatically, and imposes the symmetry condition on that boundary section.

To define symmetry conditions on a boundary section, select the **Plane of symmetry (fs=0, vn=0)** menu item in the list of boundary condition choices. No further inputs are required.

☰ **Plane of symmetry (fs=0, vn=0)**

8.6. Inflow Condition

The inflow condition allows you to specify a volumetric or a mass flow rate across a boundary section surrounded by other boundary sections with well-defined adhesion conditions (slip, no-slip, symmetry, etc.). The convention here is that an entering flow rate is considered positive. For example, the volumetric flow rate is given as m^3/s-m for a 2D planar model and m^3/s for a 3D or axisymmetric model. Similarly, the mass flow rate is given as kg/s-m for the 2D planar model and kg/s for a 3D or axisymmetric model. Flow rate is defined per unit length in the direction perpendicular to the plane of the flow.

8.6.1. Inflow Calculation for Generalized Newtonian Flow

For a generalized Newtonian flow, ANSYS POLYFLOW computes the inflow from your specified volumetric or mass flow rate in two ways using a fully developed method and/or using a constant normal force. You can either explicitly specify a method or allow ANSYS POLYFLOW choose the best method.

- Fully developed method: In this method, ANSYS POLYFLOW solves a fully developed flow problem. Here it is assumed that all derivatives normal to the inlet are zero, except the pressure gradient, which is assumed to be constant. In 2D, ANSYS POLYFLOW solves a 1D problem along the segment, calculating the velocity profile at every node of the entry section, assuming fully developed flow. Similarly, in 3D, ANSYS POLYFLOW calculates a fully developed channel flow problem in 2D, assigning the nodal velocities in the entry section. ANSYS POLYFLOW will then impose the computed velocity profile as the prescribed velocity.
- Constant normal force method: In this method, the normal force is prescribed (assumed constant on the boundary section), with its value being required to obtain the prescribed flow rate. The value of the pressure is calculated by ANSYS POLYFLOW and the velocity components are *not* prescribed.

The two methods differ mainly for short inlet sections. In this case, assuming that the normal force (mostly the pressure) is constant, the fully developed method might be a better option if the inlet section is connected to a large reservoir. In the presence of significant gravitational forces acting tangent to the inlet, the constant normal force method does not result in a fully developed flow condition.

If ANSYS POLYFLOW is allowed to choose the best method for you, it will select the fully developed method. For inverse extrusion problems where the inlet is part of the constant section (parallel dies), however, the constant normal force method will usually be chosen.

8.6.2. Inflow Calculation for Viscoelastic Flow

For a viscoelastic flow, ANSYS POLYFLOW computes a fully developed velocity profile from your specified volumetric (or mass) flow rate and imposes it as a boundary condition for the flow problem. The fully developed stress profile is also computed and imposed as a boundary condition for the (hyperbolic) constitutive equation. ANSYS POLYFLOW solves a finite-element problem along the boundary section to calculate the velocity and viscoelastic stresses at every node of the inlet section.

The validity of the inflow condition requires the computational domain to be built in such a way that the basic assumptions of fully developed Poiseuille flow are satisfied. This is usually ensured when the inlet section is long enough. For axisymmetric geometries, the inlet section must be perpendicular to the axial direction. You can therefore consider that such a boundary condition replaces a long channel upstream of the inlet section.

8.6.3. User Inputs for the Inflow Condition

To define an inflow condition on a boundary section, do the following:

1. Select the **Inflow** menu item in the list of boundary condition choices.

   Inflow

2. Enter the value of flow rate, when prompted.

   Note

   The flow rate can be mass flow rate or volumetric flow rate.

3. For generalized Newtonian flow, select the method to compute the inflow: **Automatic** (which lets ANSYS POLYFLOW choose the appropriate method for you), **Fully developed**, or **Constant normal force**. These methods are described in *Inflow Calculation for Generalized Newtonian Flow* (p. 179). Keep the default selection of **Automatic**, to allow ANSYS POLYFLOW to choose the method for you.

   Automatic

4. Select the type of flow rate: volumetric flow rate or mass flow rate.

   *The default type is **Volumetric flow rate**.*

   Volumetric flow rate

   Note

   If you select **Mass flow rate**, you must define a positive and constant density.

8.7. Outflow Condition

The outflow condition in ANSYS POLYFLOW is different for generalized Newtonian flow and viscoelastic flow.

8.7.1. Outflow Condition for Generalized Newtonian Flow

The outflow condition for generalized Newtonian flow is similar to the normal force and tangential velocity condition described in *Normal Force and Tangential Velocity Condition* (p. 183), with $f_n = 0$ and $\mathbf{v}_s = 0$. It replaces a long channel at the exit of the flow domain by a single boundary condition. Note that an outflow condition can take into account deformations of the flow domain in the downstream channel.

8.7.2. Outflow Condition for Viscoelastic Flow

Like the generalized Newtonian outflow condition, the outflow condition for a viscoelastic flow consists of a zero tangential velocity. Instead of imposing the normal force, the viscoelastic outflow condition imposes a fully developed normal velocity profile, as in the inflow condition, but extra stresses are not imposed. The viscoelastic outflow condition makes it possible to prescribe flow boundary conditions in the presence of shear and nonzero normal forces. As for the inflow condition, the computational domain must be built in such a way that the basic assumptions of 1D Poiseuille flow are satisfied at the exit. The normal velocity profile is based on the flow rate that you specify. The convention is that an exiting flow is considered positive. The outflow condition should not be imposed at the exit of a free jet of a viscoelastic flow. For such cases, a zero normal force condition (see *Normal and Tangential Force Condition* (p. 182)) is preferred. The velocity distribution cannot be properly assigned since the geometric dimensions of the free jet are *a priori* unknown.

8.7.3. User Inputs for the Outflow Condition

To define an outflow condition on a boundary section, follow these steps:

1. Select the **Outflow** menu item in the list of boundary condition choices.

   Outflow

   (For a generalized Newtonian flow, no further inputs are required.)

2. For a viscoelastic flow, enter the value of flow rate, when prompted.

3. Select the type of flow rate: volumetric flow rate or mass flow rate.

   The default type is ***Volumetric flow rate****.*

   Volumetric flow rate

   Note

   If you select **Mass flow rate**, you must define a positive and constant density.

8.8. Free Surface Condition

ANSYS POLYFLOW offers advanced features for the calculation of free surfaces. Along such boundary sections, surface forces are imposed while the fluid cannot cross the free surface. Free surfaces and their related user inputs are described in *Free Surfaces and Extrusion* (p. 311).

8.9. Normal and Tangential Velocity Condition

The normal and tangential velocity condition allows you to specify the normal and tangential velocity components on the boundary section. The normal velocity component is denoted by v_n and the tangential component by v_s. The default condition corresponds to $v_n = v_s = 0$, but you can assign nonzero values for one or both of the components. Note that, in 3D, v_s is a vector with two components, not a scalar.

It is important to note the sign convention for v_n and v_s. A positive v_n means that the fluid emerges from the domain of the sub-task; a positive v_s is oriented (in 2D) along the counterclockwise direction on the closed curve surrounding the domain of the sub-task for an external boundary; for internal boundaries of 2D domains that are not simply connected, the orientation is reversed. Thus, in an entry section, v_n will be assigned a negative value.

In 3D, it is more complicated to assign a value of v_s, primarily because the definition of v_s relies on complex conventions of orientation. As a result, in 3D, the normal and tangential velocity condition is recommended for use only for a zero tangential velocity. If the tangential velocity in a 3D problem is nonzero, you should use the Cartesian velocity condition (described in *Cartesian Velocity Condition* (p. 184)) instead.

To define the normal and tangential velocity on a boundary section, select

the **Normal and tangential velocities imposed (vn, vs)** menu item in the list of boundary condition choices.

Normal and tangential velocities imposed (vn, vs)

Then specify v_n as a **Constant**, a **Linear function of coordinates**, a **Map from CSV (Excel) file**, or a **User Defined Function**. When you select **Upper level menu** after specifying v_n, ANSYS POLYDATA will bring you to the menu where you can specify v_s in the same manner.

8.10. Normal and Tangential Force Condition

The normal and tangential force condition is similar to the normal and tangential velocity condition described in *Normal and Tangential Velocity Condition* (p. 182). This condition is typically used for exit sections of free surface problems. The normal and tangential directions are defined as described in *Normal and Tangential Velocity Condition* (p. 182), and f_n and f_s are surface force *densities* expressed, for example, in Pa. A traction force on the boundary section produces a positive f_n. A counterclockwise tangential force produces a positive value of f_s (in 2D). In 3D, due to orientation conventions (as is the case for the tangential velocity component), it is difficult to assign a nonzero value for f_s. If the tangential force in a 3D problem is nonzero, you should use the global force condition (described in *Global Force Condition* (p. 184)) instead.

Important

For viscoelastic simulations, you should not impose a zero normal force at the inlet or outlet of the computational domain where a shear flow is expected. Due to normal stresses, f_n will, in general, not be zero when the shear rate is nonzero. Special viscoelastic outflow conditions (described in *Outflow Condition* (p. 181)) are provided for this purpose. Note that it is, in general, perfectly valid to impose a zero normal force at the exit of a free surface if no traction applies. In this case, the fully developed velocity profile corresponds to a plug flow and viscoelasticity does not generate normal stresses.

To define the normal and tangential force on a boundary section, select the **Normal and tangential forces imposed (fn, fs)** menu item in the list of boundary condition choices.

Normal and tangential forces imposed (fn, fs)

Then specify f_n as a **Constant** or a **Linear function of coordinates**. When you select **Upper level menu** after specifying f_n, ANSYS POLYDATA will bring you to the menu where you can specify f_s in the same manner.

8.11. Normal Velocity and Tangential Force Condition

The normal velocity and tangential force condition combines the concepts of the normal and tangential velocity and force conditions described in *Normal and Tangential Velocity Condition* (p. 182) and *Normal and Tangential Force Condition* (p. 182). A typical use of this condition is for the upper free surface of a basin with zero normal velocity and zero shear forces (or full slip along a wall). Another example is a moving boundary with an imposed uniform value of v_n and, in case of slip, a zero value of f_s. As mentioned in the previous sections, zero values for v_n and f_s are most common, but nonzero values are also possible (although not a nonzero value for f_s in 3D). If the tangential force in a 3D problem is nonzero, you should use the global force condition (described in *Global Force Condition* (p. 184)) instead.

To define the normal velocity and tangential force on a boundary section, select the **Normal velocity and tangential force imposed (vn, fs)** menu item in the list of boundary condition choices.

Normal velocity and tangential force imposed (vn, fs)

Then specify v_n as a **Constant**, a **Linear function of coordinates**, a **Map from CSV (Excel) file**, or a **User Defined Function**. When you select **Upper level menu** after specifying v_n, ANSYS POLYDATA will bring you to the menu where you can specify f_s as a **Constant** or a **Linear function of coordinates**.

8.12. Normal Force and Tangential Velocity Condition

The normal force and tangential velocity condition combines the concepts of the normal and tangential velocity and force conditions described in *Normal and Tangential Velocity Condition* (p. 182) and *Normal and Tangential Force Condition* (p. 182). A typical use of this condition is for an exit section where the normal force is zero and the tangential velocity is constrained. Note, however, that a specific outflow condition (described in *Outflow Condition* (p. 181)) is also available. The conventions and restrictions described in *Normal and Tangential Velocity Condition* (p. 182) and *Normal and Tangential Force Condition* (p. 182) apply to this condition as well.

To define the normal force and tangential velocity on a boundary section, select the **Normal force and tangential velocity imposed (fn, vs)** menu item in the list of boundary condition choices.

Normal force and tangential velocity imposed (fn, vs)

Then specify f_n as a **Constant** or a **Linear function of coordinates**. When you select **Upper level menu** after specifying f_n, ANSYS POLYDATA will bring you to the menu where you can specify v_s as a **Constant**, a **Linear function of coordinates**, a **Map from CSV (Excel) file**, or a **User Defined Function**.

8.13. Global Force Condition

There are some problems for which you may want to impose a contact force instead of a force density along a boundary section. The contact force is the integral of the force density along the boundary section. This is typical of fiber spinning, where the drawing force (global normal force) is imposed while the diameter of the filament is a priori unknown. With the global force boundary condition, you can impose a global force on the boundary section in, for example, Newtons. The local value of f_n will then depend upon the solution itself. This condition implies that the velocity profile is constant along the boundary section, although the value of this constant velocity is not prescribed.

To define global forces on a boundary section, select the **Global force imposed** menu item in the list of boundary condition choices.

Global force imposed

Then specify constant values for F_x, F_y, and (in 3D) F_z. These are the components of the force in each coordinate direction.

8.14. Cartesian Velocity Condition

The Cartesian velocity condition allows you to prescribe a rigid-body velocity field on the boundary section. Here, the velocity is prescribed in a Cartesian reference frame instead of a normal/tangential frame. This option is especially useful for 3D flows, for which tangential directions cannot be uniquely identified.

In a general form, the rigid-body velocity $\mathbf{v}$ is obtained as the combination of a translational velocity $\mathbf{v}^*$ and an angular velocity Ω around a rotation axis:

$$\mathbf{v} = \mathbf{v}^* + \Omega \times (\mathbf{x} - \mathbf{x}^*) \qquad \textbf{(8–4)}$$

For 2D flow problems, the rotation axis is perpendicular to the plane of the computational domain, and is defined by one point. For 3D flow problems, the rotation axis is defined by two points. In *Equation 8–4* (p. 184), $\mathbf{x} - \mathbf{x}^*$ is the position vector of a point on the boundary section with respect to the axis of rotation.

The translational velocity should, in general, be used when a boundary section is moving in a fixed direction (e.g., a moving belt), while the angular velocity should be used when the boundary section is rotating (e.g., inside an extruder).

Important

For viscoelastic flows, note that only the inflow condition (described in *Inflow Condition* (p. 179)) properly imposes the extra-stress components on inlet boundary sections.

To define Cartesian velocity conditions on a boundary section, do the following:

1. Select the **Cartesian velocity imposed (vx,vy,vz)** menu item in the list of boundary condition choices.

   Cartesian velocity imposed (vx,vy,[vz])

2. Specify the translational velocity, if appropriate.

   Modify translation velocity

   In 2D, you will enter two Cartesian components of the translational velocity $\mathbf{v}^*$; in 3D, you will enter three Cartesian components.

3. Specify the rotational velocity, if appropriate.

   a. Specify the rotation axis. In 2D, you will specify the **rotation center**; in 3D, you will specify the **1st** and **2nd point of the axis**.

   b. Specify the **angular velocity** Ω.

8.15. Velocity Profile from a CSV File

In addition to the specification of boundary conditions within ANSYS POLYDATA, as described in the previous sections, you can also specify a velocity profile outside of ANSYS POLYDATA (e.g., using a spreadsheet such as Excel), and then read it in to define the conditions on a boundary section.

The format for the profile data must be a CSV (comma separated variables) file, which is a common format for tabulated data that can be read into spreadsheet programs such as Excel. The profile file should contain a list of data points (x, y, z) and a list of velocity vectors (v_x, v_y, v_z). For more information about setting up a file with the correct format, you can follow the instructions in *Using the CSV File to Initialize Solution Variables* (p. 151) for creating a template CSV file.

When ANSYS POLYFLOW reads a CSV file, it will interpolate the data (if they do not correspond exactly to the nodes in the mesh) between the listed values; the node in the mesh that is located closest to a data point will be initialized to the specified value.

To define velocity conditions on a boundary section, follow these steps:

1. Select the **Velocity profile from CSV file** menu item in the list of boundary condition choices.

   Velocity profile from CSV file

2. Specify the name of the CSV file and the label that indicates the velocity data in the CSV file.

Note that boundary conditions imposed via a CSV file are evaluated only once. In other words, if the CSV file contains a velocity distribution in space, the evaluation will be based on the geometry that is known at the beginning of the calculation, even if the boundary is moving.

8.16. Interface Condition

The interface condition establishes the continuity between the two sides of an intersection between different sub-tasks. For the momentum equation, the interface condition guarantees continuity of the velocity field and of the contact forces. The interface condition is available only for the boundary section that is defined by an intersection with an adjacent subdomain.

Important

If you prescribe an interface condition for one sub-task, you must also prescribe an interface condition for the sub-task on the other side of the boundary.

Note that it is also acceptable to define a condition other than the interface condition at the intersection between two sub-tasks. For example, you can impose a zero velocity field to model a wall of zero thickness between the two sub-tasks.

Interfaces can be fixed or moving. For a moving interface, a kinematic condition is added to the system of equations, in order to solve for the position of the interface. Moving interfaces are described in detail in *Free Surfaces and Extrusion* (p. 311). Note that a fixed interface condition defines a linear problem, unless a nonlinearity arises from another parameter or equation.

To define an interface condition between two flow sub-tasks, simply choose the **Interface** menu item in the list of boundary condition choices.

Interface

To define an interface condition between a flow sub-task and a porous (Darcy) flow sub-task, simply choose the **Interface with porous media** menu item in the list of boundary condition choices.

Interface with porous media

See *Porous Media* (p. 305) for more information about defining a porous flow sub-task.

8.17. Interface with Elastic Solid

On the fluid side, interface with elastic solid means the fluid sticks or slips on the wall. But additionally, the force applied from the fluid to the wall is computed and acts as a boundary condition on the solid. The stick condition allows the application of zero velocity along the wall. ANSYS POLYFLOW computes the force required to satisfy this condition. This method needs a stability coefficient in order to avoid numerical trouble at points where the velocity is already imposed.

This stability coefficient is dimensionless and must be small with respect to 1. If the stabilization coefficient is too big, it perturbs the solution all over the contact wall, and the velocity boundary condition will be not fully satisfied. For slip condition, a zero normal velocity component is imposed simultaneously with a relationship between the shear force and the tangential relative velocity. The relationship is the Navier's law:

$$f_s = F_{slip} \left(V_{wall} - V_s \right) \qquad \textbf{(8–5)}$$

where V_s is the tangential velocity, V_{wall} is the tangential velocity of the wall, and F_{slip} is a material parameter. In the frame of the fluid-structure interaction, V_{wall} = 0 and there is a penalty coefficient to guarantee the zero normal velocity. This coefficient must be large enough with respect to F_{slip} coefficient. If the penalty coefficient is too small, the zero normal velocity condition is not satisfied.

To define an interface with elastic solid on a boundary section, select **Interface with elastic solid menu** item in the list of boundary condition choices.

Interface with elastic solid

The default is the stick condition. You can switch between the stick and the slip conditions by clicking one of the following:

Switch to slip condition

Switch to stick condition

You can specify various settings, depending on the condition you select:

- If you are using the stick condition, specify the stability coefficient. This stability coefficient can help resolve problems that arise, for example, when boundary conditions conflict.
- If you are using the slip condition, specify the value of F_{slip} in *Equation 8–5* (p. 187), as well as the penalty coefficient.
- For either the stick or the slip condition, you can specify the interpolation rule. The intermediate field that transfers the force information between the fluid and the solid must be interpolated (as is the case for other fields). You can specify that the interpolation is: the same as that used for the velocity field (this is the default method); the same as that used for the pressure field; or constant at the level of the elements. In most cases, the default selection is suitable. When your calculation fails due to FSI conditions, it is recommended that you try the interpolation that is constant at the level of the elements.

8.18. Periodic Condition

Periodic boundary conditions are used when the physical geometry of interest and the expected flow pattern have a periodically repeating nature. This means that the flows across two opposite planes in your computational model are identical. Periodic boundary conditions can be applied to a pair of boundary sections, which are referred to as the inlet and outlet of the periodic condition. The velocity profile for the inlet and outlet will be the same, but a difference of normal force or pressure will be allowed between them.

To define periodic boundary conditions on a pair of boundary sections, select one of them (the order is not significant) and follow these steps:

1. Select the **Inlet of periodic condition** menu item in the list of boundary condition choices.

   Inlet of periodic condition

2. In the resulting **Inlet of periodic cond.** panel, select the corresponding periodic boundary section (the outlet of the periodic condition) and click **Select**.
3. Specify the transformation of coordinates and velocity field on the inlet to those along the outlet. There are two ways to do this:

   - Specify a rotation matrix and/or translation vector explicitly. For rotational periodicity, specify the rotation matrix.

     Direct modification of rotation matrix

     For translational periodicity, specify the translation vector.

     Direct modification of translation vector

     The rotation matrix and/or translation vector, once applied to the coordinates of a node along the inlet, will produce the coordinates of the corresponding node along the outlet.
   - Specify a pair of corresponding points, one on the inlet and one on the outlet.

     Definition by source-target points

     ANSYS POLYFLOW will determine the proper rotation matrix and translation vector based on these two points, and then apply it to the rest of the points on the inlet to obtain the points on the outlet.

4. Check the transformation to be sure it is consistent with the mesh.

   Test of the transformation
5. When you are satisfied with the inputs, accept the settings.

   Accept current parameters
6. Specify either the flow rate

   Flow rate imposed

   or the difference in normal force

   Normal force imposed

   between the inlet and the outlet.

Note that there are no inputs for the outlet of the periodic condition.

8.18.1. Normal Force Calculation

The default relationship is the power law:

$$f_n = -\alpha v_n^e \tag{8–6}$$

where α is the permeability of the porous wall.

Another relationship is the threshold law:

$$f_n = \begin{cases} -\alpha v_n, & v_n < y_c/\alpha \\ -y_c - \alpha_2 \left(v_n - \dfrac{y_c}{\alpha}\right), & v_n \geq y_c/\alpha \end{cases} \tag{8–7}$$

where α and α_2 are two different permeabilities, and y_c is the critical force density.

The third relationship is the asymptotic law:

$$f_n = -\alpha \left[1 - \exp\left(-\frac{v_n}{v_c} \right) \right] \tag{8–8}$$

where v_c is a scaling factor with the dimensions of the velocity.

8.18.2. User Inputs for the Porous Wall Condition

To define a porous wall condition on a boundary section, follow these steps:

1. Select the **Porous wall** menu item in the list of boundary condition choices.

   Porous wall

2. Select the law you want to use and specify the related parameters:

   - To use *Equation 8–6* (p. 189) (the default), select **F(v) = Power law**.

     F(v) = Power law

     Then specify constant values for **k** (α) and **e** (e).

   - To use *Equation 8–7* (p. 189), select **F(v) = Threshold law**.

     F(v) = Threshold law

     Then specify constant values for **k** (α), **k2** (α_2), and **yc** (y_c).

   - To use *Equation 8–8* (p. 189), select **F(v) = Asymptotic law**.

     F(v) = Asymptotic law

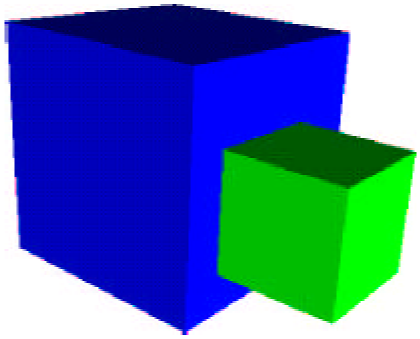
     Then specify constant values for **k** (α) and **vc** (v_c).

8.19. Non-Conformal Boundaries

It is possible for ANSYS POLYFLOW to compute a solution on a domain consisting of geometrically coincident but disconnected parts (e.g., *Figure 8.2* (p. 190)). Consider a conformal mesh to be one that reflects the continuity of the domain through element connectivity, and a non-conformal mesh to be a region meshed in different parts where the boundary regions are geometrically—but not topologically—coincident (e.g., *Figure 8.3* (p. 190)).

The non-conformal boundary condition allows you to create an approximate conformity of the finite-element representation.

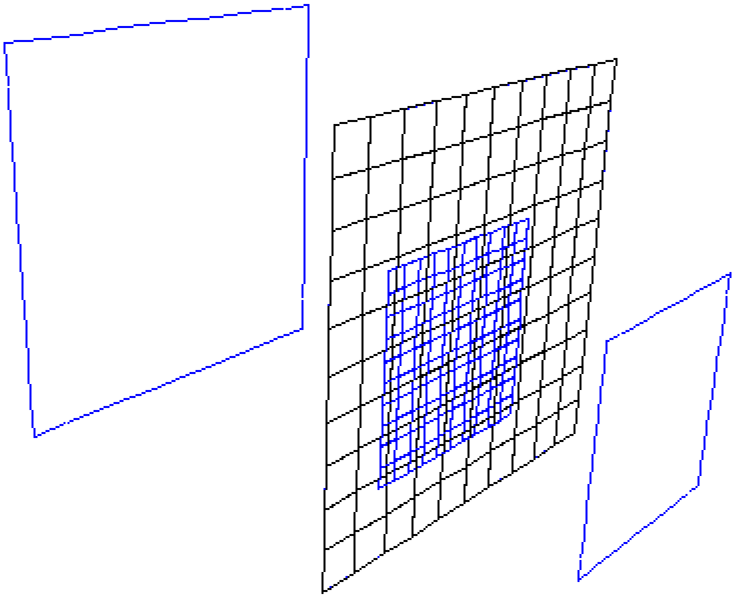
Figure 8.2 Domain with Geometrically Coincident but Disconnected Parts



For complex geometries (particularly those with regions of different scales, such as the inlet region of a profile tread die), the use of non-conformal meshes can be very beneficial, since creating a conformal mesh through all parts of the simulation domain is often difficult for such geometries.

With a non-conformal mesh, it is not necessary to propagate the small elements of the “small" region to the larger (e.g., inlet) zone. Note, however, that solutions on non-conformal meshes are generally less accurate than those on conformal meshes.

Figure 8.3 Non-Conformal Boundary Meshes



8.19.1. Solution Technique for Non-Conformal Boundaries

When two sections are associated along a non-conformal boundary, a geometrical tolerance dictates the search for correspondences. For every “source" boundary degree of freedom (at every mesh node, midface node, etc.), a correspondence is found in terms of a linear combination of “target" boundary degrees of freedom.

So source and target boundaries do not have to be coincident in the element connectivity sense; continuity is enforced by constraining the particular combination of variables on the target boundary to be equal (in an approximate sense) to the variables on the source boundary. For this technique to work well, the source boundary should correspond to the coarser mesh, and the target boundary to the finer mesh. The resulting level of accuracy will be what would be expected from the coarser mesh.

On portions of the sections that are not in contact (up to the geometrical tolerance) with a section on the corresponding boundary, a zero velocity condition will be imposed automatically, and a zero flux condition will be imposed for all other variables.

Accuracy of mass conservation is a reasonable measure of the quality of the non-conformal interface. You can check this by looking at the mass fluxes through boundaries that appear in the ANSYS POLYFLOW listing file or viewing the mass balance in ANSYS POLYDIAG.

8.19.2. Connecting Non-Conformal Boundaries

If your domain includes non-conformal boundaries between adjacent sections, you will need to “connect" them when you specify flow and/or thermal boundary conditions.

Important

As noted in *Solution Technique for Non-Conformal Boundaries* (p. 191), it is recommended that the boundary with a coarser mesh should be specified as the “source" boundary, and the boundary with a finer mesh should be specified as the “target" boundary. If both boundaries have comparable mesh resolution, you can choose either one to be the source or target boundary.

To connect non-conformal boundaries, follow these steps (after selecting the boundary that will be the source boundary):

1. Select the **Source of connected condition** menu item in the list of boundary condition choices.

   Source of connected condition

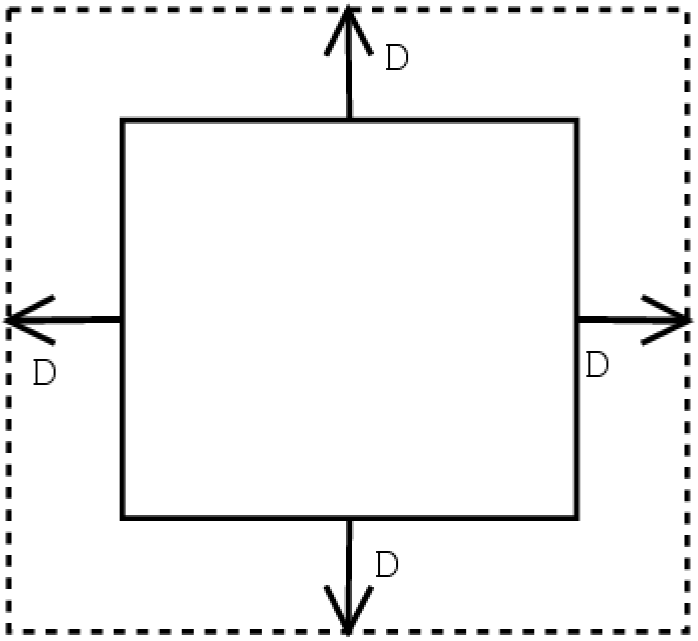
2. In the resulting panel, select the corresponding target boundary.
3. Specify the numerical parameters related to the non-conformal boundary condition.

   a. Specify the element dilatation, if necessary.

      Modify element dilatation

      This option allows for expansion of each face of the source boundary in order to detect the corresponding face on the target boundary (see *Figure 8.4* (p. 192)). The default value is set according to the dimensions of the mesh, and you will generally not need to modify it. The element dilatation must be entered in the unit of length used for the mesh.

Figure 8.4 Element Dilatation



b. Specify the amplitude of the volume generation.

Modify amplitude of volume generation

This value specifies the amount by which a source face can be expanded in order to detect the corresponding target face. The face is extended to a 3D volume in the local normal direction, by a constant amount that is precisely equal to the specified amplitude. This value must be entered in the unit of length used for the mesh.

c. Specify the stabilization factor, if necessary.

Modify stabilization factor

This parameter controls the amount of fluid that is allowed to escape through faces that are connected to only one of the connected boundaries (i.e., just to the source boundary or just to the target boundary). Ideally, no fluid should pass through these faces, but setting this value to zero will destabilize the solution. It is therefore set to a very small nonzero value, to avoid fluid loss without causing numerical instabilities. Change the value of this parameter from the default value only on the advice of your support engineer.

8.20. Specifying Conditions Using Sub-Models

Sub-models allow you to modify some equations and boundary conditions by constraining, modifying, and adding simulation variables, either internal to the domain or on its boundary. For example, you can impose a kinematic constraint on a part of a free surface to model a particular take-up device, or model a source/sink of heat or an internal force contribution. You can also model heat-conducting devices such as plates or wires embedded in the simulation domain, but distinct from the continuum being modeled.

Sub-models require the specification of at least two pieces of information:

- the type of model and the associated parameters

- the region of the mesh where the sub-model applies (called a “topo-object")

In the case of a porous jump sub-model, it is also necessary to provide a preliminary definition of a dataset, which will then be accessed by ANSYS POLYDATA when you create the sub-model. See *Defining a Material Dataset* (p. 197) for further details.

> **Important**
>
> Because of the general applicability of sub-models, ANSYS POLYDATA is not able to perform consistency checks. It is up to you to ensure the correspondence between the underlying continuum model and the sub-model.

8.20.1. The Topo-Object

The region where the sub-model applies (the topo-object) can be a subdomain, a boundary set (in which case the model applies to this region of the boundary), or an internal region of the mesh, referred to as a PMesh. PMeshes are described in *PMeshes* (p. 134).

You can define a sub-model on a group of elements, provided that they are all of the same physical dimension (0D, 1D, 2D, or 3D). Depending on the dimension of the domain of the sub-model, ANSYS POLYFLOW will apply the appropriate contribution to the governing equations.

8.20.2. Types of Sub-Models

There are eight types of sub-models:

- Imposed velocity or force

  This model imposes either the Cartesian or the normal/tangential components of the velocity or force-density vector. The units of the force density will depend on the dimension of the elements in the sub-model domain:

  - force in 0D
  - force per unit length in 1D
  - force per unit surface in 2D
  - force per unit volume in 3D

- Imposed temperature

  This model imposes either a constant value for temperature or a linear function of the coordinates.

- Imposed heat flux density

  This model adds a contribution to the energy equation that represents the heat flux density on the domain for which the sub-model applies. The units of the heat flux density will depend on the dimension of the elements in the sub-model domain:

  - flux per unit length in 1D
  - flux per unit surface in 2D

- Heat conduction problem

  This model adds a conductive contribution to the energy equation, to model a conductive wire or plate (usually represented as a PMesh) that is distinct from the material of the domain itself.

- Pointwise velocity imposed

  This model imposes the velocity at one point through a penalty formulation.

- Coordinates imposed

  This model fixes any component of the coordinates on a sub-model domain. This sub-model allows you to forbid partial or total deformation on a part of a domain subject to remeshing.

- Diffuse gray wall imposed

  This model specifies that a fraction of the incident radiative energy is dissipated as heat in the boundary, another fraction passes through the boundary, and the rest is reflected. The diffuse gray wall boundary condition is only available when modeling internal radiation (see *Internal Radiation* (p. 293)).

- Porous jump model

  This model can be selected to represent a thin, rigid membrane that is porous (e.g., a filter), which allows fluid to pass through but is associated with a drop in the pressure. The constitutive equation of the porous jump sub-model relates the pressure drop (Δp) to the normal velocity component (v_n), and is given by:

$$\Delta p = \frac{h}{\kappa} \eta H\,(T)\;v_n \qquad \textbf{(8–9)}$$

  where h is the thickness of the rigid membrane, η is the reference viscosity, and $H\,(T)$ is a function that represents the temperature dependence of the viscosity. κ is the membrane permeability, and has units of L^2 (where L is the units for length). Since various membranes can be defined, the material parameters are stored in a local database. See *Defining a Material Dataset* (p. 197) for further details.

8.20.3. Defining a Sub-Model

The procedure for defining a sub-model is as follows:

1. In the task menu, select the **Define sub-models** menu item.

   Define sub-models

2. If you need to define a topo-object, follow the procedure outlined in *Defining a Topo-Object* (p. 197).
3. If your simulation will involve a porous jump sub-model, define the necessary material dataset by following the procedure outlined in *Defining a Material Dataset* (p. 197).
4. Select the **Create a new sub-model** menu item.

   Create a new sub-model

5. Select the appropriate sub-model type.

   V or F imposed

   Temperature imposed

Heat flux density imposed

Heat conduction problem

Pointwise velocity imposed

Coordinates imposed

Diffuse gray wall imposed

Porous jump model

See *Types of Sub-Models* (p. 193) for a description of the sub-model types.

6. When prompted, specify a title for the sub-model.
7. Specify the domain of the sub-model.

   Domain of the sub-model

   You can select from the topo-objects that are available in the mesh. These can be PMeshes that were generated during the original mesh generation, topo-objects that you created in ANSYS POLYDATA (as described in *Defining a Topo-Object* (p. 197)), or intersections between boundary sets and subdomains in the mesh. Note that the notation S1*B2 indicates the intersection between subdomain 1 and boundary set 2.

   ### Important

   Note that for the imposed diffuse gray wall sub-model, only PMeshes can be selected for the domain.

8. Specify the parameters related to the sub-model type you have chosen:

   - For an imposed velocity sub-model, the inputs are as follows:

     Velocity components imposed

     – For Cartesian velocity components, keep the default mode and specify the value of v_x, v_y, and (in 3D) v_z.

     – For normal/tangential velocity components, select **Enable Normal/Tangential Mode** and then specify the value of v_n and v_s.

   - For an imposed temperature sub-model, the inputs are as follows:

     Temperature imposed

     Specify the temperature as a **constant** or as a **linear function of coordinates**.

   - For an imposed heat flux sub-model, the inputs are as follows:

Heat flux density coefficients

Enter the coefficients for the equation shown in the ANSYS POLYDATA interface.

- For a heat conduction sub-model, the inputs are as follows:

  a. Specify the matrix coefficients for the anisotropic conductivity.

  #### Anisotropy matrix coefficients

  The conductivity can either be a scalar, in which case the anisotropy matrix is the unit matrix, or a tensor. If it is a tensor, the conductivity matrix (which must be positive-definite) is the product of the scalar conductivity and the matrix specified here. You need to input the scalar components of the matrix.

  b. Specify the thermal conductivity.

  #### Thermal conductivity

  c. Specify the shell thickness/area.

  #### Shell thickness

  Because the volumetric conductivity is entered for all heat conduction sub-models, the equations must be multiplied by the shell thickness (2D) or by the wire section area (1D) before being added to the energy equation.

- For an imposed pointwise velocity sub-model, the inputs are the local requested velocity and the penalty coefficient (which must be high enough to impose the velocity effectively on the sub-model domain).
- For an imposed coordinates sub-model, the inputs allow you to fix any of the components of the coordinates (i.e., if you fix the x-component, the message displayed will be `Constraint on X-component: dX=0`) or let any component be free (i.e., if you let the x-component be free, the message displayed will be `No constraint on X-component`).
- For an imposed diffuse gray wall sub-model, the inputs include the transmittance of the boundary (τ_w in *Equation 13–15* (p. 296)), as well as the emissivity values and refractive indices assigned to the faces on each side of the boundary (ε_w and n in *Equation 13–15* (p. 296), respectively). See *User Inputs for Internal Radiation Model* (p. 297) for complete details.
- For a porous jump sub-model, the inputs are as follows:

  Establish a link to a previously defined material dataset (see *Defining a Material Dataset* (p. 197) for further details), by clicking the **Link to a material dataset** item.

Link to a material dataset

Then select the name of the relevant material dataset for the current sub-model. For the porous jump sub-model, the dataset should contain numerical values of parameters that are relevant for *Equation 8–9* (p. 194).

Next, specify the porous medium thickness (h in *Equation 8–9* (p. 194)) by clicking the **Porous medium thickness** item.

Porous medium thickness

Then enter a **New value** for the thickness in the panel that opens and click **OK**. Although the porous jump model is defined on a surface in 3D or a line in 2D, it corresponds to a porous rigid obstacle which has an actual thickness. When defining this sub-model, it is assumed that the thickness of the membrane is small compared to its other dimensions.

8.20.4. Defining a Topo-Object

If you do not have a PMesh within your computational mesh, you can create a topo-object. This allows you to define a point, line, surface, or group of elements within the domain where you want a sub-model to apply. The topo-objects you define can be either the intersection or union of two existing topo-objects. By default, all intersections between adjacent boundary sets and subdomains are predefined as topo-objects.

To define a new topo-object, perform the following steps:

1. Click the **Define topo-objects** item in the **Define sub-models** menu.

   Define topo-objects

   The **Define topo-objects** menu will open.
2. Click the **Create a new topo-object** item.

   Create a new topo-object

   The **Add topo-objects** menu will open.
3. In the **List of existing topo-objects**, select the first topo-object to be used in the intersection or union.
4. Click **Select object 1**.
5. In the **List of existing topo-objects**, select the second topo-object to be used in the intersection or union.
6. Click **Select object 2**.
7. Using the drop-down list next to **Select operator**, select either **intersection** or **union**.
8. Specify a name for the topo-object in the **Enter a new name** field.
9. Click **create**.

You can now select this topo-object as part of the domain of the sub-model, as described in *Defining a Sub-Model* (p. 194).

8.20.5. Defining a Material Dataset

When simulating a membrane using the porous jump sub-model, it is necessary to define a dataset for the material parameters in *Equation 8–9* (p. 194) by performing the following steps:

1. Click the **Define material datasets** item in the **Define sub-models** menu.

   Define material datasets

The **Define material datasets** menu opens.

2. Click the **Create a new material dataset** item.

   Create a new material dataset

   A panel opens in which you can revise the default value for **New value**, which will act as the name for the dataset. Click **OK** in the panel to close it. Then read any warning panels that open, click **OK** in these panels, and take any necessary corrective actions. The **<material dataset name>** menu will then open (where **<material dataset name>** is the name you entered in the previous panel).

3. Click the **Add data** item.

   Add data

   The **Add data to a material dataset** menu will open, where you can specify which material parameters will be defined in the dataset.

   a. Click **Add porous medium and fluid viscosity**.

      Add porous medium and fluid viscosity

   b. For nonisothermal flows, click **Add temperature dependence of viscosity**.

      Add temperature dependence of viscosity

   c. Click **Upper level menu** to return to the previous menu.

4. Define the material parameters for *Equation 8–9* (p. 194).

   a. Define the permeability (κ) of the porous membrane and the reference viscosity (η) of the fluid as it flows through the membrane.

      Porous medium and fluid viscosity

      The **Porous medium and fluid viscosity** menu will open.

      i. Click the **Modify permeability** item.

         Modify permeability

         Enter a **New value** for the permeability in the panel that opens, and click **OK**.

      ii. Click the **Modify fluid viscosity** item.

         Modify fluid viscosity

         Enter a **New value** for the fluid viscosity in the panel that opens, and click **OK**. The fluid viscosity is represented by a single number, and should be selected in accordance with the actual fluid viscosity and the geometry of the porous membrane.

      iii. Click **Upper level menu** to return to the previous menu.

b. For nonisothermal flows, define the relationship between temperature and the viscosity of the fluid as it flows through the porous membrane ($H\ (T)$).

Temperature dependence of viscosity

The **Temperature dependence of viscosity** menu will open. Select the law you want to use, and specify the relevant parameters:

- Click **No temperature dependence** if you do not want the viscosity to be temperature-dependent (i.e., $H\ (T) = 1$). No further inputs are required.
- Click **Arrhenius approximate law** to define $H\ (T)$ using *Equation 10–19* (p. 214) with temperatures at constant shear rate. The inputs for this law are α and T_{α}.
- Click **Arrhenius law** to define $H\ (T)$ using *Equation 10–17* (p. 213) with temperatures at constant shear rate. The inputs for this law are α, T_{α}, and T_0. To specify T_{α} as an absolute temperature, set the temperature shift T_0 to 0. Otherwise, set T_0 to the appropriate temperature shift and then specify T_{α} relative to T_0.
- Click **Fulcher dependence** to define $H\ (T)$ using *Equation 10–20* (p. 214). The inputs for this law are f_1, f_2, and f_3.
- Click **WLF dependence** to define $H\ (T)$ using the WLF equation (*Equation 10–21* (p. 214)) based on shear rate (i.e., with only a vertical shift in the η- $\dot{\gamma}$ diagram). The inputs for this law are c_1, c_2, T_a, and $(T_r - T_a)$. The default values for c_1 and c_2 are appropriate for many polymer processing applications.

5. Click **Upper level menu** repeatedly to return to the **Define material datasets** menu, and then repeat the steps 2.–4.(b) for each additional material dataset you want to create.

8.21. Using a Rotating Reference Frame

When you solve a problem in ANSYS POLYFLOW, you are typically modeling the flow in an inertial reference frame (i.e., in a nonaccelerating coordinate system). However, ANSYS POLYFLOW also has the ability to model flows in an accelerating reference frame. In this situation, the acceleration of the coordinate system is included in the equations of motion describing the flow.

A common example of an accelerating reference frame is flow in rotating equipment. Many such flows can be modeled in a coordinate system that is moving with the rotating equipment and thus experiences a constant acceleration in the radial direction. This class of rotating flows can be treated using a rigid rotation task in ANSYS POLYFLOW.

For a rigid rotation task, ANSYS POLYFLOW uses the relative velocity as the dependent variable, instead of the absolute velocity.

The relative velocity ($\mathbf{v}_r$) is related to the absolute velocity ($\mathbf{v}$) by the following equation:

$$\mathbf{v}_r = \mathbf{v} - \Omega \times \mathbf{x} \qquad \textbf{(8–10)}$$

where Ω is the angular velocity of the rotating frame and $\mathbf{x}$ is the position vector in the rotating frame.

To set up a rigid rotation task, choose **Rigid rotation** (in addition to any other appropriate options) as the type of task.

Create a new task

When prompted, enter the value of the angular velocity Ω.

Chapter 9: Material Properties

This chapter provides general information about material properties, including information about defining properties as algebraic functions and reading material properties from a file.

Note that specific information about individual material properties is not provided in this chapter. For information about specific property inputs, see the appropriate chapter for the type of model you are defining (e.g., see *Conduction and Convection* (p. 283) for information about thermal properties such as conductivity).

9.1. Overview of Material Properties

An important step in the setup of your model is the definition of the physical properties of the material(s) being modeled. Material properties are defined in the **Material data** menu, which allows you to input values for the properties that are relevant to the type of problem you are defining. For example, input of thermal properties such as conductivity will be possible only if you are modeling nonisothermal flow or heat conduction in a solid region. Note that the option to change the system of units for the model, which appears in the **Material data** menu, is described in *Unit Systems* (p. 129).

In addition to the basic specification of properties as constant values or linear functions of coordinates, it is also possible to specify properties as more complicated algebraic functions. This capability is described in detail in *Specifying Material Properties as Algebraic Functions* (p. 201).

Once you have defined the material properties for your model, it is possible to save them to a file for use in another model, as described in *Reading and Writing Material Data Files* (p. 204).

It is also possible to determine the appropriate property inputs based on experimental or other external data. See *Curve Fitting for Material Properties* (p. 205) for more information.

9.2. Specifying Material Properties as Algebraic Functions

Often material properties are nonlinear algebraic functions of the primary field variables, such as temperature, pressure, and chemical species, to name a few. ANSYS POLYDATA allows you to customize the definition of material properties by defining one or more functions. This feature is referred to as the PMAT feature, because it makes use of the **PMAT** button at the top of the ANSYS POLYDATA menu. The PMAT feature allows you to make a property dependent on local variables through algebraic functions that you define yourself.

Important

Note that it is also possible to use user-defined functions in conjunction with the PMAT feature. See *User-Defined Functions (UDFs)* (p. 597) for details.

9.2.1. Example

The use of PMAT is best illustrated by an example. Consider, for example, that you want to express the overall specific heat capacity as a function of two locally existing species (A and B) as follows:

$$c_p = c_{p,A} X_A + c_{p,B} X_B \tag{9–1}$$

where $c_{p,A}$ and $c_{p,B}$ are the specific heat capacities of species A and B, and X_A and X_B are the mass fractions of species A and B.

This relationship can be restated as

$$c_p = VF \tag{9–2}$$

where $V = 1$ is a specified value and

$$F = f_1 + f_2 \tag{9–3}$$

with f_1 and f_2 as linear functions of X_A and X_B, respectively.

Clearly, F can be regarded as a sum of elementary functions of the polynomial form.

9.2.2. User Inputs

The procedure for defining a property using algebraic functions is as follows:

1. Select the material property you want to define using one or more algebraic functions. For the example presented in *Example* (p. 202), select **Heat capacity per unit mass**.
2. Click the **PMAT** button at the top of the ANSYS POLYDATA menu. The button will change to **PMAT [on]** to indicate that the PMAT feature is enabled.
3. Specify the related parameters for the selected property. For example, enter the value of the coefficient **a** for the heat capacity. The value that you specify will be used as V in *Equation 9–2* (p. 202).

   When you accept the new value for the parameter, ANSYS POLYDATA will bring you to the **Functional dependence of...** menu for the specified parameter.
4. In the **Functional dependence of...** menu, specify the type of functional dependence. The choices are as follows:

   - **Reciprocal product**:

$$F\,(f_1\,,\,f_2\,,\;\ldots\,)\;=\frac{1}{\prod_{i\,=\,1}^{n} f_i} \tag{9–4}$$

- **Reciprocal sum**:

$$F\,(f_1\,,\,f_2\,,\;\ldots\,)\;=\frac{1}{\sum\limits_{i\,=\,1}^{n}\;f_i} \tag{9–5}$$

- **Product**:

$$F\,(f_1\,,\,f_2\,,\;\ldots\,)\;=\prod_{i\,=\,1}^{n} f_i \tag{9–6}$$

- **Sum**:

$$F\,(f_1\,,\,f_2\,,\;\ldots\,)\;=\sum_{i\,=\,1}^{n}\;f_i \tag{9–7}$$

- **f's-independent**

$$F\left(f_1\right)\;=1 \tag{9–8}$$

For the example in *Example* (p. 202), select **Sum**.

5. Define each function f_i.

   a. Create a function.

   Create a new function

   b. Select the function name (**f1(...)**, by default, for the first one you define).

   f1(...)

   c. Select the appropriate type of function from the list of choices.

   d. Enter the relevant coefficients at the bottom of the menu.

   e. Repeat these steps for each function f_i.

6. When you are done specifying properties with PMAT dependence, click the **PMAT** button again to turn it off. The button will change to **PMAT [off]** to indicate that access to PMAT has been disabled, although the defined function still exists.

You can use the **LSPM** button to list the field dependences that are currently defined. If you need to modify a dependence that is listed, you can simply select it in the list and make the modifications; you do not need to return to the menu where you originally defined the function.

9.2.3. Compressible Flows

A PMAT dependence can also be applied to density to model a compressible flow. For such flows with variable overall density, ANSYS POLYFLOW does not solve for the volume but rather solves the mass conservation equation. The overall density is either treated as an additional variable or algebraically substituted.

The treatment used depends on the nature of the term to be evaluated: if a gradient must be computed, then the density is considered as a variable; if an algebraic relationship must be computed, then the density is simply substituted. The resulting formulation is a mixed method with an increase in the number of unknown variables.

This solution procedure, however, has several advantages. The implementation is rather easy and, most importantly, it allows you to model an equation of state of any kind while still keeping the Newton-Raphson rate of convergence.

When the density varies, velocity is no longer divergence-free, as it is for incompressible flows. Since mass is conserved, the flow kinematics as well as the volume of the material may change dramatically.

For example, consider a problem in which the density decreases as chemical reactions proceed. In a confined geometry, the fluid will accelerate; in a free surface problem, large swelling ratios will be observed, and it may be useful to consider a gradual solution strategy using an evolution technique to introduce this nonlinearity.

9.3. Reading and Writing Material Data Files

In cases where you are planning to solve several problems that use the same material, you can define the material properties for the first such problem and then reuse them in the other problems. This saves you time, because you will not need to repeat the definition of the same properties for each subsequent problem.

To save your material property definitions to a file, select the **Save in a Material Data File** item in the **Material data** menu.

Save in a Material Data File

ANSYS POLYDATA will ask you for the name of the file where the data are to be saved.

To read the material property definitions into a later ANSYS POLYDATA session, select the **Read an Old Material Data File** item in the **Material data** menu.

Read an Old Material Data File

ANSYS POLYDATA will ask you for the name of the file where the material data are stored, and then read the material property definitions from the specified file. You can then modify them within ANSYS POLYDATA, if necessary.

9.4. Curve Fitting for Material Properties

The ANSYS POLYFLOW package includes a preprocessor called ANSYS POLYMAT, which allows you to determine the material property definitions that correspond to experimental data (or other external data) that you have for the material. ANSYS POLYMAT analyzes the data you provide, and performs curve fitting to obtain the appropriate inputs to ANSYS POLYDATA. To perform curve fitting on your data, select the **Curve Fitting** item in the **Material data** menu in ANSYS POLYDATA.

Curve Fitting

ANSYS POLYDATA will launch the ANSYS POLYMAT program for you. See the separate ANSYS POLYMAT User's Guide for information about using ANSYS POLYMAT.

You can also start ANSYS POLYMAT from ANSYS POLYMAN, as described in *Starting the Programs* (p. 40).

9.5. Using Material Data from the CAMPUS Database

Material properties from the CAMPUS material database can be imported into ANSYS POLYMAT, the preprocessor for material property definition and curve fitting. See the separate ANSYS POLYMAT User's Guide for details.

9.6. Using Material Data from the ANSYS POLYFLOW Databases

Material properties from the material database `Material_Data` are provided with the ANSYS POLYFLOW package in the following directory:

- For Linux:

  `path/ansys_inc/v140/polyflow/polyflow14.0.x/Material_Data`

- For Windows:

  `path\ANSYS Inc\v140\polyflow\polyflow14.0.x\Material_Data`

where *`path`* is the directory in which ANSYS POLYFLOW has been installed and x represents the appropriate number for the release (e.g., `0` for `polyflow14.0.0`).

Within the `Material_Data` directory, you will find five subdirectories: `Miscellaneous`, `Shell_Materials`, `Extrusion`, `BlowMolding`, and `BlowMolding_viscoelastic`. Further information about these subdirectories is provided in the sections that follow.

All parameters in the material data files are provided in the SI system of units or MKS, while temperatures are provided in Kelvin. Upon selection and reading of such a material data file, the program asks whether you want to define another system of units; when applicable, a units conversion is performed in ANSYS POLYDATA based on your specifications (see *Converting to a New Unit System* (p. 129)).

Note that ANSYS POLYDATA will always open the material database from the installation directory when reading an old material data file. To access a material data file available in your current working directory, simply erase the directory path from the browser filter. You may also navigate via the browser.

9.6.1. The `Miscellaneous` and `Shell_Materials` Directories

The `Miscellaneous` directory contains basic information for fluids encountered in the processing industry, ranging from water to metal. A file from the `Miscellaneous` material database can be imported into ANSYS POLYFLOW using the **Read an Old Material Data File** menu item, as described in *Reading and Writing Material Data Files* (p. 204). For example, the file HDPE.493.mat incorporates typical material data for modelling the viscosity of an HDPE at a temperature of 493 K ($220^{\circ}C$).

The Shell_Materials directory contains material data that is appropriate when defining a blow molding or thermoforming application on a membrane element, where the rheology is described by means of an integral KBKZ model (see *3D Viscoelastic Blow Molding Simulations* (p. 384)).

9.6.2. The Extrusion, BlowMolding, and BlowMolding_viscoelastic Directories

The Extrusion, BlowMolding, and BlowMolding_viscoelastic directories contain material data files which incorporate generic or relevant rheological and thermal data for common materials, such as LDPE, HDPE, PVC, and PS. Each of these directories is specialized for the application and model suggested in the corresponding directory name. More precisely, the Extrusion directory contains material data files to be used for the simulation of extrusion flows in which a generalized Newtonian fluid model is invoked. The BlowMolding directory contains material data files to be used for the simulation of blow molding cases in which a Newtonian fluid model is invoked, for 2D and 3D geometries, as well as for shell simulations. The BlowMolding_viscoelastic directory contains material data files to be used for the simulation of blow molding cases in which an integral KBKZ fluid model is invoked for shell simulations.

In each of these directories, you can find data files for common materials. The file names are designed to allow easy identification, and to conform to a structure or nomenclature. The name structure consists of the following: the process being simulated, the generic name of the material, the thermal attribute, and the temperature. The thermal attribute conveys whether the data is for isothermal or nonisothermal conditions; in the latter case, additional thermal parameters are provided in the file. For isothermal data, the selected temperature is used in the name; for nonisothermal data, it is the reference temperature that is used in the name.

For example, in the Extrusion directory, you will find files like Extrusion_LLDPE_isoth_463K.mat and Extrusion_LLDPE_nonisoth_463K.mat; these files contain relevant data for LLDPE, which can be used for the simulation of an extrusion process under isothermal and nonisothermal conditions, respectively. For the isothermal condition, the selected temperature is 463 K, while for the nonisothermal condition, the rheological data is provided at a reference temperature of 463 K, and incorporates a temperature dependence.

Chapter 10: Generalized Newtonian Flow

ANSYS POLYFLOW can be used to solve several types of flows, ranging from Newtonian inelastic to non-Newtonian viscoelastic. Several shear-rate-dependent and temperature-dependent viscosity laws are supplied.

This chapter describes generalized Newtonian flows, and provides information on how to model them with ANSYS POLYFLOW.

10.1. Introduction

In addition to the viscoelastic flows discussed in *Viscoelastic Flows* (p. 225), ANSYS POLYFLOW can also be used to model generalized Newtonian flows. The generalized Newtonian category of flow includes both Newtonian flows and inelastic non-Newtonian flows. ANSYS POLYFLOW has a number of modeling options available for simulations of these types of flows. Several shear-rate-dependent viscosity laws are supplied, and a temperature-dependent or constant viscosity can also be modeled.

As is well known, the range of validity of the Newtonian constitutive equation is limited to low-molecular-weight, homogeneous liquids. The flow phenomena observed with polymeric fluids, for example, cannot be predicted by the classical Navier-Stokes equations. Non-Newtonian behavior has many facets. Among them are the shear-rate dependence of the shear viscosity, the presence of normal stresses in viscometric flows, high resistance to elongational deformation, and memory effects associated with the elasticity of the fluid. Non-Newtonian inelastic flows exhibit a shear-rate dependence of the shear viscosity, but the other phenomena are characteristics of viscoelastic flows.

ANSYS POLYFLOW offers a wide variety of constitutive models for non-Newtonian inelastic fluids. It is, however, essential to be aware that none of these models leads to realistic predictions in *all* types of deformation of any particular non-Newtonian fluid. In many cases, the use of non-Newtonian inelastic models will capture the physics of the flow with sufficient accuracy. The only modeling issue then is to identify a suitable nonlinear viscosity function that will fit available viscosity data.

For background information about generalized Newtonian flow, see [*1*] (p. 715) or [*2*] (p. 715) or any similar handbook or textbook on rheology.

10.2. Theory and Equations

10.2.1. Basic Equations

For a generalized Newtonian flow, ANSYS POLYFLOW solves the momentum equations, the incompressibility equation, and (for nonisothermal flows) the energy equation. The form of the momentum equations is

$$-\nabla p + \nabla \cdot \mathbf{T} + \mathbf{f} = \rho \mathbf{a} \tag{10–1}$$

where $p =$ pressure

$\mathbf{T} =$ extra-stress tensor

$\mathbf{f} =$ volume force

$\rho =$ density

$\mathbf{a} =$ acceleration

The incompressibility equation is

$$\nabla \cdot \mathbf{v} = 0 \tag{10–2}$$

where $\mathbf{v}$ is velocity.

The energy equation is presented as *Equation 13–4* (p. 284) in *Theory* (p. 283).

For a generalized Newtonian fluid,

$$\mathbf{T} = 2\eta \mathbf{D} \tag{10–3}$$

where $\mathbf{D}$ is the rate-of-deformation tensor and η can be a function of local shear rate $\dot{\gamma}$, temperature T, or both. The local shear rate is defined as

$$\dot{\gamma} = \sqrt{2\mathrm{tr}\,(\mathbf{D}^2)} \tag{10–4}$$

In a simple shear flow, $\dot{\gamma}$ reduces to the velocity gradient.

When nonisothermal flow is modeled, ANSYS POLYFLOW calculates the temperature, velocity, and pressure fields simultaneously (i.e., fully coupled, unless otherwise specified by a change in the default numerical parameters).

10.2.2. Shear-Rate-Dependent Viscosity Laws

Several viscosity laws are available for generalized Newtonian flows. The isothermal viscosity laws will be presented in this section, and *Temperature-Dependent Viscosity Laws* (p. 212) describes their extension to include temperature dependence in nonisothermal flows.

10.2.2.1. Constant

For Newtonian fluids, a constant viscosity

$$\eta = \eta_0 \tag{10–5}$$

can be specified. η_0 is referred to as the Newtonian or zero-shear-rate viscosity.

10.2.2.2. Power Law

The power law for viscosity is

$$\eta = K\ (\lambda \dot{\gamma})^{\ n-1} \tag{10–6}$$

where K is the consistency factor, λ is the natural time or the reciprocal of a reference shear rate, and n is the power-law index, which is a property of a given material.

The power law is commonly used to describe the viscous behavior of polymeric materials, such as polyethylene, with shear rates ranging over 2 to 3 decades. However, it fails to describe the behavior at low shear rates. If the behavior at low shear rates needs to be fitted as well, the Bird-Carreau or Cross law will capture the plateau zone of the viscosity curve for low shear rates better than the power law.

10.2.2.3. Bird-Carreau Law

The Bird-Carreau law for viscosity is

$$\eta = \eta_{\infty} + \left(\eta_{0} - \eta_{\infty}\right)\ (1 + \lambda^{2} \dot{\gamma}^{2})^{\frac{n-1}{2}} \tag{10–7}$$

where

η_{∞} = infinite-shear-rate viscosity

η_{0} = zero-shear-rate viscosity

λ = natural time (i.e., inverse of the shear rate at which the fluid changes from Newtonian to power-law behavior)

n = power-law index

The Bird-Carreau law is commonly used when it is necessary to describe the low-shear-rate behavior of the viscosity. It differs from the Cross law primarily in the curvature of the viscosity curve in the vicinity of the transition between the plateau zone and the power law behavior.

10.2.2.4. Cross Law

The Cross law for viscosity is

$$\eta = \frac{\eta_{0}}{1 + (\lambda \dot{\gamma})^{\ m}} \tag{10–8}$$

where

η_{0} = zero-shear-rate viscosity

λ = natural time (i.e., inverse of the shear rate at which the fluid changes from Newtonian to power-law behavior)

m = Cross-law index (= $1 - n$ for large shear rates)

Like the Bird-Carreau law, the Cross law is commonly used when it is necessary to describe the low-shear-rate behavior of the viscosity. It differs from the Bird-Carreau law primarily in the curvature of the viscosity curve in the vicinity of the transition between the plateau zone and the power law behavior.

10.2.2.5. Modified Cross Law

A modified Cross law for viscosity is also available:

$$\eta = \frac{\eta_0}{(1 + \lambda \dot{\gamma})^{m}} \tag{10–9}$$

This law can be considered a special case of the Carreau-Yasuda viscosity law (*Equation 10–15* (p. 212)), where the exponent a has a value of 1.

10.2.2.6. Bingham Law

The Bingham law for viscosity is

$$\eta = \begin{cases} \eta_0 + \dfrac{\tau_0}{\dot{\gamma}}, & \dot{\gamma} \geq \dot{\gamma}_c \\ \eta_0 + \tau_0 \dfrac{\left(2 - \dfrac{\dot{\gamma}}{\dot{\gamma}_c}\right)}{\dot{\gamma}_c}, & \dot{\gamma} < \dot{\gamma}_c \end{cases} \tag{10–10}$$

where τ_0 is the yield stress and $\dot{\gamma}_c$ is the critical shear rate, beyond which Bingham's constitutive equation is applied. For shear rates less than $\dot{\gamma}_c$, the behavior of the fluid is normalized in order to guarantee appropriate continuity properties in the viscosity curve.

The Bingham law is commonly used to describe materials such as concrete, mud, dough, and toothpaste, for which a constant viscosity after a critical shear stress is a reasonable assumption, typically at rather low shear rates.

10.2.2.7. Modified Bingham Law

A modified Bingham law for viscosity is also available:

$$\eta = \eta_0 + \tau_0 \left(\frac{1 - \exp\left(-m\dot{\gamma}\right)}{\dot{\gamma}} \right) \tag{10–11}$$

where $m = 3/\dot{\gamma}_c$.

Compared to the standard Bingham law, the modified Bingham law is an analytic expression, which means that it may be easier for ANSYS POLYFLOW to calculate, leading to a more stable solution. The value $m = 3/\dot{\gamma}_c$ has been selected so that the standard and modified Bingham laws exhibit the same behavior above the critical shear rate, $\dot{\gamma}_c$.

10.2.2.8. Herschel-Bulkley Law

The Herschel-Bulkley law for viscosity is

$$\eta = \begin{cases} \dfrac{\tau_0}{\dot{\gamma}} + K\left(\dfrac{\dot{\gamma}}{\dot{\gamma}_c}\right)^{n-1}, & \dot{\gamma} > \dot{\gamma}_c \\ \dfrac{\tau_0\left(2 - \dfrac{\dot{\gamma}}{\dot{\gamma}_c}\right)}{\dot{\gamma}_c} + K\left[(2-n) + (n-1)\dfrac{\dot{\gamma}}{\dot{\gamma}_c}\right], & \dot{\gamma} \le \dot{\gamma}_c \end{cases} \tag{10–12}$$

where τ_0 is the yield stress, $\dot{\gamma}_c$ is the critical shear rate, K is the consistency factor, and n is the power-law index.

Like the Bingham law, the Herschel-Bulkley law is commonly used to describe materials such as concrete, mud, dough, and toothpaste, for which a power-law viscosity after a critical shear stress is a reasonable assumption. In addition to the transition behavior between a flow and no-flow regime, the Herschel-Bulkley law exhibits a shear-thinning behavior that the Bingham law does not.

10.2.2.9. Modified Herschel-Bulkley Law

A modified Herschel-Bulkley law is also available:

$$\eta = \tau_0\left(\frac{1 - \exp\left(\dfrac{-3\dot{\gamma}}{\dot{\gamma}_c}\right)}{\dot{\gamma}}\right) + K\left(\frac{\dot{\gamma}}{\dot{\gamma}_c}\right)^{n-1} \tag{10–13}$$

Compared to the standard Herschel-Bulkley law, the modified Herschel-Bulkley law is an analytic expression, which means that it may be easier for ANSYS POLYFLOW to calculate, leading to a more stable solution. The integer value 3 that appears in the argument of the exponential term has been selected so that the standard and modified Herschel-Bulkley laws exhibit the same behavior above the critical shear rate, $\dot{\gamma}_c$.

10.2.2.10. Log-Log Law

The log-log law for viscosity is

$$\eta = \eta_0 10^{a_0 + a_1[\log(\dot{\gamma}/\dot{\gamma}_c)] + a_{11}[\log(\dot{\gamma}/\dot{\gamma}_c)^2]} \tag{10–14}$$

where η_0 is the zero-shear-rate viscosity and a_0, a_1, and a_{11} are the coefficients of the polynomial expression.

This viscosity law is purely empirical, but sometimes provides a better fit to experimental data than the others. Nevertheless, you should pay special attention to the coefficients you specify for the log-log law, as detailed below.

The function is a parabola in the $(\log(\dot{\gamma}), \log(\eta))$ space. Depending on the values of the polynomial coefficients, the viscosity may decrease as the shear rate approaches zero, which does not reflect physical behavior. Moreover, for high shear rates, the slope of the curve may be less than -1, which is also not physical. When you are using the log-log law, you must therefore ensure that the range of shear rates in your application lies within the range of physically-acceptable shear rates for the law. This is accomplished by careful specification of the polynomial coefficients.

Important

Note that, for nonisothermal flows using the log-log law, the mixed-dependence law (described in *Mixed-Dependence Law* (p. 215)) must be used for the thermal dependence of the viscosity.

10.2.2.11. Carreau-Yasuda Law

The Carreau-Yasuda law for viscosity is

$$\eta = \eta_\infty + \left(\eta_0 - \eta_\infty\right) [1 + (\lambda\dot{\gamma})^a]^{\frac{n-1}{a}} \qquad \textbf{(10–15)}$$

where

η_0 = zero-shear-rate viscosity

η_∞ = infinite-shear-rate viscosity

λ = natural time (i.e., inverse of the shear rate at which the fluid changes from Newtonian to power-law behavior)

a = index that controls the transition from the Newtonian plateau to the power-law region

n = power-law index

The Carreau-Yasuda law is a slight variation on the Bird-Carreau law (*Equation 10–7* (p. 209)). The addition of the exponent a allows for control of the transition from the Newtonian plateau to the power-law region. A low value (a < 1) lengthens the transition, and a high value (a > 1) results in an abrupt transition.

10.2.3. Temperature-Dependent Viscosity Laws

If the flow is nonisothermal, the temperature dependence of the viscosity must be taken into account along with the shear-rate dependence. The viscosity law can be factorized as follows:

$$\eta = H(T)\,\eta_0(\dot{\gamma}) \qquad \textbf{(10–16)}$$

where $H(T)$ is the Arrhenius law (or one of the other available laws) and $\eta_0(\dot{\gamma})$ is the viscosity law at some reference temperature T_α (as computed by one of the shear-rate-dependent laws described above).

10.2.3.1. Arrhenius Law

The Arrhenius law is given as

$$H(T) = \exp\left[\alpha\left(\frac{1}{T - T_0} - \frac{1}{T_\alpha - T_0}\right)\right] \tag{10–17}$$

where α is the ratio of the activation energy to the thermodynamic constant, and T_α is a reference temperature for which $H(T) = 1$. The temperature shift T_0 is set to 0 by default, and corresponds to the lowest temperature that is thermodynamically acceptable. Therefore T and T_α are absolute temperatures. They can also be defined relative to a non-absolute temperature scale, in which case T_0 corresponds to the absolute zero temperature in the current temperature scale.

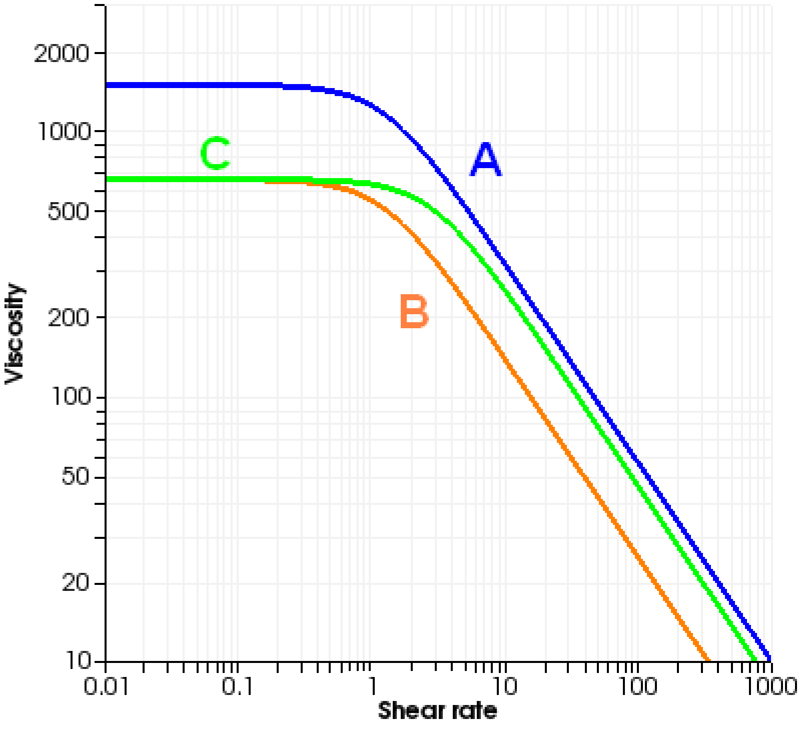
Two versions of the Arrhenius law are available: one based on shear rate and one based on shear stress, as described below. Both methods provide similar, although not identical, results for most polymers.

10.2.3.2. Arrhenius Shear Rate vs. Arrhenius Shear Stress

Not all polymers strictly follow *Equation 10–16* (p. 212). While this requires detailed measurements of the viscosity as a function of the temperature, most polymers follow a viscosity-temperature dependence that is slightly different from *Equation 10–16* (p. 212).

Consider a typical viscosity curve such as the one shown in *Figure 10.1* (p. 213). Curve A represents the viscosity at a reference temperature, say 180°. This viscosity curve, which has been modeled using a Bird-Carreau law, presents a plateau zone for low shear rates.

Figure 10.1 Typical Viscosity Curve



If the polymer were to follow *Equation 10–16* (p. 212), curve A would translate at a different temperature, say 200°, into curve B. This is the behavior modeled with either the Arrhenius or the approximate Arrhenius laws. However, sometimes the viscosity at 200° follows curve C; that is, not only is the absolute value of the viscosity decreased, but also the point of departure from the constant-viscosity regime moves toward the right in the diagram. This is precisely what the "shear stress" version of the temperature-dependence laws models, using the following equation:

$$\eta = H(T)\,\eta_0\,(H(T)\,\dot{\gamma}) \tag{10–18}$$

The law of *Equation 10–18* (p. 214) corresponds to a vertical shift of the viscosity curve in a viscosity-shear-stress diagram, so it is referred to as the Arrhenius shear-stress law.

Note that a power-law model does not reflect the difference between Arrhenius shear rate and Arrhenius shear stress because of the absence of a transition zone.

10.2.3.3. Approximate Arrhenius Law

The approximate Arrhenius law is written as follows:

$$H(T) = \exp\,[-\alpha\,(T - T_\alpha)\,] \tag{10–19}$$

The behavior described by *Equation 10–19* (p. 214) is similar to that described by *Equation 10–17* (p. 213) in the neighborhood of T_α. *Equation 10–19* (p. 214) is valid as long as the temperature difference $T - T_\alpha$ is not too large. As for the Arrhenius law, two versions are available: one based on shear rate and one based on shear stress, as described above.

10.2.3.4. Fulcher Law

Another definition for $H(T)$ comes from the Fulcher law [*11*] (p. 715):

$$H(T) = 10^{\,-f_1 + \frac{f_2}{T - f_3}} \tag{10–20}$$

where f_1, f_2, and f_3 are the Fulcher constants. The Fulcher law is used mainly for glass.

10.2.3.5. WLF Law

The Williams-Landel-Ferry (WLF) equation is a temperature-dependent viscosity law that fits experimental data better than the Arrhenius law for a wide range of temperatures, especially close to the glass transition temperature:

$$\ln\,(H(T)) = \frac{c_1\,(T_r - T_a)}{c_2 + T_r - T_a} - \frac{c_1\,(T - T_a)}{c_2 + T - T_a} \tag{10–21}$$

where c_1 and c_2 are the WLF constants, and T_r and T_a are reference temperatures.

10.2.3.6. WLF Shear-Stress Law

The WLF law described above is based on shear rate. As for the Arrhenius law, there is also a version of the WLF law based on shear stress. In this version, the viscosity η is computed from *Equation 10–18* (p. 214), with $H(T)$ computed from the WLF law, *Equation 10–21* (p. 214). As for the Arrhenius shear-stress law, an increase in temperature will result in a shifting of the viscosity curve downward and to the right.

10.2.3.7. Mixed-Dependence Law

For the mixed-dependence law (which can be used only in conjunction with the log-log law for shear-rate dependence), the function η is written as

$$\eta = \eta_l H(T, \dot{\gamma}) \tag{10–22}$$

where η_l is computed from the log-log law (*Equation 10–14* (p. 211)) and

$$H(T, \dot{\gamma}) = 10^{a_2 (T - T_0) + a_{22} (T - T_0)^2 + a_{12} (T - T_0) \log(\dot{\gamma}/\dot{\gamma}_c)} \tag{10–23}$$

In this equation, a_2, a_{22}, and a_{12} are the coefficients of the polynomial expression, and T_0 is the lowest temperature that is thermodynamically acceptable, with respect to the current temperature scale. Typically, if the units for temperature are Kelvin, T_0 will be 0; if the units for temperature are Celsius, T_0 will be -273.15.

10.3. Problem Setup

10.3.1. General Procedure

The basic steps for setting up a generalized Newtonian flow are as follows:

1. Create a sub-task for the generalized Newtonian flow problem.

   Create a sub-task

   a. Select the appropriate problem type from the **Create a sub-task** menu.

      Generalized Newtonian isothermal flow problem

      or

      Generalized Newtonian non-isothermal flow problem

   b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. Define the material properties.

   Material data

   a. Define the shear-rate dependence of the viscosity.

Shear-rate dependence of viscosity

Select the viscosity law you want to use, and specify the relevant parameters:

- Select **Constant viscosity** to specify a constant (Newtonian) viscosity. The only input required is the value of η_0 in *Equation 10–5* (p. 208) (referred to as **fac** in ANSYS POLYDATA).
- Select **Bird-Carreau law** to use the Bird-Carreau viscosity law. The inputs for this law are η_0 (referred to as **fac** in ANSYS POLYDATA), λ (referred to as **tnat**), n (**expo**), and η_∞ (**facinf**) in *Equation 10–7* (p. 209).
- Select **Power law** to use the power law for viscosity. The inputs for this law are K (referred to as **fac** in ANSYS POLYDATA), n (**expo**), and λ (**tnat**) in *Equation 10–6* (p. 209). The default value for **tnat** is 1, which results in the classical expression for the power law.
- Select **Bingham law** to use the Bingham viscosity law. The inputs for this law are η_0 (referred to as **fac** in ANSYS POLYDATA), τ_0 (referred to as **ystr**), and $\dot{\gamma}_c$ (**gcrit**) in *Equation 10–10* (p. 210).
- Select **Herschel-Bulkley law** to use the Herschel-Bulkley viscosity law. The inputs for this law are τ_0 (referred to as **fac1** in ANSYS POLYDATA), K (referred to as **fac2**), n (**expo**), and $\dot{\gamma}_c$ (**gcrit**) in *Equation 10–12* (p. 211).
- Select **Cross law** to use the Cross law for viscosity. The inputs for this law are η_0 (referred to as **fac** in ANSYS POLYDATA), λ (referred to as **tnat**), and m (**expom**) in *Equation 10–8* (p. 209).
- Select **Log-Log law** to use the log-log law for viscosity. The inputs for this law are η_0 (referred to as **fac** in ANSYS POLYDATA), $\dot{\gamma}_c$ (referred to as **gcrit**), and the coefficients a_0, a_1, and a_{11} in *Equation 10–14* (p. 211).

  Use the mixed-dependence law for the temperature dependence of viscosity if you choose to use the log-log law for a nonisothermal flow.
- Select **modified Bingham law** to use the modified Bingham law for viscosity. The inputs for this law are η_0 (referred to as **fac** in ANSYS POLYDATA), τ_0 (referred to as **ystr**), and $\dot{\gamma}_c$ (**gcrit**) in *Equation 10–11* (p. 210).
- Select **modified Herschel-Bulkley law** to use the modified Herschel-Bulkley law for viscosity. The inputs for this law are τ_0 (referred to as **fac1** in ANSYS POLYDATA), K (referred to as **fac2**), n (**expo**), and $\dot{\gamma}_c$ (**gcrit**) in *Equation 10–13* (p. 211).
- Select **Carreau-Yasuda law** to use the Carreau-Yasuda law for viscosity. The inputs for this law are η_0 (referred to as **fac** in ANSYS POLYDATA), λ (referred to as **tnat**), η_∞ (**facinf**), n (**expo**), and a (**expoa**) in *Equation 10–15* (p. 212).
- Select **modified Cross law** to use the modified Cross law for viscosity. The inputs for this law are η_0 (referred to as **fac** in ANSYS POLYDATA), λ (referred to as **tnat**), and m (**expom**) in *Equation 10–9* (p. 210).

b. For nonisothermal flows, define the temperature dependence of the viscosity.

Temperature dependence of viscosity

Select the law you want to use, and specify the relevant parameters:

- Select **No temperature dependence** if you do not want the viscosity to be temperature-dependent (i.e., $H(T) = 1$ in *Equation 10–16* (p. 212)). No further inputs are required.
- Select **Arrhenius approximate law** to use *Equation 10–19* (p. 214) with temperatures at constant shear rate. The inputs for this law are α and T_α.
- Select **Arrhenius law** to use *Equation 10–17* (p. 213) with temperatures at constant shear rate. The inputs for this law are α, T_α, and T_0. To specify T_α as an absolute temperature, set the temperature shift T_0 to 0.

  Otherwise, set T_0 to the appropriate temperature shift and then specify T_α relative to T_0.
- Select **Arrhenius approximate shear stress law** to use *Equation 10–19* (p. 214) with temperatures at constant shear stress. The inputs are the same as for **Arrhenius approximate law**.
- Select **Arrhenius shear stress law** to use *Equation 10–17* (p. 213) with temperatures at constant shear stress. The inputs are the same as for **Arrhenius law**.
- Select **Mixed dependence** to use the mixed-dependence law (*Equation 10–23* (p. 215)). The inputs for this law are a_2, a_{22}, a_{12}, and T_0.

Important

The mixed-dependence law is available only if you have selected the log-log law for shear-rate dependence. It is the only law for temperature dependence that can be used with the log-log law.

- Select **Fulcher dependence** to use *Equation 10–20* (p. 214). The inputs for this law are f_1, f_2, and f_3.
- Select **WLF dependence** to use the WLF equation (*Equation 10–21* (p. 214)) based on shear rate (i.e., with only a vertical shift in the η- $\dot{\gamma}$ diagram). The inputs for this law are c_1, c_2, T_a, and $(T_r - T_a)$. The default values for c_1 and c_2 are appropriate for many polymer processing applications.
- Select **WLF shear stress dependence** to use the WLF equation (*Equation 10–21* (p. 214)) based on shear stress (i.e., with a vertical and horizontal shift in the η- $\dot{\gamma}$ diagram). The inputs are the same as for **WLF dependence**.

c. If inertia, heat convection, or natural convection are to be taken into account in the calculation, define the density, inertia terms, and gravity. (By default, density is equal to zero, inertia terms are neglected, and gravitational acceleration is equal to zero.) For many processing applications, the Reynolds number is so low that inertia terms can safely be neglected. Even in the absence of inertia terms, however, it may be necessary to assign nonzero values for density and gravitational acceleration, since they influence heat capacity and buoyancy forces, respectively.

  i. Set the density.

  Density

  Select **Modification of density** and enter a new value.

  ii. Enable the inertia terms in the momentum equations.

Inertia terms

Select **Inertia will be taken into account** to enable the inertia terms. To disregard the inertia terms, you can select **Inertia will be neglected**, the default setting. The option to take inertia into account will not be available if the density is equal to zero. You will need to specify a nonzero density first, in order to enable the inertia terms.

iii. Set the gravitational acceleration.

Gravity

Select **Modification of gx** and set the gravitational acceleration in the x direction. Repeat for the y and z components.

d. Set any other appropriate material properties (such as thermal conductivity, heat capacity, or thermal expansion coefficient). For nonisothermal flows, for example, see *Problem Setup* (p. 286) for instructions.

See *Using Evolution to Compute Generalized Newtonian Flow* (p. 222) for suggestions about using evolution to define material properties.

4. Define the flow boundary conditions.

Flow boundary conditions

See *Boundary Conditions* (p. 173) for details. See also *Using Evolution to Compute Generalized Newtonian Flow* (p. 222) for suggestions about using evolution to define boundary conditions.

5. For nonisothermal flows, define the thermal boundary conditions.

Thermal boundary conditions

See *Boundary Conditions* (p. 173) and *Problem Setup* (p. 286) for details.

6. (optional) Modify the interpolation schemes used for the momentum and incompressibility equations.

Interpolation

See *Controlling the Interpolation* (p. 218) for details.

10.3.2. Controlling the Interpolation

The incompressibility equation (*Equation 10–2* (p. 208)) requires the divergence of the velocity field to vanish. In the finite-element velocity-pressure formulation, it is not possible to exactly satisfy the incompressibility constraint. ANSYS POLYFLOW relies on a mixed finite-element method, which requires the satisfaction of some conditions on the polynomial degrees for the velocity and pressure representations.

10.3.2.1. Interpolation for Nonisothermal Flows

The hybrid integration rule is recommended for transient nonisothermal flow or heat conduction problems involving thermal shocks (e.g., a glass pressing problem). Indeed, in cases that involve thermal shocks, this technique has been found to remove the numerical wiggles of the temperature field. The selection of the hybrid integration rule is computationally more expensive than the standard rule. It is slightly more expensive in terms of CPU, while there is no extra cost regarding memory. If no thermal shock is present, the standard and hybrid integration rules provide essentially the same results.

In order to select the hybrid integration rule, you need to perform the following steps:

1. Select **Advanced options** in the **Interpolation** menu.
2. Change the integration rule from the **standard** method (the default) to the **hybrid** integration rule.

The hybrid method is only available for the following:

- **Heat conduction problem**
- **Generalized Newtonian non-isothermal flow problem**
- nonisothermal molds

> **Note**
>
> The hybrid method is not available if the interpolation for temperature is of the type **4 x 4 element for temperature**.

10.3.2.2. Finite-Element Interpolation for 2D Models

For 2D (and 2D 1/2) flows, the default method involves a quadratic representation of the velocity field and a linear-continuous representation of the pressure field. Such a representation is valid for quadrilateral and triangular elements, and has been used extensively in fluid mechanics problems.

In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **Quadratic velocities, linear pressure** menu item, which is enabled by default.

The quadratic-linear representation, however, is characterized by a relatively low number of incompressibility constraints in comparison with the momentum equations. A more accurate method can be obtained with a discontinuous representation for the pressure, which then becomes a first-order polynomial in each element. This quadratic-linear-discontinuous representation is sometimes used in free surface or thermal convection problems. In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **Quadratic velocities, linear discontinuous pressure** menu item.

A linear representation of the velocity field with a constant pressure in each element is also available, although this method does not satisfy the LBB criterion [*17*] (p. 715) (or min-max condition). To avoid spurious pressure modes, the pressure is stabilized by an artificial compressibility term. The linear representation is a computationally inexpensive method that can give satisfactory results with appropriate boundary conditions (mostly of the Neumann type). In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **Linear velocities, constant pressure** menu item.

10.3.2.3. Finite-Element Interpolation for 3D Models

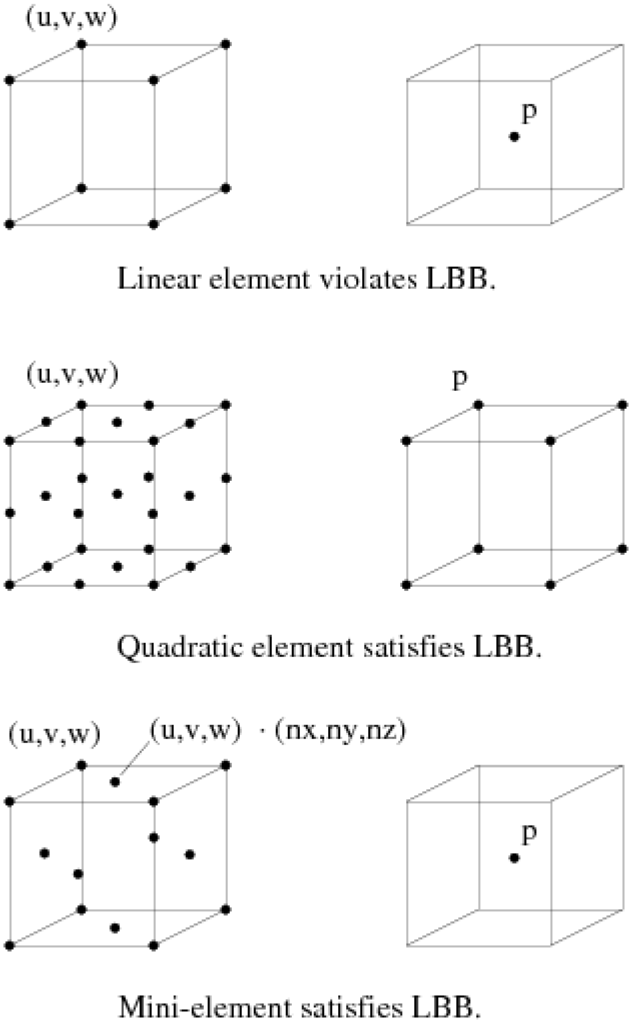
For 3D flows, the default representation is the mini-element developed by Fortin [*10*] (p. 715). The mini-element is implemented as a correction to the linear element: a degree of freedom is introduced on the midfaces of the elements, but only for the normal part of the velocity vector, as indicated in *Figure 10.2* (p. 220). This minimizes the cost of the correction and solves all problems linked to incompressibility (LBB condition).

The cost of the mini-element is similar to that of the simpler linear element, but the mini-element scheme is more accurate. The mini-element method is therefore recommended for most 3D flows, since it provides reasonably accurate results at lower cost than the quadratic methods. In the **Interpolation** menu in ANSYS POLYDATA, the mini-element option corresponds to the **Mini-element for velocities, constant pressure** menu item.

The major drawback of the mini-element is its low pressure accuracy due to the piecewise-constant finite-element pressure representation. This can cause problems for some meshes that exhibit a large aspect ratio (i.e., meshes for which the dimension in one direction is much smaller than in another direction, so that a strong pressure gradient develops). The problem becomes worse if the mesh includes elements that are strongly deformed (i.e., elements whose opposite sides are of different dimensions).

In order to address this problem without imposing the use of the CPU-intensive quadratic approximation (described below), a combination of the mini-element with a linear pressure is provided. In the **Interpolation** menu in ANSYS POLYDATA, the mini-element with linear pressure option corresponds to the **Mini-element for velocities, linear pressure** menu item.

Figure 10.2 Implementation of the Mini-Element



Linear element violates LBB.

Quadratic element satisfies LBB.

Mini-element satisfies LBB.

This option greatly improves the pressure representation in cases where the accuracy of the standard mini-element is not good. The mini-element with linear pressure uses “bubble" velocity shape functions at element centers. Even with this additional bubble shape function at the element centers, there are rare boundary condition cases where the element violates incompressibility conditions. In order to avoid solution difficulties, the incompressibility equation is stabilized with an artificial compressibility that is small enough to not be a problem in practical flow situations.

The quadratic-linear representation (**Quadratic velocities, linear pressure**) is the most accurate and the most expensive of all representations, and the quadratic-linear-discontinuous representation (**Quadratic velocities, linear discontinuous pressure**) may be more accurate for free surface or thermal convection problems. However, using the quadratic representation on a dense 3D finite-element mesh can be computationally expensive and require a significant amount of memory.

The linear representation of the velocity field described previously for 2D models is also valid for 3D models. In fact, it can be especially useful for 3D models, since it has a low computational cost. It is also the only interpolation available for 3D cases involving contact (e.g., as encountered in blow molding and thermoforming).

10.3.2.4. Viscosity-Related Iterations

For a shear-rate-dependent viscosity, ANSYS POLYFLOW will, by default, use Newton-Raphson iterations to solve the nonlinearities. In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **Newton iterations on viscosity(g)** item listed in the **Current setup** at the top of the menu. (Since it is enabled by default, it will *not* appear in the menu item list itself, *only* in the **Current setup** list.)

For power-law or Bird-Carreau fluids with a power-law index less than 0.7, Picard (fixed-point) iterations are recommended instead of the default Newton-Raphson iterations.

The Picard scheme typically requires 20 iterations or more to converge to a tolerance of 10^{-3}, while the Newton-Raphson scheme often requires 3 to 4 iterations. However, the radius of the convergence disk for the Newton-Raphson scheme is small for power-law fluids. The Picard scheme provides better convergence behavior when the power-law index is low. In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **Picard iterations on viscosity(g)** menu item.

Another option for a low power-law index is to use the default Newton-Raphson technique with an evolution scheme that gradually decreases the power-law index. See *Using Evolution to Compute Generalized Newtonian Flow* (p. 222) for details.

10.3.2.5. Interpolation for Nonisothermal Flows

For nonisothermal flows, several types of interpolation are available in ANSYS POLYFLOW to calculate the temperature field. You should select the interpolation type according to the Péclet number:

$$\mathrm{Pe} = \frac{\rho c_p v L}{k} \qquad \textbf{(10–24)}$$

where v is a characteristic velocity of the flow, and L is a characteristic length of the flow. When the Péclet number is low, you should use quadratic elements (the default) for the temperature in 2D. In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **Quadratic element for temperature** menu item.

For low-Péclet-number 3D simulations where you need to reduce the CPU time and/or problem size, you can use linear elements for temperature instead of the default quadratic elements. In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **Linear element for temperature** menu item.

For moderate Péclet numbers, subdivision into 2×2 linear subelements with a Galerkin method is recommended for the energy equation. In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **2x2 element for temperature** menu item.

For flows with high Péclet numbers, you should use 4×4 subelements with a streamline-upwind Petrov-Galerkin method for the energy equation. In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **4x4 element for temperature** menu item combined with the **Upwinding in the energy equation** menu item.

This method refines the mesh for temperature without increasing the cost of the velocity-pressure calculation. Note, however, that the **4x4 element for temperature** is not available for 3D simulations. For high-Péclet-number 3D cases, the **2x2 element for temperature** is recommended instead.

The need for a special treatment of the energy equation can be seen when the temperature field exhibits wiggles (oscillations). These indicate that the mesh is too coarse. Such behavior also occurs in convection-dominated problems.

You can assign different interpolation types to different subdomains, using the **Sub-interpolation** option in the **Interpolation** menu. For example, you can use 2×2 subelements in one subdomain, and 4×4 in another. See *Problem Setup* (p. 286) for details.

10.3.2.6. Quadratic and Linear Coordinates

Linear coordinates (the default) are recommended for flows that do not involve surface tension. Quadratic coordinates should be used for flows involving surface tension (e.g., coating flows). See *Controlling the Interpolation* (p. 326) for more details.

10.3.3. Using Evolution to Compute Generalized Newtonian Flow

There are many sources of nonlinearity in generalized Newtonian flow:

- inertia terms, characterized by the Reynolds number
- heat transfer by convection, characterized by the Péclet number
- natural convection caused by buoyancy forces
- heat generation by viscous dissipation
- nonlinear dependence of viscosity upon shear rate and temperature

For these problems, it is generally impossible to reach a converged solution for the nominal values of the material parameters in one step starting from scratch. Instead, it is necessary to gradually approach these final values by incremental steps. The evolution technique described in *Evolution* (p. 517) is readily available for an automatic incrementation toward the desired solution.

10.3.3.1. Sample Applications

Below, some sample applications of the evolution technique to generalized Newtonian flow are listed:

- For problems with inertia, you can relate the density ρ to the evolution parameter S. The Reynolds number will thus be directly related to the evolution parameter.
- For problems with a high Péclet number, you can relate the thermal conductivity k to $1/S$. When S is small, k is large, and the energy equation is diffusion-dominated. When S increases, k decreases, approaching the correct value. Note that in this case, the initial value of S cannot be zero (the default value). Similarly, you can relate the heat capacity c_p to S and obtain the same effect.

  Note that this technique may not be applicable for some problems involving strong viscous heating dissipation.

- For natural convection problems, you can relate the thermal expansion coefficient (β in *Equation 13–9* (p. 285)) or the gravity to the evolution parameter S.
- For problems with viscous dissipation, you can apply an evolution technique to the scaling factor for the viscous dissipation terms. Applying an evolution function $f\ (S)\ = S$ on this scaling factor allows you to introduce viscous dissipation effects gradually. You can also apply an evolution technique to the cause of the viscous dissipation (e.g., the flow rate in the entry section). Another approach is to relate k to $1/S$ and evolve from the simple heat diffusion problem to the convection-diffusion problem.
- Shear-thinning fluids are very good candidates for the evolution technique. Consider a power-law fluid, or any fluid that behaves like a power-law fluid in the actual range of shear rate. When the power-law index n is low, it is generally impossible to obtain a converged solution with the Newton-Raphson method if you impose the actual value of n from the start.

  There are two remedies to this problem. You can use the Picard iteration for the viscosity instead of the Newton-Raphson iteration, as described in *Viscosity-Related Iterations* (p. 221). Such a method generally converges, but the convergence can be slow. The other remedy is to associate n with the evolution parameter $1/S$, so that n decreases as S increases.

  Successive calculations with decreasing values of n, using Newton-Raphson iterations, provide a successful technique for calculating such complex flows.

  Note that, for very low values of n (0 to 0.2), it may still be necessary to use Picard iterations instead of Newton-Raphson iterations with evolution. Many iterations with the Picard scheme will be required to reach an appropriate level of convergence. With the Picard iteration, the convergence rate of the solver can be rather slow, especially with a low power index; and reaching a given relative convergence on the velocity does not necessarily guarantee an upper bound of the departure with respect to the actual solution. When using the Picard iteration, it is therefore suggested that you invoke a more severe relative convergence criterion on the velocity field, on the order of 10^{-4} (or sometimes 10^{-5}).

Chapter 11: Viscoelastic Flows

ANSYS POLYFLOW can be used to solve several types of flows, ranging from Newtonian inelastic to non-Newtonian viscoelastic.

This chapter describes viscoelastic flows, and provides information on how to model them with ANSYS POLYFLOW.

11.1. Overview of Viscoelastic Flow

ANSYS POLYFLOW is a finite-element program primarily designed for the analysis of industrial flow processes dominated by nonlinear viscous phenomena and viscoelastic effects. It is used in a wide variety of applications, including polymer and rubber processing, food rheology, glassworks furnaces, and many other problems involving the flow of non-Newtonian fluids. The theoretical foundation of ANSYS POLYFLOW is provided by the general principles of continuum mechanics, together with phenomenological or kinetic theoretical models for describing the rheological behavior of the fluid.

The range of validity of the Newtonian constitutive equation is limited to low-molecular-weight, homogeneous liquids. The flow phenomena observed with polymeric fluids, for example, cannot be predicted by the classical Navier-Stokes equations. Non-Newtonian behavior has many facets. Among them are the shear-rate dependence of the shear viscosity, the presence of normal stresses in viscometric flows, high resistance to elongational deformation, and memory effects associated with the elasticity of the fluid. The theoretical challenge is to translate the complex rheological behavior of polymeric fluids into suitable equations, and to use these models to predict flows in complex geometries.

11.1.1. Modeling Viscoelastic Flow

ANSYS POLYFLOW offers a wide variety of constitutive models for both non-Newtonian inelastic and viscoelastic fluids. It is, however, essential to be aware that none of these models leads to realistic predictions in *all* types of deformation of any particular non-Newtonian fluid.

In many cases, the use of non-Newtonian inelastic models will capture the physics of the flow with sufficient accuracy. The only modeling issue then is to identify a suitable nonlinear viscosity function that will fit available viscometric data. The problem is more complex, however, when viscoelastic effects must be taken into account.

Important

If the modeling procedure is to be realistic, you should select a viscoelastic model that at least gives satisfactory predictions in the standard rheometrical tests that appear most relevant for the flow under consideration. To help determine an appropriate viscoelastic model, you can use the viscoelastic fitting tool in the ANSYS POLYMAT preprocessor. See *Curve Fitting for Material Properties* (p. 205) and the ANSYS POLYMAT User's Guide for details.

Current formulations of viscoelastic flows lead to highly nonlinear problems whose mathematical nature combines ellipticity and hyperbolicity in a rather subtle way. In addition, most viscoelastic flows of practical interest involve internal and boundary layers in the stress and velocity fields, as well as strong singularities.

Differential Viscoelastic Models (p. 226) describes the differential approach to computing viscoelastic flow, and *Integral Viscoelastic Models* (p. 251) describes an integral approach. The differential approach is appropriate for most practical applications, while the integral approach is provided mainly as a tool for rheology research. The numerical methods are more robust for the differential approach; for the integral approach, the available numerical methods converge slowly. Finally, *Simplified Viscoelastic Model* (p. 264) describes a simplified viscoelastic model, which, via appropriate simplifications from the point of view of rheology and mathematics, enables the qualitative prediction of viscoelastic effects at a reduced computational cost.

This chapter assumes a basic familiarity with the phenomena associated with viscoelastic flow, and will not serve as an initial introduction to the topic. For background information about viscoelastic flow, see [*1*] (p. 715) or any similar handbook or textbook on rheology.

11.2. Differential Viscoelastic Models

The generalized Newtonian flow models discussed in *Generalized Newtonian Flow* (p. 207) are unable to describe viscoelastic phenomena related to normal stresses and stress relaxation, for example. Typically, vortex generation, extrudate swelling of some melts, and drag enhancement are due to normal-stress differences and high extensional viscosity that are characteristic of viscoelastic fluids. Information about differential modeling for viscoelastic flow is presented in this section. See *Integral Viscoelastic Models* (p. 251) for information about the integral modeling approach.

The differential approach to modeling viscoelastic flow is appropriate for most practical applications. Many of the most common rheological models for viscoelastic flow are provided in ANSYS POLYFLOW, including Maxwell, Oldroyd, Phan-Thien-Tanner, Giesekus, FENE-P, POM-POM, and Leonov. Appropriate choices for the viscoelastic model and related parameters can yield qualitatively and quantitatively accurate representations of viscoelastic behavior.

You can sometimes improve accuracy if you use multiple relaxation times to better fit the viscoelastic behavior at different shear rates. If required, you can even use different viscoelastic models for the different relaxation times, although this has a very limited physical basis.

Information about the models available in ANSYS POLYFLOW and the equations solved for each is presented in *Theory and Equations* (p. 227), and instructions for using these models are provided in *Problem Setup for Differential Viscoelastic Flows* (p. 237) – *Computing Differential Viscoelastic Flow* (p. 250).

11.2.1. Theory and Equations

11.2.1.1. Extra-Stress Tensor

For viscoelastic flows, the total extra-stress tensor is decomposed into a viscoelastic component $\mathbf{T}_1$ and a purely-viscous component $\mathbf{T}_2$:

$$\mathbf{T} = \mathbf{T}_1 + \mathbf{T}_2 \tag{11–1}$$

$\mathbf{T}_1$ is computed differently for each type of viscoelastic model. $\mathbf{T}_2$ is an optional (but recommended, as discussed in *Setting the Viscosity Ratio* (p. 245)) component, which is always computed from

$$\mathbf{T}_2 = 2\eta_2\mathbf{D} \tag{11–2}$$

where $\mathbf{D}$ is the rate-of-deformation tensor and η_2 is the viscosity factor for the Newtonian (i.e., purely-viscous) component of the extra-stress tensor. The viscosity ratio η_r is defined as η_2/η. The relationship of η_1 and η_2 to η is expressed by

$$\eta_1 = \left(1 - \eta_r\right)\eta \tag{11–3}$$

and

$$\eta_2 = \eta_r\eta \tag{11–4}$$

The extra stress $\mathbf{T}_2$, when present in *Equation 11–1* (p. 227), is usually interpreted as the solvent contribution to the stress in polymeric solutions, or as the stress response associated with very fast relaxation modes. As discussed here and in *Theory and Equations* (p. 251), the presence of a purely-viscous component has a significant impact on the mathematical properties of the equations governing viscoelastic flows, and always improves the convergence of the numerical method.

When a multi-mode viscoelastic model is used, the total extra-stress tensor is decomposed into a sum of individual viscoelastic components, and one purely-viscous component. To prevent ambiguous definition of the purely-viscous component, the corresponding viscosity factor is defined together with the first mode. Consequently, the remaining modes will not contain any purely-viscous components. You should be sure to specify carefully the values for η_1 and η_r for the first mode.

ANSYS POLYFLOW solves a fully-coupled system of equations, where the stress (one stress field for each mode, or relaxation time), velocity, and pressure are computed simultaneously. In the case of moving boundaries (free surfaces or moving interfaces), position variables are also coupled to the stress, velocity, and pressure (by default). A full Newton-Raphson scheme is used for the computation.

11.2.1.2. Basic Equations

For a differential viscoelastic flow, ANSYS POLYFLOW solves the constitutive equations for the extra-stress tensor, the momentum equations, the incompressibility equation, and (for nonisothermal flows) the energy equation.

For the constitutive equations for $\mathbf{T}$ (*Equation 11–1* (p. 227)), $\mathbf{T}_l$ is computed from a differential equation or from an algebraic equation involving a state variable (configuration tensor), which obeys a differential equation. Models of the so-called Oldroyd family (including the Maxwell, Oldroyd-B, White-Metzner, Phan-Thien-Tanner, and Giesekus models) obey a differential equation written in terms of the extra-stress tensor $\mathbf{T}_l$.

Next to models of the so-called Oldroyd family, there are also other models that are written in terms of quantities that more or less refer to the topology of macromolecular chains (deformation, orientation, stretching, etc.). For the FENE-P model, the extra-stress tensor is algebraically derived (as described later in this section) from a configuration tensor, which itself obeys a differential equation. Similarly, in the **POM-POM** model, the extra-stress tensor is algebraically derived from an orientation tensor and a stretching variable, both of which obey differential equations. Eventually, in the Leonov model for filled materials, the extra-stress tensor is algebraically obtained from two configuration tensors (for free and trapped chains, respectively), their inverse, and a structural variable that dictates the debonding of chains or the conversion from trapped to free status for chains. These configuration tensors and the structural variable obey differential equations.

For the models belonging to the Oldroyd family, $\mathbf{T}_l$ is computed from the following equation:

$$g\,(\mathbf{T}_l)\ \cdot \mathbf{T}_l + \lambda \frac{\delta \mathbf{T}_l}{\delta t} = 2\eta_l \mathbf{D} \qquad \textbf{(11–5)}$$

where $\mathbf{g}\,(\mathbf{T}_l)$ is a model-specific function, λ is a model-specific relaxation time, and η_l is a model-specific viscosity factor for the viscoelastic component of $\mathbf{T}$. The relaxation time λ is defined as the time required for the shear stress to be reduced to about one third of its original equilibrium value when the strain rate vanishes. A high relaxation time indicates that the memory retention of the flow is high. A low relaxation time indicates significant memory loss, gradually approaching Newtonian flow (zero relaxation time).

The term $\frac{\delta \mathbf{T}}{\delta t}$ is an objective derivative defined as a linear combination of lower- and upper-convected derivatives:

$$\frac{\delta \mathbf{T}}{\delta t} = \frac{\xi}{2}\overset{\Delta}{\mathbf{T}}_l + \left(1 - \frac{\xi}{2}\right)\overset{\nabla}{\mathbf{T}}_l \qquad \textbf{(11–6)}$$

for $0 \le \xi \le 2$. $\overset{\Delta}{\mathbf{T}}_l$ is the lower-convected time derivative of the viscoelastic extra stress:

$$\overset{\Delta}{\mathbf{T}}_l = \frac{D\mathbf{T}_l}{Dt} + \mathbf{T}_l \cdot \nabla \mathbf{v}^T + \nabla \mathbf{v} \cdot \mathbf{T}_l \qquad \textbf{(11–7)}$$

and $\overset{\nabla}{\mathbf{T}}_l$ is the upper-convected time derivative of the viscoelastic extra stress:

$$\overset{\nabla}{\mathbf{T}}_1 = \frac{D\mathbf{T}_1}{Dt} - \mathbf{T}_1 \cdot \nabla \mathbf{v} - \nabla \mathbf{v}^T \cdot \mathbf{T}_1 \tag{11–8}$$

For differential viscoelastic fluids with multiple relaxation times, $\mathbf{T}_1$ represents the sum of all viscoelastic contributions, each obeying its own constitutive equation (*Equation 11–6* (p. 228)) and characterized by its own material parameters (η_1, η_r, λ, etc.) for the model-specific function $\mathbf{g}(\mathbf{T}_1)$. Note that the purely-viscous component of the extra-stress tensor is defined through the first mode only; more precisely, the corresponding viscosity will be given by the product $\eta_1\eta_r$.

It is interesting to note that most viscoelastic models involve an equation written in terms of extra-stress components. This is the case for the models belonging to the Oldroyd family. As described later in this section, the FENE-P model involves an equation written in terms of a configuration tensor, while the POM-POM model involves two equations written in terms of an orientation tensor and a stretching variable. Also, the Leonov model for filled materials involves three equations, for free macromolecular chains, for trapped macromolecular chains, and for the structural (debonding) variable that controls the transition between trapped and free status. The viscoelastic extra-stress tensor is connected to the configuration tensor or to the orientation tensor and stretching variable through an algebraic equation. This may affect the available numerical schemes for solving viscoelastic flows with the FENE-P, POM-POM, or Leonov models.

The momentum equation is:

$$-\nabla p + \nabla \cdot (\mathbf{T}_1 + \mathbf{T}_2) + \mathbf{f} = \rho \mathbf{a} \tag{11–9}$$

The incompressibility equation is:

$$\nabla \cdot \mathbf{v} = 0 \tag{11–10}$$

where $\mathbf{v}$ is velocity.

The energy equation is presented as *Equation 13–6* (p. 284) in *Theory* (p. 283).

11.2.1.3. Viscoelastic Models

11.2.1.3.1. Upper-Convected Maxwell Model

The Maxwell model is one of the simplest viscoelastic constitutive equations. It exhibits a constant viscosity and a quadratic first normal-stress difference. In view of its rheological simplicity, it is recommended only when little information about the fluid is available, or when a qualitative prediction is sufficient. Despite its numerical simplicity, this model can be difficult from a computational point of view.

For the upper-convected Maxwell model, $\mathbf{T}_1$ is computed from

$$\mathbf{T}_1 + \lambda \overset{\nabla}{\mathbf{T}}_1 = 2\eta_1 \mathbf{D} \tag{11–11}$$

The purely-viscous component of the extra stress ($\mathbf{T}_2$) is equal to zero in the Maxwell model.

11.2.1.3.2. Oldroyd-B Model

The Oldroyd-B model is, like the Maxwell model, one of the simplest viscoelastic constitutive equations. It is slightly better than the Maxwell model, because it allows for the inclusion of the purely-viscous component of the extra stress, which can lead to better behavior of the numerical scheme. Oldroyd-B is a good choice for fluids that exhibit a very high extensional viscosity.

For the Oldroyd-B model, $\mathbf{T}_1$ is computed from *Equation 11–11* (p. 229) and the purely-viscous component $\mathbf{T}_2$ is computed (optionally) from *Equation 11–2* (p. 227). When a multi-mode viscoelastic model is used, the purely-viscous component of the extra-stress tensor is defined through the first mode only; more precisely, the corresponding viscosity will be given by the product $\eta_1 \eta_r$.

11.2.1.3.3. White-Metzner Model

Most fluids are characterized by shear-thinning and non-quadratic first normal-stress difference. With the White-Metzner model, it is possible to reproduce such viscometric features.

The White-Metzner model computes $\mathbf{T}_1$ from

$$\mathbf{T}_1 + \lambda \overset{\nabla}{\mathbf{T}}_1 = 2\eta_1 \mathbf{D} \tag{11–12}$$

and $\mathbf{T}_2$ is computed (optionally) from *Equation 11–2* (p. 227).

The relaxation time (λ) and the viscosity (η) can be constant or represented by the power law or the Bird-Carreau law for shear-rate dependence.

The power-law representation of the total viscosity is

$$\eta = K_\eta \left(\Lambda_\eta \dot{\gamma}\right)^{n_\eta - 1} \tag{11–13}$$

where K_η is the consistency factor, n_η is the power-law index, and Λ is the natural time.

The Bird-Carreau representation of the viscosity is

$$\eta = \left(\eta_0 - \eta_\infty\right) \left(1 + \Lambda_\eta^2 \dot{\gamma}^2\right)^{\frac{n_\eta - 1}{2}} + \eta_\infty \tag{11–14}$$

where Λ is the natural time (i.e., inverse of the shear rate at which the fluid changes from Newtonian to power-law behavior) and n_η is the power-law index.

η_1 and η_2 are then computed from *Equation 11–3* (p. 227) and *Equation 11–4* (p. 227).

The power-law representation of the relaxation time is

$$\lambda = K_{\lambda}\left(\Lambda_{\lambda}\dot{\gamma}\right)^{n_{\lambda}-1} \tag{11–15}$$

The Bird-Carreau representation of the relaxation time is

$$\lambda = \lambda_{0}\left(1+\Lambda_{\lambda}^{2}\dot{\gamma}^{2}\right)^{\frac{n_{\lambda}-1}{2}} \tag{11–16}$$

11.2.1.3.4. Phan-Thien-Tanner Model

The Phan-Thien-Tanner (PTT) model is one of the most realistic differential viscoelastic models. It exhibits shear thinning and a non-quadratic first normal-stress difference at high shear rates.

The PTT model computes $\mathbf{T}_1$ from

$$\exp\left[\frac{\varepsilon\lambda}{\eta_1}\mathrm{tr}\left(\mathbf{T}_1\right)\right]\mathbf{T}_1 + \lambda\left[\left(1-\frac{\xi}{2}\right)\overset{\nabla}{\mathbf{T}}_1 + \frac{\xi}{2}\overset{\Delta}{\mathbf{T}}_1\right] = 2\eta_1\mathbf{D} \tag{11–17}$$

and $\mathbf{T}_2$ is computed (optionally) from *Equation 11–2* (p. 227). When a multi-mode viscoelastic model is used, the purely-viscous component of the extra-stress tensor is defined through the first mode only; more precisely, the corresponding viscosity will be given by the product $\eta_1\eta_r$.

ξ and ε are material parameters that control, respectively, the shear viscosity and elongational behavior. In particular, a nonzero value for ε leads to a bounded steady extensional viscosity.

11.2.1.3.5. Giesekus Model

Like the PTT model, the Giesekus model is one of the most realistic differential viscoelastic models. It exhibits shear thinning and a non-quadratic first normal-stress difference at high shear rates.

The Giesekus model computes $\mathbf{T}_1$ from

$$\left(\mathbf{I}+\frac{\alpha\lambda}{\eta_1}\mathbf{T}_1\right)\cdot\mathbf{T}_1 + \lambda\overset{\nabla}{\mathbf{T}}_1 = 2\eta_1\mathbf{D} \tag{11–18}$$

and $\mathbf{T}_2$ is computed (optionally) from *Equation 11–2* (p. 227). When a multi-mode viscoelastic model is used, the purely-viscous component of the extra-stress tensor is defined through the first mode only; more precisely, the corresponding viscosity will be given by the product $\eta_1\eta_r$.

$\mathbf{I}$ is the unit tensor and α is a material constant. A nonzero value for α leads to a bounded steady extensional viscosity and a shear-rate dependence of the shear viscosity.

11.2.1.3.6. FENE-P Model

The FENE-P model is derived from molecular theories. In the simplest representation, the molecules are described as dumbbells, built of two spheres linked together by a nonlinear spring. Unlike in the Maxwell model, however, the springs are allowed only a finite extension, so that the energy of deformation of the dumbbell becomes infinite for a finite value of the spring elongation. This model predicts a realistic shear thinning of the fluid and a first normal-stress difference that is quadratic for low shear rates and has a lower slope for high shear rates.

The FENE-P model computes $\mathbf{T}_l$ from a configuration tensor $\mathbf{A}$ (state variable) on the basis of an algebraic equation, as follows:

$$\mathbf{T}_l = \frac{\eta_l}{\lambda}\left[\frac{\mathbf{A}}{1-\mathrm{tr}\,(\mathbf{A})/(3L^2)} - \frac{\mathbf{I}}{1-1/L^2}\right] \tag{11–19}$$

where $\mathbf{A}$ is computed from the following differential equation:

$$\frac{\mathbf{A}}{1-\mathrm{tr}\,(\mathbf{A})/(3L^2)} + \lambda\overset{\nabla}{\mathbf{A}} = \frac{\mathbf{I}}{1-1/L^2} \tag{11–20}$$

and L is the ratio of the maximum length of the spring to its length at rest. In this expression, L (or L^2) must be strictly larger than 1. Furthermore, as L^2 becomes infinite, the FENE-P model becomes equivalent to the Maxwell model. Finally, note that a slightly different expression exists for the FENE-P constitutive model, which differs only by the definition of L.

$\mathbf{T}_2$ is computed (optionally) from *Equation 11–2* (p. 227).

See [*3*] (p. 715) for additional information about the FENE-P model.

11.2.1.3.7. POM-POM Model [DCPP]

A viscoelastic constitutive equation, referred to as the **POM-POM** model, was introduced by McLeish and Larson [*20*] (p. 716). The pom-pom molecule consists of a backbone to which q arms are connected at both extremities. In a flow, the backbone orients in a Doi-Edwards reptation tube consisting of the neighboring molecules, while the arms may retract into that tube when high deformation is applied. The concept of the pom-pom macromolecule makes the model suitable for describing the behavior of branched polymers. The approximate differential form of the model is based on equations of macromolecular orientation, and macromolecular stretching in relation to changes in orientation [*20*] (p. 716).

The model has undergone subsequent changes and improvements to make it suitable for software implementation and to introduce a nonzero second normal stress difference. The formulation implemented in ANSYS POLYDATA, referred to as DCPP, is developed by Clemeur, Rutgers and Debbaut [*6*] (p. 715).

The DCPP model computes $\mathbf{T}_l$ from an orientation tensor, $\mathbf{S}$ and a stretching scalar $\wedge$ (states variables), on the basis of the following algebraic equation:

$$\mathbf{T}_l = \frac{G}{1-\xi}\,(3\wedge^2\mathbf{S} - \mathbf{I}) \tag{11–21}$$

where G is the shear modulus and ξ is a nonlinear material parameter (the nonlinear material parameter will be introduced later on). The state variables S and $\wedge$ are computed from the following differential equations:

$$\lambda\left[\left(1-\frac{\xi}{2}\right)\overset{\nabla}{\mathbf{S}}+\frac{\xi}{2}\overset{\Delta}{\mathbf{S}}\right]+\lambda\,(1-\xi)\;[2\mathbf{D}:\mathbf{S}]\,\mathbf{S}+\frac{1}{\wedge^{2}}\left[\mathbf{S}-\frac{\mathbf{I}}{3}\right]=0 \tag{11–22}$$

$$\lambda_s\frac{D\wedge}{Dt}-\lambda_s\,(\nabla\mathbf{v}:\mathbf{S})\wedge+(\wedge-1)\,e^{\frac{2\,(\wedge-1)}{q}}=0 \tag{11–23}$$

In these equations, λ and λ_s are the relaxation times associated to the orientation and stretching mechanisms respectively. The ratio, λ/λ_s should be within the range 2 to 10.

In the stretching (*Equation 11–23* (p. 233)), q characterizes the number of dangling arms (or priority) at the extremities of the pom-pom molecule or segment. It is an indication of the maximum stretching that the molecule can undergo, and therefore of a possible strain hardening behavior. q can be obtained from the elongational behavior. ξ is a nonlinear parameter that enables the introduction of a nonvanishing second normal stress difference in the DCPP model.

A multi-mode DCPP model can also be defined where the total viscoelastic extra-stress tensor given by *Equation 11–21* (p. 232) will be split into individual contributions. Each contribution will involve an orientation tensor governed by *Equation 11–22* (p. 233) and a stretching variable governed by *Equation 11–23* (p. 233). A few guidelines are required for the determination of the several linear and nonlinear parameters.

Consider a multi-mode DCPP model characterized by N modes sorted with increasing values of relaxation times λ_i (increasing seniority). The linear parameters, λ_i and G_i in *Equation 11–21* (p. 232) and *Equation 11–22* (p. 233) characterize the linear viscoelastic behavior of the model and can be determined with the usual procedure. Then the relaxation times (λ_{si}) for stretching should be determined. Following Inkson et al. [*16*] (p. 715), λ_i/λ_{si} should be equal to $4\pi^2$ times the average number of entanglements of backbone section. Though this value is usually unknown, it should be within the range of 2 to 10. For a completely unentangled polymer segment you may accept the physical limit of $\lambda_i = \lambda_{si}$. λ_{si} should also satisfy the constraint $\lambda_{i-1} \le \lambda_{si} \le \lambda_i$, since λ_{i-1} sets the fundamental diffusion time for the branch point controlling the relaxation of polymer segment (i).

The parameter q_i indicating the number of dangling arms (or priority) at the extremities of a pom-pom segment i, also indicates the maximum stretching that can be undergone by that segment, and therefore its possible strain hardening behavior. For a multi-mode DCPP model, both seniority and priority are assumed to increase together towards the inner segments; hence q_i should also increase with λ_i. The parameter q_i can be obtained from the elongational behavior.

ξ_i is a fifth set of nonlinear parameters. They control the ratio of second to first normal stress differences. The value of parameter ξ_i should range between 0 and 1. For moderate values, ξ_i corresponds to twice the ratio of the second to the first normal stress difference, and may decrease with increasing seniority.

As opposed to the models whose constitutive equations are directly expressed in terms of an extra-stress tensor, the planar flow of a pom-pom fluid may exhibit a stress component in the transverse direction. For such planar and axisymmetric flows, four components will be evaluated for the orientation tensor $\mathbf{S}$.

$\mathbf{T}_2$ is computed (optionally) from *Equation 11–2* (p. 227).

11.2.1.3.8. Leonov Model

Elastomers are usually filled with carbon black and/or silicate. From the point of view of morphology, macromolecules at rest are trapped by particles of carbon black, via electrostatic forces of the van der Waals type. Under a deformation field, electrostatic bonds can break and macromolecules become free, while a reverse mechanism may develop when the deformation ceases. Thus the macromolecular system may consist of trapped and free macromolecules, with a reversible transition from one state to the other.

Leonov and Simhambhatla have developed a rheological model for the simultaneous prediction of the behavior of trapped and free macromolecular chains. The model is for filled elastomers, and involves two tensor quantities and a scalar one. One of the tensor quantities focuses on the behavior of the free macromolecular chains of the elastomer, while the other focuses on the trapped macromolecular chains.

The model exhibits a yielding behavior. It is intrinsically nonlinear, since the nonlinear response develops and is observable at early deformations. By comparison, models of the Oldroyd family and the POM-POM [DCPP] model are extrinsically nonlinear, since the nonlinear regime develops with kinematics.

Extended information on the Leonov model can be found in the PhD thesis [*26*] (p. 716) by Murthy V. Simhambhatla (supervised by Arkadii I. Leonov) at the University of Akron, as well as in [*25*] (p. 716). Hereafter, we briefly recall the Leonov model, as well as its important features.

In a single-mode approach, the total stress tensor $\mathbf{T}$ can be decomposed as the sum of free and trapped contributions, as follows:

$$\mathbf{T} = \mathbf{T}_c + \mathbf{T}_k + \mathbf{T}_2 \tag{11–24}$$

In the previous equation, subscripts c and k refer to the free and trapped parts, respectively. Each of these contributions obeys its own equation. In particular, they invoke their own deformation field described by means of Finger tensors. An elastic Finger tensor $\mathbf{C}$ is defined for the free chains, which obeys the following equation:

$$2\lambda \overset{\nabla}{\mathbf{C}} + b\ (I_1, I_2)\ \ [\mathbf{C}^2 + \mathbf{C}\ (I_2 - I_1)\ /3 - \mathbf{I}]\ = 0 \tag{11–25}$$

where λ is the relaxation time and $\mathbf{I}$ is the unit tensor. I_1 and I_2 are the first invariant of $\mathbf{C}$ and $\mathbf{C}^{-1}$, respectively, and are defined in the following manner:

$$I_1 = \mathrm{tr}\ (\mathbf{C}) \tag{11–26}$$

$$I_2 = \mathrm{tr}\left(\mathbf{C}^{-1}\right) \tag{11–27}$$

Equation 11–25 (p. 234) also involves a material function $b\ (I_1, I_2)$, which describes the deformation history dependence. Although Leonov [*18*] (p. 715) has suggested various expressions, the following expression has been deemed to be endowed with enough regularity, for the time being:

$$b\ (I_1, I_2)\ = exp\left(m\left(\frac{I_2}{3} - 1 \right) \right) \quad \textbf{(11–28)}$$

The parameter m must be $\geq$ 0. This parameter slightly increases the amount of shear thinning.

Similarly, an elastic Finger tensor $\mathbf{K}$ is defined for the trapped chains, which obeys the following equation:

$$2\lambda \overset{\nabla}{\mathbf{K}} + f\ (\xi)\ \ [\mathbf{K}^2 + \mathbf{K}\,(J_2 - J_1)\ /3 - \mathbf{I}]\ = 0 \quad \textbf{(11–29)}$$

where J_1 and J_2 are the first invariant of $\mathbf{K}$ and $\mathbf{K}^{-1}$, respectively, and are defined in the following manner:

$$J_1 = \mathrm{tr}\ (\mathbf{K}) \quad \textbf{(11–30)}$$

$$J_2 = \mathrm{tr}\left(\mathbf{K}^{-1}\right) \quad \textbf{(11–31)}$$

In the equation for the trapped chains, the variable ξ quantifies the degree of structural damage (debonding factor), and is the fraction of the initially trapped chains that are debonded from the filler particles during flow. The function $f\ (\xi)$ is a scaling factor for the relaxation time λ, and is referred to as the “mobility function". A phenomenological kinetic equation is suggested for ξ:

$$q\lambda \frac{d\xi}{dt} + \xi = \lambda\,\dot{\gamma}\ (1 - \xi)\ /\gamma^{*} \quad \textbf{(11–32)}$$

In *Equation 11–32* (p. 235), $\dot{\gamma}$ is the local shear rate (or equivalently, the generalized shear rate) defined as the second invariant of the rate of deformation tensor $\mathbf{D}$, while γ^{*} is the generalized shear at which yielding occurs (yielding strain). Also, q is a dimensionless time factor, which may delay or accelerate debonding. For the mobility function $f\ (\xi)$ appearing in the constitutive *Equation 11–29* (p. 235) for $\mathbf{K}$, the following form is suggested, based on [*7*] (p. 715):

$$f\ (\xi\ )\ =\ \lfloor\ (1 - \xi\ )^{\ -\nu} - 1 + k\nu\ \rfloor\ /\ \nu \quad \textbf{(11–33)}$$

where ν is a power index in the mobility function.

Let us momentarily discard the parameter k. The above selection for the mobility function endows the rheological properties with a yielding behavior. When $f\ (\xi)$ is large (or unbounded), the algebraic term dominates the constitutive equation for $\mathbf{K}$, and the solution is expected to be $\mathbf{K} = \mathbf{I}$. When $f\ (\xi)$ is vanishing, $\mathbf{K}$ becomes governed purely by a transport equation. This may lead to trouble when solving a complex steady flow. Indeed, boundary conditions may become inappropriate when secondary motions (vortices) are encountered, and the system matrix may become singular. Finally, when $f\ (\xi)$ is small and nonvanishing, the situation is similar to the one involving a very long relaxation time λ. Based on this, parameter k can be understood as the value of the mobility function under no debonding.

The original equation is recovered from [7] (p. 715) when $k = 0$. Actually, for nonvanishing k, the suggested *Equation 11–33* (p. 235) departs from the reference one only by a small amplitude, and at low values of ξ.

Finally, in order to relate the Finger tensors to the corresponding stress tensor, potential functions are required. For $\mathbf{C}$ and $\mathbf{K}$, the following expressions are suggested:

$$W_c = \frac{3G}{2\ (n+1)} \left\{ (1-\beta) \left[(I_1/3)^{\ n+1} - 1 \right] + \beta \left[(I_2/3)^{\ n+1} - 1 \right] \right\} \tag{11–34}$$

$$W_k = \frac{3G}{2\ (n+1)} \left\{ (1-\beta) \left[(J_1/3)^{\ n+1} - 1 \right] + \beta \left[(J_2/3)^{\ n+1} - 1 \right] \right\} \tag{11–35}$$

where G is the shear modulus, while β and n are a coefficient and a power index in the potential function, respectively. Also, $0 \le \beta \le 1$ and $n \ge 0$.

It is interesting to note that β has no effect on the shear viscosity, while it contributes to a decrease of the elongational viscosity. On the other hand, the parameter n increases both shear and elongational viscosities. From there, stress contributions from free and trapped chains in *Equation 11–24* (p. 234) are given by the following equations, respectively:

$$\mathbf{T}_c = 2\ (\alpha + \xi)\ /\ (1+\alpha) \left[(\mathbf{C} - \mathbf{I})\ \frac{\partial W_c}{\partial I_1} - \left(\mathbf{C}^{-1} - \mathbf{I} \right) \frac{\partial W_c}{\partial I_2} \right] \tag{11–36}$$

$$\mathbf{T}_k = 2\ (1 - \xi)\ /\ (1+\alpha) \left[(\mathbf{K} - \mathbf{I})\ \frac{\partial W_s}{\partial J_1} - \left(\mathbf{K}^{-1} - \mathbf{I} \right) \frac{\partial W_s}{\partial J_2} \right] \tag{11–37}$$

where parameter α is the initial ratio of free to trapped chains in the system. A vanishing value of α indicates that all chains are trapped at rest, while a large value of α indicates a system that essentially consists of free chains. These stress components are subsequently incorporated into the momentum equation, and are subjected to appropriate initial and boundary conditions for $\mathbf{C}$, $\mathbf{K}$ and ξ. A comparison of the formulations developed by Leonov (see [26] (p. 716) and [25] (p. 716)) with the extra-stress contributions $\mathbf{T}_c$ and $\mathbf{T}_k$ that result from *Equation 11–36* (p. 236) and *Equation 11–37* (p. 236) reveals that a change has been introduced in the latter, which is motivated by the need of obtaining zero stresses at rest. As a matter of fact, the change does not affect the predictions of velocity, swelling, and total stress tensor. It affects the spherical part of the contributions $\mathbf{T}_c$ and $\mathbf{T}_k$, which can be compensated via the pressure.

$\mathbf{T}_2$ is computed from *Equation 11–2* (p. 227). However, it is important to mention that the additional viscosity factor η_2 should not vanish when starting a calculation with the Leonov constitutive model. Optionally, an evolution function can be subsequently applied to this parameter.

11.2.1.4. Temperature Dependence of Viscosity and Relaxation Time in Non-Isothermal Flows

The viscosity in a nonisothermal differential viscoelastic flow can be temperature-dependent. As described in *Theory and Equations* (p. 207), the viscosity will be multiplied by a temperature shift function *H(T)*. For nonisothermal differential viscoelastic flows, the relaxation time is multiplied by the same temperature shift function. Temperature-dependent functions available for nonisothermal differential viscoelastic flows are the Arrhenius law, the Arrhenius approximate law, and the WLF law, all described in *Theory and Equations* (p. 207).

11.2.1.5. The Components of a Tensor

As discussed previously in this section, tensorial quantities are calculated when a viscoelastic flow is solved. With the exception of velocity gradient and vorticity tensor, the calculated tensors are symmetric, and may involve vanishing components, depending on the geometry. Typically, the number of components is as follows:

- For 2D planar cases, three components are calculated.
- For 2D axisymmetric cases, four components are calculated.
- For all other cases, six components are calculated.

As is the case for the velocity components, tensor components must also be numbered. A pair of integer values is used for identifying a tensor component: the ij component of a tensor corresponds to the contribution along the ith direction as seen on an infinitesimal surface element whose normal vector is along the jth direction. In a Cartesian coordinate system, the integers 1, 2, and 3 correspond respectively to the x, y, and z directions. In a cylindrical coordinate system (for axisymmetric geometry), the integers 1, 2, and 3 correspond respectively to the r, z, and θ directions.

11.2.2. Problem Setup for Differential Viscoelastic Flows

The basic steps for setting up a differential viscoelastic flow are as follows:

1. Create a sub-task for the differential viscoelastic flow problem.

   Create a sub-task

   a. Select the appropriate problem type from the **Create a sub-task** menu.

      Differential viscoelastic isothermal flow problem

      or

      Differential viscoelastic nonisothermal flow problem

   b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. Define the material properties.

Material data

a. Select the differential viscoelastic model to be used, and set the related parameters. See *Choosing the Differential Viscoelastic Model* (p. 241) for guidelines on choosing an appropriate model.

Differential viscoelastic models

i. Select **1-st viscoelastic model** to specify the first model to be used.

1-st viscoelastic model

ii. Select the model you want to use, and specify the relevant parameters:

- Select **Maxwell model** to use the upper-convected Maxwell model (*Equation 11–11* (p. 229)). The inputs for this model are η_2 (**visc** in ANSYS POLYDATA) and λ (**trelax**).

- Select **Oldroyd-B model** to use the Oldroyd-B model. The inputs for this model are λ (**trelax**) in *Equation 11–11* (p. 229), and the total viscosity η (**visc**) and the viscosity ratio η_r (**ratio**).

  By default, η_r is set to zero, meaning that there is no purely-viscous component of the extra-stress tensor. To include the purely-viscous component ($\mathbf{T}_2$ in *Equation 11–2* (p. 227)), set η_r to a nonzero value. η_1 and η_2 will be computed from *Equation 11–3* (p. 227) and *Equation 11–4* (p. 227).

  If you do not want to include the purely-viscous component, simply keep the default value of zero for η_r (**ratio**) and η_1 will be equal to the full value of η (**visc**).

  If you want to include a purely-viscous component, specify the value for η_r. When a multi-mode viscoelastic model is used, the purely-viscous component of the extra-stress tensor is defined through the first mode only; more precisely, the corresponding viscosity will be given by the product $\eta_1\eta_r$. See *Setting the Viscosity Ratio* (p. 245) for more information about setting the viscosity ratio.

- Select **White-Metzner model** to use the White-Metzner model (*Equation 11–12* (p. 230)). The inputs for this model are the viscosity function for η, the relaxation time function for λ, and the viscosity ratio η_r (**ratio**).

  For the viscosity function, choose the desired model for computing viscosity and enter the related parameters:

  – For a constant viscosity, select **Constant viscosity** and then specify the value of η (**fac**).

  – To use the Bird-Carreau law (*Equation 11–14* (p. 230)), select **Bird Carreau law** and specify values for η_0 (**fac**), Λ (**tnat**), n_η (**expo**), and η_∞ (**facinf**).

  – To use the power law (*Equation 11–13* (p. 230)), select **Power law** and specify values for K_η (**fac**), n_η (**expo**), and Λ_η (**tnat**).

  For all three methods, η_1 and η_2 will be computed from *Equation 11–3* (p. 227) and *Equation 11–4* (p. 227). As described for the Oldroyd-B model above, specify a nonzero value for the viscosity ratio if you want to include the purely-viscous component of the extra stress. When a multi-mode viscoelastic model is used, the purely-viscous component of the extra-stress tensor is defined through the first mode only; more precisely, the corresponding viscosity will be given by the product $\eta_1 \eta_r$.

For the relaxation time function, choose the desired method for computing the relaxation time and enter the related parameters:

- For a constant relaxation time, select **Constant relaxation** and then specify the value of λ (**facr**).
- To use the Bird-Carreau law (*Equation 11–16* (p. 231)), select **Bird Carreau law** and specify values for λ_0 (**facr**), Λ (**tnatr**), and n_λ (**expor**).
- To use the power law (*Equation 11–15* (p. 231)), select **Power law** and specify values for K_λ (**facr**), n_λ (**expor**), and Λ_λ (**tnatr**).

- Select **Phan Thien - Tanner model** to use the PTT model (*Equation 11–17* (p. 231)). The inputs for this model are the total viscosity η (**visc**), the relaxation time λ (**trelax**), the material parameters ε (**eps**) and ξ (**xi**), and the viscosity ratio η_r (**ratio**).

  As described in *Setting the Viscosity Ratio* (p. 245), the PTT model requires a purely-viscous component of the extra-stress tensor for stability purposes. In addition, for a single-mode PTT model, η_r must be at least 1/9. See also the comments about viscosity ratio in the description of the Oldroyd-B model, above.

- Select **Giesekus model** to use the Giesekus viscoelastic model (*Equation 11–18* (p. 231)). The inputs for this model are the total viscosity η (**visc**), the relaxation time λ (**trelax**), the material constant α (**alfa**), and the viscosity ratio η_r (**ratio**).

  As described in *Setting the Viscosity Ratio* (p. 245), the Giesekus model may require a purely-viscous component of the extra-stress tensor for stability purposes. See also the comments about viscosity ratio in the description of the Oldroyd-B model, above.

- Select **FENE-P** to use the FENE-P model (*Equation 11–19* (p. 232)). The inputs for this model are the total viscosity η (**visc**), the relaxation time λ (**trelax**), the square of the length ratio L^2 (**Lsqrd**), and the viscosity ratio η_r (**ratio**).

- Select **POM-POM [DCPP]** to use the DCPP model (*Equation 11–21* (p. 232), *Equation 11–22* (p. 233) and *Equation 11–23* (p. 233)). The inputs for this model are the additional parameters: additional viscosity η_2 (**visc2**), relaxation time λ (**Trelax**), shear modulus G (**GO**), relaxation time for the stretching mechanism λ_s (**Tlambda**), number of arms q (**NbArms**) and the parameter controlling the second normal stress difference ξ (**xi**).

  If you want to include a purely viscous component to the extra-stress tensor, specify a nonzero value for the additional viscosity (**visc2**) (as per the Oldroyd-B model).

- Select **Leonov** to use the Leonov model (*Equation 11–24* (p. 234) – *Equation 11–37* (p. 236)). The inputs for this model are the following parameters: additional viscosity η_2 (**visc**), shear modulus G (**GO**), relaxation time λ (**trelax**), initial ratio of free to trapped chains α (**alpha**), coefficient β in potential function W (**beta**), power index n in potential function W (**n**), parameter m for function of deformation history dependence $b\ (I_1, I_2)$ (**m**), power index

ν in mobility function $f\ (\xi)$ (**nu**), mobility function under no debonding k (**k**), dimensionless time factor q (**q**), and yielding strain γ^{*} (**gamma***).

From the point of view of rheology and numerical simulation, for single-mode and multi-mode fluid models, a purely viscous contribution must be added to the total extra-stress tensor. This is largely motivated by the fact that the matrix of the discretized system can be singular when all fields are initialized to values that correspond to the solution at rest. Hence, the first or only mode will always be accompanied by a Newtonian stress contribution, which has a corresponding viscosity that receives a unit value by default. You can modify this value. It may be possible to apply an evolution to this Newtonian viscosity, which may eventually vanish.

Also, as suggested previously, a nonvanishing value *k* should be selected for the mobility function under no debonding. It may be possible to apply an evolution scheme in order to reduce it.

As can be seen, besides controlling the linear properties via parameters *G* and λ, the model involves two functions and several nonlinear parameters. In a single-mode approach, the influence of these parameters on the viscometric and elongational properties can be easily identified, and appropriate values can be selected accordingly. By default, the nonlinear parameters are assigned values that are relevant from the point of view of rheology. In a multi-mode approach, corresponding nonlinear parameters should preferably be identical for each mode, in order to facilitate the definition of a flow case.

b. For nonisothermal flows, specify the temperature dependence of the viscosity.

Temperature dependence of viscosity

You can specify no temperature dependence, or select the Arrhenius law, Arrhenius approximate law, or WLF law. See *Theory and Equations* (p. 207) and *Problem Setup* (p. 215) for details.

c. If inertia, heat convection, or natural convection are to be taken into account in the calculation, define the density, inertia terms, and gravity. (By default, density is equal to zero, inertia terms are neglected, and gravitational acceleration is equal to zero.) For many processing applications, the Reynolds number is so low that inertia terms can safely be neglected. Even in the absence of inertia terms, however, it may be necessary to assign nonzero values for density and gravitational acceleration, since they influence heat convection and buoyancy forces, respectively.

i. Set the density.

Density

Select **Modification of density** and enter a new value.

ii. Enable the inertia terms in the momentum equations.

Inertia terms

Select **Inertia will be taken into account** to enable the inertia terms. (To disregard the inertia terms, you can select **Inertia will be neglected**, the default setting.) Note that the option to take inertia into account will not be available if the density is equal to zero. You will need to specify a nonzero density first, in order to enable the inertia terms.

iii. Set the gravitational acceleration.

Gravity

Select **Modification of gx** and set the gravitational acceleration in the x direction. Repeat for the y and z components.

d. Set any other appropriate material properties. For nonisothermal flows, for example, see *Problem Setup* (p. 286) for instructions.

See *Computing Differential Viscoelastic Flow* (p. 250) for suggestions about using evolution to define material properties.

4. Define the flow boundary conditions.

   Flow boundary conditions

   See *Boundary Conditions* (p. 173) for details.

5. For nonisothermal flows, define the thermal boundary conditions.

   Thermal boundary conditions

   See *Boundary Conditions* (p. 173) and *Problem Setup* (p. 286) for details.

6. (optional) Modify the interpolation scheme used for the stresses.

   Interpolation

   See *Selecting the Interpolation* (p. 246) for details.

11.2.3. Choosing the Differential Viscoelastic Model

11.2.3.1. Analyzing the Problem

Proper selection of a differential viscoelastic model is certainly one of the most important and difficult aspects of the simulation of a viscoelastic flow. In order to make an educated choice, you should collect as much data about the fluid properties as you can. The ANSYS POLYMAT module of ANSYS POLYFLOW can be used to compare experimental data with model predictions, in order to help you determine appropriate viscoelastic models. See the ANSYS POLYMAT User's Guide for details.

11.2.3.1.1. Viscoelasticity

Typical information about viscoelastic fluid properties includes the following:

- steady viscometric properties (shear viscosity η and first normal-stress difference N_1). These data characterize the fluid in the presence of large shear deformations.
- oscillatory viscometric properties (storage and loss moduli G' and G''). These data characterize the fluid in the presence of small deformations, and are also known as linear viscoelastic data.

- elongational (extensional) viscosity. Although obtaining experimental data for elongation is difficult, knowledge of the elongational viscosity is essential for choosing the viscoelastic model and estimating the values of the associated parameters when the flow under consideration involves a significant amount of elongation (e.g., film casting, fiber spinning). Sometimes, qualitative information may help.

This information is not enough to estimate the importance of viscoelasticity in a given process. You will also need to characterize the flow itself and compare a characteristic (relaxation) time of the material to a characteristic time of the flow. In many situations, the flow can be characterized by a critical shear rate The critical shear rate should be understood as a wall shear rate in a region of high gradient. In an extrusion process, for example, a critical shear rate will occur at the wall in the vicinity of the die exit. In a contraction or expansion flow, the critical shear rate will occur in the smallest section.

It should also be noted that, due to technological limitations of some rheometry equipment, it is not always possible to obtain viscoelastic data in the range of shear rates (or frequencies) where the process operates. In this case, the only option is to extrapolate experimental data for higher shear rates or frequencies, and select a model on a qualitative basis. See the ANSYS POLYMAT User's Guide for information about using the ANSYS POLYMAT module for this purpose.

11.2.3.1.2. The Weissenberg Number

The knowledge of both the fluid properties and the flow field can help you identify the viscoelastic character of the flow. A dimensionless number used to determine the viscoelasticity of a flow is the Weissenberg number We, which is the product of the relaxation time λ by a typical shear rate $\dot{\gamma}$:

$$\mathrm{We} = \lambda \dot{\gamma} \qquad \textbf{(11–38)}$$

For an axisymmetric flow problem, a first estimate of the shear rate is given by

$$\dot{\gamma} = \frac{4Q}{\pi r^3} \qquad \textbf{(11–39)}$$

where Q is the volumetric flow rate and r is the radius.

When We is low, generalized Newtonian models are sufficient to describe the flow. Only at higher We values are viscoelastic models required to characterize memory effects. A high value of We means that a problem is difficult to solve numerically, because of the nonlinear nature of the system.

11.2.3.1.3. Inertia Effects

In addition to viscoelasticity, inertia terms can also affect the flow behavior. This can be the case for low-viscosity polymer calculations. To characterize the influence of inertia, the Reynolds number is used:

$$\mathrm{Re} = \frac{\rho v L}{\eta} \qquad \textbf{(11–40)}$$

The combination of the Weissenberg number and the Reynolds number leads to the viscoelastic Mach number:

$$M = \sqrt{\mathrm{We}\,(\mathrm{Re})} \qquad \textbf{(11–41)}$$

For viscoelastic flows characterized by a Mach number greater than 1 (i.e., with inertia forces present), the unknown fields (stresses and velocity) are simultaneously controlled by viscoelastic forces and inertia. From a mathematical point of view, both stress and velocity fields are convected. This usually increases the level of nonlinearity. From a physical viewpoint, the combination of viscoelasticity and inertia can lead to phenomena that are not observed under other circumstances. The delayed die swell is a typical example. In this case, the swelling of the fluid does not occur immediately at the die exit, but some distance beyond, as if the fluid is not “aware" that it has entered a free region.

A particular strategy is required for high-Mach-number flows. In a physical experiment, the geometry is fixed and the flow rate will increase from a low value to the prescribed value. This leads to a simultaneous increase of viscoelastic and inertia effects (of both We and Re). Therefore, you should take these effects into account from the beginning of the calculation. In the frame of an evolution strategy, this means that We and Re should simultaneously increase. You can accomplish this by increasing the flow rate or by increasing the relaxation time λ and fluid density ρ through the same evolution function.

11.2.3.1.4. Storage and Loss Moduli

When data for the storage and loss moduli G' and G'' are available, their intersection (occurring at a shear rate $\dot{\gamma}_I = 1/\lambda$) is often a reasonable choice for selecting a typical relaxation time. Flows characterized by a typical shear rate lower than $\dot{\gamma}_I$ are essentially dominated by viscous forces, while viscoelastic effects may play an important role in flows characterized by a shear rate greater than $\dot{\gamma}_I$.

The viscoelastic model you select should reproduce the experimental behavior for shear rates in the vicinity of the characteristic shear rate of the flow to be simulated. This may lead to inappropriate model properties for low or very high shear rates, but such extremes are usually not encountered in real flow conditions. The FENE-P model is sometimes better than the others at covering the shear-thinning behavior of real polymers, at least for a single relaxation time. However, other models, such as Giesekus or DCPP, can also provide results with a similar level of quality, with one or multiple relaxation modes. Finally, for filled materials, the Leonov model can be considered. A single-mode representation is able to reproduce realistic properties, such as shear thinning.

In many circumstances, linear properties are easier to obtain than nonlinear ones. The use of empirical rules can be considered for evaluating other properties, when possible. Alternatively, it may be that only the shear viscosity is available. Usually, when the choice is permitted, shear viscosity should be considered before the linear properties.

11.2.3.2. Guidelines for Model Selection

11.2.3.2.1. Maxwell Model

The default viscoelastic model is the Maxwell model. As noted in *Theory and Equations* (p. 227), the Maxwell model is one of the simplest viscoelastic constitutive equations (but also the most difficult numerically). It exhibits a constant viscosity and a quadratic first normal-stress difference. Due to its simplicity, it is recommended only when little information about the fluid is available, or when a qualitative prediction is sufficient. You should switch to at least an Oldroyd-B model for any real polymer.

11.2.3.2.2. Oldroyd-B Model

The Oldroyd-B model is, like the Maxwell model, one of the simplest viscoelastic constitutive equations. It is slightly better than the Maxwell model, because it allows for the inclusion of the purely-viscous component of the extra stress, which can lead to better behavior of the numerical scheme. Oldroyd-B is a good choice for fluids that exhibit a very high extensional viscosity.

11.2.3.2.3. White-Metzner Model

The White-Metzner model is able to reproduce viscometric features such as the shear thinning and non-quadratic first normal-stress difference that characterize most fluids. The White-Metzner model also provides additional flexibility, by allowing the use of different functions for the shear-rate dependence of the viscosity and relaxation time. When experimental data about the shear viscosity and first normal-stress difference are available, you can easily obtain the material parameters for the White-Metzner model by curve fitting. First, fit the shear viscosity data. Then select the function for the relaxation time on the basis of the first normal-stress difference in simple shear flow.

The Bird-Carreau law is recommended for the relaxation time, because it yields a constant (and bounded) relaxation time at low shear rates, and usually makes the use of an evolution strategy successful. The power-law dependence for the relaxation time should be avoided, because it leads to high relaxation times for low shear rates.

Despite its interesting features from a viscometric viewpoint, the White-Metzner model sometimes causes spurious oscillations in the solution at high shear rates. As a result, it is not generally recommended; the PTT, Giesekus, and FENE-P models are preferred instead.

11.2.3.2.4. PTT Model

The PTT model is one of the most realistic differential viscoelastic models. It exhibits shear thinning and a non-quadratic first normal-stress difference at high shear rates. A nonzero value for ε will lead to a bounded, steady extensional viscosity.

However, the PTT model requires a purely-viscous component to be added to the extra-stress tensor for stability reasons. See *Setting the Viscosity Ratio* (p. 245) for details. The PTT model provides poor control of the shear viscosity at high shear rates when used with a single relaxation time. The use of multiple relaxation times can improve the control.

11.2.3.2.5. Giesekus Model

Like the PTT model, the Giesekus model is one of the most realistic differential viscoelastic models. It exhibits shear thinning and a non-quadratic first normal-stress difference at high shear rates. A nonzero value for α will lead to a bounded steady extensional viscosity.

For this model, a purely-viscous component must be added to the extra-stress tensor in the Giesekus model for stability reasons. See *Setting the Viscosity Ratio* (p. 245) for details.

11.2.3.2.6. FENE-P Model

The FENE-P model is derived from molecular theories and is based on the assumption that the material behaves as a series of dumbbells linked together by springs. Unlike in the Maxwell model, the springs are allowed only a finite extension, so that the energy of deformation of the dumbbell becomes infinite for a finite value of the spring elongation. This model predicts a realistic shear thinning of the fluid and a first normal-stress difference that is quadratic for low shear rates and has a lower slope for high shear rates.

In practice, it has been observed that the FENE-P model accurately models viscometric properties for a number of fluids.

11.2.3.2.7. POM-POM Model [DCPP]

The pom-pom molecule consists of a backbone to which q arms are connected at both extremities. In a flow, the backbone may orient in a Doi-Edwards reptation tube consisting of the neighboring molecules, while the arms may retract into that tube.

The concept of the pom-pom macromolecule makes the model suitable for describing the behavior of branched polymers. The approximate differential form of the model is based on the equations of macromolecular orientation and macromolecular stretching in connection with changes in orientation.

In this construction, the pom-pom molecule is allowed only a finite extension, which is controlled by the number of dangling arms. In particular, the strain hardening properties are dictated by the number of arms. Beyond that, the model predicts realistic shear thinning behavior, as well as a first and a possible second normal stress difference.

11.2.3.2.8. Leonov Model

Rubber compounds consist of an elastomer matrix filled with carbon black and/or silicate. From the point of view of morphology, elastomer macromolecules at rest are trapped by particles of carbon black, via electrostatic forces of the van der Waals type. Under deformation, these electrostatic bonds can break and macromolecules become free. A reverse mechanism develops when the deformation ceases. In general, elastomer systems consist of trapped and free macromolecules, with a reversible transition from one state to the other.

The Leonov model for filled elastomers is able to predict the macroscopic behavior of free and trapped chains under deformation, as well as the transition from one state to the other. It involves two tensor quantities and a scalar one. One of the tensor quantities focuses on the behavior of the free macromolecular chains of the elastomer, while the other focuses on the trapped macromolecular chains. The model is intrinsically nonlinear, since the nonlinear response develops and is observable at early deformations.

11.2.4. Setting the Viscosity Ratio

For simple shear flows, some of the differential viscoelastic models require a purely-viscous component to be added to the extra-stress tensor for stability reasons. This is true for the PTT model when ξ is nonzero, and for the Giesekus model when α is greater than 0.5. In these cases, the ratio of the Newtonian viscosity to the total viscosity must be greater than or equal to 1/9. The inclusion of a purely-viscous component is recommended for other models as well, in order to make the problem more stable numerically. Eventually, a nonvanishing value has to be selected for the additional Newtonian viscosity of the Leonov model. When needed, an evolution scheme can be applied on it.

When a multi-mode viscoelastic model is used, the purely-viscous component of the extra-stress tensor is defined through the first mode only; more precisely, for models of the Oldroyd family and for the FENE-P model, the corresponding viscosity will be given by the product $\eta_1 \eta_r$. For the DCPP and Leonov models, you will explicitly enter the corresponding.

The addition of a purely-viscous component affects both the shear viscosity and the normal-stress difference of the model. Shear thinning is still present, but the viscosity curve also shows a plateau zone at high shear rates, while the normal-stress difference is uniformly reduced.

11.2.5. Selecting the Interpolation

11.2.5.1. Interpolation for Pressure and Velocity

For information about selecting the interpolation for pressure and velocity, see *Controlling the Interpolation* (p. 218).

11.2.5.2. Interpolation for Viscoelastic Stresses

Differential viscoelastic flows explicitly introduce viscoelastic extra-stress variables or state variables in addition to velocity and pressure variables. This family of mixed representations is described as the mixed-mixed element, because the velocities act as constraints on the extra-stress tensor, and the mass conservation equation is a constraint on the velocity field.

Several interpolation schemes are available in ANSYS POLYFLOW. The selection of an interpolation scheme for a field must obey certain rules (of the LBB type). Most of the schemes available in ANSYS POLYDATA imply quadratic velocities and linear pressures in 2D, and use the mini-element for velocities and pressure in 3D. In applications involving contact, the linear velocity is used, with a constant interpolation for the pressure. For the viscoelastic tensors (extra-stress, configuration, orientation), (bi)quadratic, 2x2, 4x4 bilinear, EVSS and DEVSS interpolations are available depending on the selected model and geometry.

In EVSS interpolation, the constitutive equation is not solved in terms of $\mathbf{T}$ (viscoelastic extra-stress tensor). Instead, $\mathbf{T}$ is split into purely viscous and an elastic components. This split form is substituted into the constitutive equation, which is rewritten in terms of the elastic component and solved. Combining both viscous and elastic components recovers the actual viscoelastic stress tensor. Further details on the EVSS method can be found in [*22*] (p. 716) and [*31*] (p. 716).

The use of the (bi)linear interpolation for the elastic component of the stress and rate-of-deformation tensors makes the EVSS method one of the cheapest from the computational point of view. The EVSS formulation is not applicable to FENE-P, DCPP, and Leonov viscoelastic models. The DEVSS method is an alternative for these models.

The DEVSS method consists of adding a purely viscous term into the momentum equation that is expressed in terms of velocity unknowns, and removing an equivalent contribution expressed in terms of rate-of-deformation unknowns. Such a formulation keeps the constitutive equation unchanged. Further details can be found in [*14*] (p. 715). The DEVSS method, like EVSS, uses the (bi)linear interpolation for the stress and rate-of-deformation tensors making it one of the cheapest from the computational point of view. A numerical viscosity factor has to be specified for the DEVSS method which can be accessed through the advanced features in the menu.

By default, ANSYS POLYFLOW considers the sum of all individual viscosity factors from all the modes. The value is automatically updated when data of the rheological model is altered. You can also specify a preferred value, which will not be updated, unless explicitly stated. In addition, the selection of a particular interpolation for the viscoelastic extra stress can be combined with a streamline-upwinding (SU) or a Galerkin technique. Streamline upwinding Petriv-Galerkin is also available, in combination with the 4x4 bilinear sub-interpolation for the viscoelastic stress.

The default upwinding schemes have been chosen for optimal stability and changing them is not recommended unless you want to experiment with a particular numerical scheme. For 2D viscoelastic flows with a single mode, you can use the so-called 4×4-SU interpolation, which combines a high discretization level for the extra-stress tensor and the streamline upwinding method. This combination is robust for solving problems where elasticity plays a significant role, especially in the presence of flow singularities, such as reentrant corners or die-exit lips. However, this method is computationally expensive.

11.2.5.2.1. The Streamwise Approximation for Tensors (SAFT) Technique

Especially for 3D flows, viscoelastic flows calculations are often computationally expensive. This originates from the number of viscoelastic unknowns that are introduced with each of the modes. A fact is that shear stress and normal stress are the only nonvanishing components of the viscoelastic stress tensor in a simple shear flow; frequently, these stress contributions dominate in a general flow situation, while the contributions in the transverse direction are considered negligible. It is therefore reasonable to use an approximate integration of the viscoelastic constitutive equations, which tracks unknown tensors in the streamwise direction. This optional technique is referred to as the streamwise approximation for tensors (SAFT). The SAFT technique is applied to tensors invoked in viscoelastic flow simulations, and reduces the number of viscoelastic unknowns by a factor of two. Note that in order to achieve this reduction, some quantities are discarded and retrieved on the basis of an algebraic equation instead of the original differential equation.

Because the SAFT technique assumes that the transverse components of the stress tensor are negligible, it is not suitable for fluid models characterized by a nonvanishing second normal stress or normal stress difference. Hence, the SAFT technique is only made available for certain constitutive equations—namely, Maxwell, Oldroyd, and White-Metzner, as well as Phan Thien-Tanner when the second normal stress difference vanishes—and is not available for other viscoelastic models. As long as the individual contributions obey the previously listed constitutive equations, the SAFT technique can be applied to single-mode as well as multi-mode models. In addition to the constitutive equation limitations, the SAFT technique is not available for flow induced crystallization modeling (see *Flow Induced Crystallization (FIC)* (p. 271)), nor for the Narayanaswamy model used in the prediction of residual stresses and deformations (see *Residual Stresses and Deformations* (p. 493)).

The objective of the SAFT technique is to reduce the calculation time. It can be used only with the EVSS and DEVSS techniques, and is available only for flow simulations that are calculated using a 3D mesh (i.e., it is not available for problems that are 2D axisymmetric, 2.5D planar and axisymmetric, etc.). Note that the SAFT approximation performs better in combination with the DEVSS technique. Also, the DEVSS technique involves a numerical viscosity factor, which can possibly be adjusted in order to compensate for the discarded stress components.

By discarding the transverse components of the stress tensor, the SAFT technique affects the balance of force; this in turn can affect some predictions. Investigations that evaluated the effect of the SAFT technique on predicted results have found that the most visible consequence is the possibility of increased extrudate swelling. Hence, it is a good practice to evaluate the consequences of the SAFT technique for a given rheological model under specific geometric and kinematic conditions. Increasing the numerical viscosity invoked by the DEVSS technique leads to an artificial increase of the corresponding transverse contribution, which can control such swelling. This originates from the fact that the SAFT technique applies only to explicitly calculated tensorial quantities.

11.2.5.2.2. Default Options and Parameters

- In most cases of differential viscoelastic flows, the default scheme for $\mathbf{T}$ is the EVSS formulation combined with an SU technique ([*22*] (p. 716) and [*31*] (p. 716)). In ANSYS POLYDATA, this option corresponds to the menu item, **EVSS SU for stresses** under the **Interpolation** menu. The EVSS element explicitly introduces an interpolation for the rate-of-deformation tensor (common to all modes), and an interpolation for each component of the viscoelastic extra-stress tensor.

This allows ANSYS POLYFLOW to use a lower interpolation order for the stress field. The EVSS element is therefore computationally less expensive than the 4x4 element, especially for the multiple-mode case. However, as mentioned previously, the EVSS technique is not available for FENE-P, DCPP, and Leonov models, and the quadratic and DEVSS-SU interpolations are selected by default for 2D and 3D flows, respectively. The quadratic interpolation is also available for the other models (only for 2D isothermal flow with a single relaxation time). In the **Interpolation** menu in ANSYS POLYDATA, this option corresponds to the **Quadratic element for stresses** menu item. This element is not as stable as the 4x4 and EVSS elements, but requires less CPU time and memory than the 4x4 element.

- All components of tensors involved in a viscoelastic flow simulation are calculated by default. It may be useful to invoke the SAFT technique for large cases, in order to speed up the calculation.

11.2.5.2.3. Selecting an Interpolation

- If you want to use the classical MIX1 algorithm (when available), select **Quadratic element for stresses**. No other parameters must be adjusted.

  Quadratic element for stresses

- If you have to use the algorithm based on the enriched interpolation for the viscoelastic stresses (when available), respectively with streamline upwinding technique and with streamline upwinding/Petrov-Galerkin algorithm, select:

  4x4 SU element for stresses

  4x4 SUPG element for stresses

  No other parameters must be adjusted.

- If you want the EVSS algorithm (when available), with and without the streamline upwinding technique, select:

  EVSS for stresses

  EVSS SU for stresses

  No other parameters must be adjusted.

- If you want the DEVSS algorithm with and without the streamline upwinding technique.

  DEVSS for stresses

  DEVSS SU for stresses

- If you want to enable the SAFT technique, select **Enable SAFT technique**

  Enable SAFT technique

  Note that the SAFT technique is only available if the EVSS-SU or DEVSS-SU method is selected. Additionally, if the DEVSS-SU method is selected, the numerical viscosity can be modified via the advanced options (as explained in a description that follows). If another interpolation is subsequently selected, the SAFT technique is automatically disabled. You can disable the previously selected SAFT technique manually by selecting:

Disable SAFT technique

When the SAFT technique is invoked, related information may be found in the listing file. First of all, statements indicating the reduction of the number of unknowns are written. Next, some convergence messages will be duplicated for the viscoelastic fields; when such messages are produced, they concern the original full tensor as well as the one that results from the SAFT technique. In actuality, the first tensor depends on the second one.

- To enter a value for the numerical viscosity factor, select **Advanced options** and choose either manual or automatic setting of the DEVSS numerical viscosity.

  Advanced options

  When you select the DEVSS or DEVSS-SU algorithm, it is strongly recommended that you keep a nonvanishing value for the numerical viscosity factor. The matrix of the system can become singular when a vanishing viscosity factor is used, as a result of the linear interpolation of the variable connected to the constitutive equation.

  Also, when invoking the SAFT technique, the viscosity factor can be adjusted (e.g., on the basis of a known result), in order to compensate for excessive swelling. Some simulation results have been shown to better match reference solutions when the numerical viscosity factor is about four times the sum of partial viscosities. This statement is empirical, and it is quite possible that a lower or a larger value may be needed under some circumstances.

11.2.5.2.4. Combining Interpolation

In some situations, the use of a robust finite-element interpolation (i.e., involving many stress variables, at greater CPU expense) is required only in a small part of the computational domain. In such cases, you can select different interpolation types on different subdomains, using the **Sub-interpolation** menu item at the bottom of the **Interpolation** menu. For viscoelastic flows, this item is available only for interpolating the viscoelastic extra stresses of 2D isothermal viscoelastic flows with a single relaxation time.

For accessing the sub-interpolation menu for the extra-stress of a 2D isothermal viscoelastic flow, first select the 4x4 interpolation (SU or SUPG) for the extra-stress tensor, as explained above, then select

Sub-interpolation

and then identify the domain where a different interpolation is required.

> **Important**
>
> Sub-interpolating can lead to some reduction in CPU time and memory, while retaining the advantages of robustness. However, recent (and continuing) improvements in computer speed make sub-interpolation schemes less attractive.

11.2.5.2.5. Iterative Scheme for Viscosity and Relaxation Time

When you specify a power-law function for viscosity and/or relaxation time in conjunction with the White-Metzner model, ANSYS POLYFLOW will, by default, use Newton-Raphson iterations. In some cases, it may be better to use Picard iterations. See *Viscosity-Related Iterations* (p. 221) for details.

11.2.5.3. Interpolation for Non-Isothermal Flows

The interpolation schemes available for nonisothermal differential viscoelastic flow calculations are the same as those available for nonisothermal generalized Newtonian flow calculations. See *Interpolation for Nonisothermal Flows* (p. 221) for details.

11.2.6. Computing Differential Viscoelastic Flow

11.2.6.1. Using Evolution

Most steady-state viscoelastic flow problems require an evolution scheme to converge. You should generally start with a low Weissenberg number, and gradually increment it within an evolution. Although the 4×4 element is robust enough to allow the calculation of viscoelastic flows at high Weissenberg numbers, in most situations the first evolution step should be made at We of the order of 0.3 to 0.5. The viscoelastic character of the flow depends on both the relaxation time λ and the flow rate Q. An evolution strategy can therefore be applied to either one, with an increasing evolution function $f\ (S)$.

When an evolution strategy is used, the initial and final values for the evolution parameter S are important data. Again, based on the values of λ and Q, the first nonlinear solution (evaluated at S_0) should be selected so that the initial value of We is about 0.3 to 0.5. If the evolution scheme is applied to λ, the initial value of S can be zero.

If the evolution scheme is applied to Q, a nonzero value for S is required, in order to start with a nonzero velocity field. A nonzero starting value for Q is always required when moving boundaries are involved. The final value of S should be selected appropriately for the flow problem being solved. See *Evolution* (p. 517) for complete details about using evolution.

11.2.6.2. Sample Applications

Consider a viscoelastic flow characterized by a typical shear rate $\dot{\gamma} = 200$ s^{-1} and a flow rate $Q = 1.5$ cm^3/s. From viscometric data, the relaxation time is 0.1 s. Thus, $\mathrm{We} = 20$.

The procedure for applying an evolution scheme to the flow rate is as follows. To start the calculations, the initial value of We should be lower than 0.3. The initial flow rate Q_0 should be specified so that the corresponding shear rate is about 3 s^{-1}. S_0 should therefore be set so that $Q_0 = Qf\ (S)\ = 0.00225$ cm^3/s. In the frame of the evolution strategy, ANSYS POLYFLOW will increase the flow rate from the initial value up to the final value.

A more complex situation may occur for a White-Metzner model using a Bird-Carreau law with a low power-law index for the shear viscosity (see *Theory and Equations* (p. 227)). In this case, the initial flow rate must meet two criteria: the initial value of We must be lower than 0.3 and the initial characteristic shear rate must be located in the plateau zone of the viscosity curve ($\dot{\gamma} < 1/\Lambda_\eta$).

For such complex problems, the evolution strategy can be applied simultaneously on different parameters (relaxation time, power-law index, etc.) with different evolution functions $f\ (S)$. Still, the easiest approach involves applying the evolution strategy only on the flow rate Q. Indeed, a low flow rate leads to a weakly nonlinear problem, while, from a physical viewpoint, the evolution strategy applies on the flow rate rather than on the fluid model itself.

11.2.7. Supported Features for Differential Viscoelastic Flow Calculations

In terms of geometry, modeling, and numerical options, several are available for differential viscoelastic flow calculations, but some combinations of them are not. All types of geometries (2D planar, 2D axisymmetric with or without swirl, 2D channel flow, and 3D) can be used in a differential viscoelastic flow calculation. For numerics, the mixed, EVSS and DEVSS methods are available. Finally, isothermal, nonisothermal, steady-state, time-dependent, single-mode, and multi-mode viscoelastic models are all available.

The EVSS method can be used for steady-state and transient calculations of isothermal and nonisothermal viscoelastic flows with a single mode or multiple modes, in 2D (planar, axisymmetric, channel, and swirling) and 3D geometries. However, this feature applies only to models whose constitutive equation is written in terms of the extra-stress tensor (models of the so-called Oldroyd family).

The DEVSS method can be used for steady-state and transient calculations of isothermal and nonisothermal viscoelastic flows, with a single mode or multiple modes, in 2D (planar, axisymmetric, channel and swirling) and 3D geometries. This feature applies to all models. The mixed methods can be used for steady-state and transient calculations of isothermal viscoelastic flows with a single relaxation mode, in 2D (planar, axisymmetric, channel, or swirling) geometries, for all differential viscoelastic models described in *Differential Viscoelastic Models* (p. 226). For blow molding and thermoforming applications or film casting applications, see *Blow Molding and Thermoforming* (p. 379) or *Film Casting* (p. 417) for additional information.

11.3. Integral Viscoelastic Models

In addition to the differential approach to solving viscoelastic flow (described in *Differential Viscoelastic Models* (p. 226)), ANSYS POLYFLOW also provides an integral approach.

11.3.1. Introduction

While the differential approach is well-suited for practical applications, the integral approach is generally used for advanced rheological research. ANSYS POLYFLOW provides several numerical models for viscoelastic flow, including Doi-Edwards and KBKZ. Appropriate choices for the viscoelastic model and related parameters can yield qualitatively and quantitatively accurate representations of viscoelastic behavior.

Information about the models available in ANSYS POLYFLOW and the equations solved for each is presented in *Theory and Equations* (p. 251), and instructions for using these models are provided in *Problem Setup for Integral Viscoelastic Flows* (p. 256) – *Additional Hints* (p. 264). Note that the integral approach to modeling viscoelastic flow is limited to 2D flows; it cannot be applied to 3D flows.

11.3.2. Theory and Equations

11.3.2.1. Extra-Stress Tensor

Note

Equation 11–1 (p. 227) – *Equation 11–4* (p. 227) are used to compute the extra-stress tensor for integral viscoelastic flow as well as differential viscoelastic flow. See *Theory and Equations* (p. 227) for details.

11.3.2.2. Basic Equations

For an integral viscoelastic flow, ANSYS POLYFLOW solves the constitutive equations for the extra-stress tensor, the momentum equations, the incompressibility equation, and (for nonisothermal flows) the energy equation. For the constitutive equations for $\mathbf{T}$ (*Equation 11–1* (p. 227)), $\mathbf{T}_I$ is computed at time t from the following equation:

$$\mathbf{T}_I(t) = \int_0^\infty M(s)\left[\Phi_I(I_I,I_2)\,C_t^{-1}(t-s) + \Phi_2(I_I,I_2)\,C_t(t-s)\right]ds \tag{11–42}$$

where

$M(s)$ = model-specific memory (kernel) function

Φ_I = model-specific function of the scalar invariants

Φ_2 = model-specific function of the scalar invariants

C_t = Cauchy-Green strain tensor

t = current time

s = metric for time integrals (or past time measured with respect to t)

I_I and I_2 are the scalar invariants of the Cauchy-Green strain tensor:

$$I_1 = \mathrm{tr}\left(C_t^{-1}\right) \tag{11–43}$$

and

$$I_2 = \mathrm{tr}\,(C_t) \tag{11–44}$$

The various integral viscoelastic models are characterized by the form of the functions $M(s)$, $\Phi_I(I_I,I_2)$, and $\Phi_2(I_I,I_2)$.

The momentum and incompressibility equations are provided in *Equation 11–9* (p. 229) and *Equation 11–10* (p. 229), respectively. See *Theory and Equations* (p. 227) for details.

For nonisothermal flows, the energy equation is presented as *Equation 13–6* (p. 284) in *Theory* (p. 283). To obtain $\mathbf{T}$ from *Equation 11–1* (p. 227), $\mathbf{T}_I$ can be computed from the isothermal constitutive equation (*Equation 11–42* (p. 252)), provided that a modified time scale ξ is used for evaluating the strain history:

$$\mathbf{T}_I = \int_0^\infty M(s(\xi))\left[\Phi_I(I_I,I_2)\,C_t^{-1}(t-\xi) + \Phi_2(I_I,I_2)\,C_t(t-\xi)\right]\frac{ds}{d\xi}d\xi \tag{11–45}$$

The modified time scale is related to s through the following equation:

$$s = \int_0^{\xi} \phi \left(T \left(t - \chi \right) \right) d\chi \tag{11–46}$$

where ϕ is the shift function, which can be obtained from steady-state shear-viscosity curves at different temperatures. This is the principle of time-temperature equivalence. Equations for ϕ are presented later in this section.

The numerical technique used for solving integral viscoelastic flows implements an uncoupled scheme, where the computation of the viscoelastic extra stress is performed separately from that of the flow kinematics (and temperature). In this aspect, the technique differs from those used for differential viscoelastic flows where stresses, velocity, and pressure variables are all calculated simultaneously.

With an integral model, on the basis of known velocity and temperature fields, ANSYS POLYFLOW first computes the viscoelastic extra stress by means of the constitutive equations (*Equation 11–42* (p. 252) and *Equation 11–45* (p. 252)).The kinematics and temperature fields are then updated by solving *Equation 11–9* (p. 229), *Equation 11–10* (p. 229), and *Equation 13–6* (p. 284), where the viscoelastic stresses act as a known pseudo-body-force term. (This means that stresses are computed at the Gauss point of the finite element.)

11.3.2.3. Viscoelastic Models

11.3.2.3.1. Doi-Edwards Model

The Doi-Edwards model is characterized by shear thinning and a non-quadratic first normal-stress difference at high shear rates. It also predicts a nonvanishing second normal-stress difference and a finite steady extensional viscosity. In the Doi-Edwards model, $\mathbf{T}_l$ is computed from

$$\mathbf{T}_l = \int_0^{\infty} \frac{96\eta}{\pi^4 \lambda^2} \sum_{k=0}^{\infty} \exp\left(\frac{-\left(2k+1\right)^2 s}{\lambda} \right) \left[\Phi_l C_t^{-1} \left(t - s \right) + \Phi_2 C_t \left(t - s \right) \right] ds \tag{11–47}$$

where

$$\Phi_l = 5 \left[I_l + 2 \left(I_2 + 3.25 \right)^{0.5} - 1 \right]^{-1} \tag{11–48}$$

and

$$\Phi_2 = -\Phi_l \left(I_2 + 3.25 \right)^{-0.5} \tag{11–49}$$

In *Equation 11–47* (p. 253), k represents the relaxation mode.

$\mathbf{T}_2$ (optional, but strongly recommended) is computed from *Equation 11–2* (p. 227).

11.3.2.3.2. KBKZ Model

The KBKZ model provides additional accuracy by including a damping function in its constitutive equations. $\mathbf{T}_1$ is computed from

$$\mathbf{T}_1 = \frac{1}{1-\theta}\int_0^\infty \sum_{i=1}^{N} \frac{\eta_i}{\lambda_i^2} \exp\left(\frac{-s}{\lambda_i}\right) H(I_1, I_2) \left[C_t^{-1}(t-s) + \theta C_t(t-s)\right] ds \tag{11–50}$$

and $\mathbf{T}_2$ (optional, but strongly recommended) is computed from Equation 11–2 (p. 227).

In Equation 11–50 (p. 254), i represents the relaxation mode and θ is a scalar parameter that controls the ratio of the normal-stress differences:

$$\frac{N_2}{N_1} = \frac{\theta}{1-\theta} \tag{11–51}$$

and H is the damping function. The simplest case (Lodge-Maxwell model) is for no damping: $H = 1$ and $\theta = 0$.

The Papanastasiou-Scriven-Macosko (PSM) model computes H from

$$H = \frac{\alpha}{\alpha + I - 3} \tag{11–52}$$

where α is a material parameter that primarily influences the shear-thinning behavior.

The Wagner model computes H from

$$H = \exp\left(-n(I-3)^{0.5}\right) \tag{11–53}$$

where n is a material parameter that influences both the shear viscosity and the elongational behavior of the material.

The reversible PSM model uses Equation 11–52 (p. 254), allowing H to decrease and increase along a particle trajectory. The irreversible PSM model allows H only to decrease.

Similarly, the reversible Wagner model uses Equation 11–53 (p. 254), allowing H to decrease and increase along a particle trajectory. The irreversible Wagner model allows H only to decrease.

In both Equation 11–52 (p. 254) and Equation 11–53 (p. 254), I is computed from

$$I = \beta I_1 + (1-\beta) I_2 \tag{11–54}$$

where I_1 and I_2 are given by Equation 11–43 (p. 252) and Equation 11–44 (p. 252). β is a material parameter that influences only the elongational behavior of the material.

11.3.2.3.3. Equivalent Generalized Newtonian Models

The generalized Newtonian model that is equivalent to an integral model in simple steady shear (i.e., the generalized Newtonian model that exhibits the same steady shear viscosity law as the integral model) is

$$\mathbf{T}_I = 2\eta_{VE}\mathbf{D} \tag{11–55}$$

where

$$\eta_{VE} = \frac{\mathbf{T}_s(\dot{\gamma})}{\dot{\gamma}} \tag{11–56}$$

$\mathbf{T}_s(\dot{\gamma})$ is given by the viscoelastic constitutive equation (*Equation 11–47* (p. 253) or *Equation 11–50* (p. 254)) in a simple shear flow. $\mathbf{T}_2$ (optional, but recommended) is computed from *Equation 11–2* (p. 227).

The solution for the equivalent generalized Newtonian model can be compared with the viscoelastic model results to determine the relative contribution of elasticity to the behavior of the flow being modeled.

11.3.2.4. Temperature Shift Functions for Non-Isothermal Flows

Three models are available for the temperature shift function ϕ in *Equation 11–46* (p. 253) the Arrhenius law, the Arrhenius approximate law, and the WLF law, all described in *Temperature-Dependent Viscosity Laws* (p. 212). It is also possible to eliminate the temperature dependence using a temperature shift function equal to 1.

11.3.2.5. Numerical Method for Integral Viscoelastic Flow

The numerical simulation requires the simultaneous solution of the constitutive equations (*Equation 11–1* (p. 227), *Equation 11–2* (p. 227), and *Equation 11–42* (p. 252)), together with the momentum equations (*Equation 11–9* (p. 229)) and the incompressibility condition (*Equation 11–10* (p. 229)).

In order to obtain convergence of the uncoupled scheme, an evolution scheme that implements a progressive change from a Newtonian solution to the integral viscoelastic solution is recommended. The scheme is based on the so-called evolutive viscosity η_E, which determines a Newtonian (or generalized Newtonian) stress tensor $\mathbf{T}_E$. An extra-stress tensor $\mathbf{T}(\omega)$ is defined as follows:

$$\mathbf{T}(\omega) = \omega\mathbf{T} + (1-\omega)\mathbf{T}_E \tag{11–57}$$

where $\mathbf{T}_E = 2\eta_E\mathbf{D}$ and ω is a parameter that depends on the evolution parameter S.

Replacing $\mathbf{T}$ by $\mathbf{T}(\omega)$ in *Equation 11–9* (p. 229) yields

$$-\nabla p + \nabla\cdot\left(\omega\mathbf{T} + (1-\omega)\,\mathbf{T}_E\right) + \mathbf{f} = \rho\mathbf{a} \tag{11–58}$$

When ω is zero, the problem is purely Newtonian (when η_E is constant) or generalized Newtonian (when η_E varies). When $\omega = 1$, *Equation 11–58* (p. 256) defines the viscoelastic problem. In an evolution scheme, $\omega = 0$ at $S_{init} = 0$ and $\omega = 1$ at $S_{final} = 1$.

For every fixed value of ω, an iterative procedure is used to find velocity and pressure (and temperature, for nonisothermal flows) fields that satisfy *Equation 11–58* (p. 256). The increments of the velocity and pressure, $\delta\mathbf{v}$ and δp, are calculated as follows:

$$\nabla\cdot\delta\mathbf{v} = -\nabla\cdot\mathbf{v} \tag{11–59}$$

$$\begin{aligned}-\nabla\delta p + \eta_E\nabla\cdot\left(\nabla\delta\mathbf{v} + \nabla\delta\mathbf{v}^T\right) - \rho\left(\delta\mathbf{v}\cdot\nabla\mathbf{v} + \mathbf{v}\cdot\nabla\delta\mathbf{v}\right) + \mathbf{f} = \\ \nabla p - \nabla\cdot\left(\omega\mathbf{T} + (1-\omega)\,\mathbf{T}_E\right) - \mathbf{f} + \rho\mathbf{v}\cdot\nabla\mathbf{v}\end{aligned} \tag{11–60}$$

When $\omega = 1$, *Equation 11–60* (p. 256) becomes

$$\begin{aligned}-\nabla\delta p + \eta_E\nabla\cdot\left(\nabla\delta\mathbf{v} + \nabla\delta\mathbf{v}^T\right) - \rho\left(\delta\mathbf{v}\cdot\nabla\mathbf{v} + \mathbf{v}\cdot\nabla\delta\mathbf{v}\right) + \mathbf{f} = \\ \nabla p - \nabla\cdot\mathbf{T} - \mathbf{f} + \rho\mathbf{v}\cdot\nabla\mathbf{v}\end{aligned} \tag{11–61}$$

See *Using Evolution to Compute Integral Viscoelastic Flow* (p. 262) for details about using this numerical scheme.

11.3.3. Problem Setup for Integral Viscoelastic Flows

The basic steps for setting up an integral viscoelastic flow are as follows:

1. Create a sub-task for the integral viscoelastic flow problem.

   Create a sub-task

   a. Select the appropriate problem type from the **Create a sub-task** menu.

      Integral viscoelastic isothermal flow problem

      or

      Integral viscoelastic non-isothermal flow problem

   b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. Define the material properties.

   Material data

   a. Select the integral viscoelastic model to be used and set the related parameters. See *Choosing the Integral Viscoelastic Model* (p. 260) for guidelines on choosing an appropriate model.

      Integral Viscoelastic models

      i. The KBKZ model is the default **Type of model** listed at the top of the menu. To use the Doi-Edwards model, select **Switch to Doi-Edwards Model**.

         Switch to Doi-Edwards Model

         To switch back to the KBKZ model, select **Switch to KBKZ Model**

         Switch to KBKZ Model

      ii. If you want to perform the simulation for the generalized Newtonian model that has the same shear viscosity as the selected viscoelastic model (using *Equation 11–55* (p. 255)), choose **Switch to Generalized Newtonian Flow.**

         Switch to Generalized Newtonian Flow

         As mentioned in *Equivalent Generalized Newtonian Models* (p. 255), the solution for the equivalent generalized Newtonian model can be compared with the viscoelastic model results to determine the relative importance of elasticity on the behavior of the flow being modeled.

         To switch back to the selected viscoelastic model, select **Switch to Viscoelastic Model**.

         Switch to Viscoelastic Model

      iii. Specify the range of relaxation modes for the model.

         Modify the spectrum

         A. Specify how you want to define the relaxation modes. By default, the relaxation spectrum is defined by means of (viscosity, relaxation time) pairs. To specify (relaxation force, relaxation time) pairs, select **Switch to Elasticity - Relaxation time data**.

            Switch to Elasticity-Relaxation time data

            To switch back to the (viscosity, relaxation time) pairs, select **Switch to Viscosity - Relaxation time data**.

            Switch to Viscosity-Relaxation time data

         B. (for KBKZ model only) Specify the number of relaxation modes. You can define up to 8 modes.

            Number of relaxation modes

Since the Doi-Edwards model performs a summation over an infinite number of relaxation modes (see *Equation 11–47* (p. 253)), the specification of an explicit number of modes is irrelevant.

C. Specify the data pairs, either for all relaxation modes or for just a selected few.

- To specify data pairs for all modes, select **Modify the whole spectrum**.

  Modify the whole spectrum

  When prompted, enter the requested data.

- To specify data pairs for a subset of the modes, select **Modify some values of the spectrum**.

  Modify some values of the spectrum

  Then select each mode to be modified and enter the requested data when prompted.

When you are satisfied with all settings, select **No modification** to accept them.

iv. (for KBKZ model only) Specify the damping function.

Modify the damping function

- Select **Lodge-Maxwell model** (the default) for no damping. No further inputs are required.
- Select **Reversible Papanastasiou-Scriven model** to use the reversible version of *Equation 11–52* (p. 254). The inputs for this model are α in *Equation 11–52* (p. 254) and β in *Equation 11–54* (p. 254).
- Select **Irreversible Papanastasiou-Scriven model** to use the irreversible version of *Equation 11–52* (p. 254). The inputs are the same as for the reversible version.
- Select **Reversible Wagner model** to use the reversible version of *Equation 11–53* (p. 254). The inputs for this model are n in *Equation 11–53* (p. 254) and β in *Equation 11–54* (p. 254).
- Select **Irreversible Wagner model** to use the irreversible version of *Equation 11–53* (p. 254). The inputs are the same as for the reversible version.

See *Choosing the Integral Viscoelastic Model* (p. 260) for more information about setting parameters for the damping function.

v. Specify the ratio of the normal-stress differences (for KBKZ model only).

Ratio of the diff. of Norm. stresses (N2/N1)

This value is used in *Equation 11–51* (p. 254) to compute the value of θ that appears in *Equation 11–50* (p. 254).

vi. Specify the additional viscosity.

Modification of the additional viscosity

To include the purely-viscous component of the extra stress tensor (T_2 in *Equation 11–2* (p. 227)), set η_2 (the additional viscosity) to a nonzero value.

vii. Specify the evolutive viscosity.

Management of the evolutive viscosity

See *Setting the Evolutive Viscosity* (p. 261) for details.

viii. Define the numerical integration scheme for the viscoelastic model.

Numerical integration

See *Additional Strategies for Convergence* (p. 263) for details.

ix. For nonisothermal flows, specify the temperature shift (ϕ in *Equation 11–46* (p. 253)).

Temperature dependence

You can specify no temperature dependence, or select the Arrhenius law, Arrhenius approximate law, or WLF law. See *Temperature-Dependent Viscosity Laws* (p. 212) and *Problem Setup* (p. 215) for details.

b. If inertia, heat convection, or natural convection are to be taken into account in the calculation, define the density, inertia terms, and gravity. (By default, density is equal to zero, inertia terms are neglected, and gravitational acceleration is equal to zero.) For many processing applications, the Reynolds number is so low that inertia terms can safely be neglected. Even in the absence of inertia terms, however, it may be necessary to assign nonzero value for density and gravitational acceleration, since they influence heat capacity and buoyancy forces.

i. Set the density.

Density

Select **Modification of density** and enter a new value.

ii. Enable the inertia terms in the momentum equations.

Inertia terms

Select **Inertia will be taken into account** to enable the inertia terms. (To disregard the inertia terms, you can select **Inertia will be neglected**, the default setting.) Note that the option to take inertia into account will not be available if the density is equal to zero. You will need to specify a nonzero density first, in order to enable the inertia terms.

iii. Set the gravitational acceleration.

Gravity

Select **Modification of gx** and set the gravitational acceleration in the x direction. Repeat for the y component.

   c. Set any other appropriate material properties. For nonisothermal flows, for example, see *Problem Setup* (p. 286) for instructions.

4. Define the flow boundary conditions.

   Flow boundary conditions

   See *Boundary Conditions* (p. 173) for details.

5. For nonisothermal flows, define the thermal boundary conditions.

   Thermal boundary conditions

   See *Boundary Conditions* (p. 173) and *Problem Setup* (p. 286) for details.

11.3.4. Choosing the Integral Viscoelastic Model

The guidelines for analyzing the problem that were presented in *Choosing the Differential Viscoelastic Model* (p. 241) for differential viscoelastic flow are also valid for integral viscoelastic flow. Specific guidelines for integral viscoelastic model selection are provided below. The ANSYS POLYMAT module of ANSYS POLYFLOW can be used to compare experimental data with model predictions, in order to help you determine appropriate viscoelastic models. See the ANSYS POLYMAT User's Guide for details.

Both the Doi-Edwards model and the KBKZ model provide realistic simulations of viscoelastic flow behavior, although the KBKZ model provides more flexibility in the choice of the damping function.

11.3.4.1. Doi-Edwards Model

Theoretically, the Doi-Edwards model has an infinite number of relaxation times (represented by the k summation in *Equation 11–47* (p. 253)) determined by two parameters: the main relaxation time and the zero-shear-rate viscosity. This model is characterized by shear thinning and a non-quadratic first normal-stress difference at high shear rates. It also predicts a nonzero second normal-stress difference and a finite steady extensional viscosity.

For simple shear flows, however, a purely-viscous component must be added to the extra-stress tensor for stability purposes. The viscosity associated with the purely-viscous stress leads to a plateau zone at high shear rates. Thus, the slope of the curve $\log(\eta)$ as a function of $\log(\dot{\gamma})$ is greater than -1.

11.3.4.2. KBKZ Model

The KBKZ model adds another level of complexity compared to the Doi-Edwards model. In addition to the spectrum that describes the linear viscoelastic behavior of the material, you can also define a damping function. Both the PSM and the Wagner damping functions involve two material constants that mainly control the shear and elongational behavior, respectively: α and β for the PSM model, and n and β for the Wagner model. Another parameter, θ, controls the normal-stress-difference ratio. See *KBKZ Model* (p. 254) for details about the equations used for the damping function.

For the PSM model, high values of α lead to a large zone of the zero-shear-rate plateau. $\alpha = \infty$ corresponds to a constant damping function. In the latter case, the constitutive equation for the Maxwell-B model (*Equation 11–11* (p. 229)) is recovered. If the value of α is decreased, the zero-shear-rate plateau moves toward lower shear rates. In the case of the Wagner model, high values of n lead to a short plateau of zero shear rate, while a small value of n gives a plateau for a large range of shear rates.

For small values of I (defined by *Equation 11–54* (p. 254)), both the PSM and the Wagner damping functions are very similar. For high values of I, however, the exponential function of the Wagner model decreases more rapidly than the rational function of the PSM model. High values of I occur in the case of large deformations.

The β parameter has no effect on shear viscosity, nor on the first and second normal-stress differences. It affects only the extensional viscosity. A value of zero for β leads to an unbounded steady extensional viscosity. Increasing β decreases the maximum value of the steady extensional viscosity curve.

For both the PSM and the Wagner models, it is possible to introduce the concept of irreversibility originally mentioned by Wagner [*32*] (p. 716). The idea is that the damping function must never be allowed to increase along the past trajectory of a fluid particle. According to Wagner, this is a realistic assumption when intermolecular association occurs, as in an entanglement, for example. For a material moving at a high flow rate through a contraction followed by an expansion, it is reasonable not to allow the damping function to increase again after the entanglement.

The KBKZ model also makes use of the normal-stress-difference ratio. This ratio does not affect the shear viscosity or the normal-stress differences, but it does have an impact on the extensional viscosity.

Like the Doi-Edwards model, the KBKZ model requires a purely-viscous component of the extra-stress tensor in order to avoid instability in simple shear flows at high shear rates. For further information, see [*13*] (p. 715).

11.3.5. Setting the Evolutive Viscosity

Although the evolutive viscosity η_E (see *Numerical Method for Integral Viscoelastic Flow* (p. 255)) has been introduced for numerical reasons only, it is recommended that you specify its value based on physical considerations. Two possibilities are available: constant and variable evolutive viscosity.

By default, the evolutive viscosity is a constant value computed from

$$\eta_E = \sum_{i=1}^{N} \eta_i \tag{11–62}$$

where η_i are the viscosities of the viscoelastic modes, as specified in each viscoelastic model. This evolutive viscosity is independent of the shear rate. If you prefer to specify a constant value explicitly, select **Modification of the evolutive viscosity** in the **Evolutive Viscosity** menu. (The default value is zero, which indicates that *Equation 11–62* (p. 261) will be used to compute η_E.) The recommended value is the zero-shear-rate viscosity.

You can also specify a variable evolutive viscosity, which is computed from

$$\eta_E = \frac{\mathbf{T}_s(\dot{\gamma})}{\dot{\gamma}} \tag{11–63}$$

This evolutive viscosity exhibits the same shear-rate dependence as the integral fluid model itself.

To change to a variable evolutive viscosity, select **Switch to Variable Evolutive Viscosity** in the **Evolutive Viscosity** menu.

From a numerical point of view, the iterative scheme based on the constant evolutive viscosity seems to be more stable and is generally recommended. In some cases, however, constant-viscosity Newtonian solutions do not exist, but a solution can be found when the viscosity is shear-rate-dependent. This is the case for some problems with very large free-surface deformations (e.g., film blowing dies).

If you use *Equation 11–63* (p. 261) (variable evolutive viscosity), the system solved by ANSYS POLYFLOW has the following form:

$$\mathbf{A}\,(\dot{\gamma})\,\delta X = \mathbf{B}\,(\dot{\gamma}) \tag{11–64}$$

where $\mathbf{A}\,(\dot{\gamma})$ and $\mathbf{B}\,(\dot{\gamma})$ depend on the shear rate. This iterative scheme is less stable than if η_E is constant. In order to avoid numerical instabilities, it is possible to keep the evolutive viscosity in matrix $\mathbf{A}$ constant and limit the use of a variable evolutive viscosity to the right-hand side. Then the following system is solved instead:

$$\mathbf{A}\delta X = \mathbf{B}\,(\dot{\gamma}) \tag{11–65}$$

Equation 11–65 (p. 262) will be solved by ANSYS POLYFLOW if you use *Equation 11–62* (p. 261) for a constant evolutive viscosity or specify your own constant value explicitly.

In all cases, both techniques are equivalent once $\omega = 1$. Using the technique of *Equation 11–65* (p. 262), it is possible to start an integral evolution task from a generalized Newtonian solution.

11.3.6. Using Evolution to Compute Integral Viscoelastic Flow

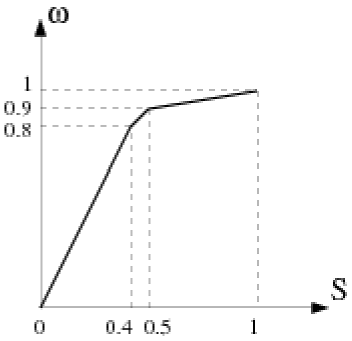
Most steady-state viscoelastic flow problems require an evolution scheme to converge. For integral viscoelastic flow, the evolution scheme consists of increasing ω from 0 to 1 by small steps until convergence is obtained at $\omega = 1$ (see *Numerical Method for Integral Viscoelastic Flow* (p. 255) for details). In other words, the viscoelastic stress is increased while its Newtonian counterpart decreases. Typical evolution parameters are given in *Table 11.1: Evolution Parameters for Integral Viscoelastic Calculations* (p. 262).

Table 11.1 Evolution Parameters for Integral Viscoelastic Calculations

Parameter	Value
S_{init}	0
S_{final}	1
ΔS_{init}	0.01
ΔS_{max}	0.05
ΔS_{min}	0.0001
prediction	explicit Euler

At the beginning of the evolution ($\omega \approx 0$), ω increases quickly because the solution is easy to obtain. Near the end of the evolution ($\omega \approx 1$) ω increases very slowly. The law that ANSYS POLYFLOW uses to link S to ω (shown in *Figure 11.1* (p. 263)) reflects this behavior.

Figure 11.1 ω as a Function of S



Here, continuity in the evolution scheme is achieved at $S = 0$; that is, the solution of the initial flow problem is identical to the limit of the evolution scheme for $S = 0$. Toward the end of the evolution task (usually around $\omega \approx 0.97$), convergence can become very slow. In order to avoid unnecessary iterations, you may want to start the post-convergence task directly from a converged solution at $\omega = 0.97$. See also *Additional Strategies for Convergence* (p. 263).

11.3.7. Additional Strategies for Convergence

As discussed in *Using Evolution to Compute Integral Viscoelastic Flow* (p. 262), it is strongly recommended that you define your viscoelastic flow problem using an evolution scheme. In addition, due to the use of an uncoupled formulation for the viscoelastic extra stresses and the velocity and pressure variables, a typical S step will generally require a larger number of iterations than for a Newton-Raphson coupled scheme. In order to minimize the CPU time, which can be quite large for integral viscoelastic simulations (especially if the problem involves recirculation zones), some modifications to the numerical scheme parameters are recommended.

It is generally not necessary to perform many iterations with a very small convergence criterion for a given value of S. Since many S steps are required, and the intermediate steps are not physically meaningful, it is more economical to require a strict convergence tolerance only when S reaches its final value. A convergence criterion of 0.01 with 5 iterations is usually used for the intermediate S steps.

The convergence criterion and the number of iterations can be set in the **Numerical parameters** menu. See *Using the Solver* (p. 575) for details.

At the end of the evolution scheme, additional iterations of the integral problem should be performed in steady-state mode with a lower value for the convergence criterion. Typically, in order to reach a converged final solution, you should perform additional iterations with a convergence criterion of 0.001 or 0.0001. You can accomplish this by executing an additional task after the original task, or by continuing the ANSYS POLYFLOW calculation starting from the solution obtained at S_{final}. This task is referred to as a post-convergence task.

For free surface or moving interface problems, improved numerical stability can be obtained if you decouple the velocity and pressure (and temperature, for nonisothermal flows) variables from the position variables, as described below.

The items in the **Numerical integration** menu allow you to fine-tune the integration method. These items are not recommended for general use.

11.3.7.1. Calculations Involving Moving Boundaries

In order to solve an integral viscoelastic problem that includes one or more free surfaces or moving interfaces, the following procedure is recommended:

1. Start the integral viscoelastic task in evolution mode with upwinding on the free surfaces, and use an uncoupled scheme for the moving boundaries. Set the convergence criterion to 0.01, in order to reduce the number of iterations performed.
2. Solve the post-convergence task by defining the same integral problem in steady-state mode, with upwinding on the free surfaces and an uncoupled scheme for the moving boundaries. Set the convergence criterion to 0.001 to obtain an accurate solution.

In the decoupled approach, ANSYS POLYFLOW computes the velocity and the pressure fields on a fixed domain. Then, based on the fixed velocity and pressure fields, the position of the free surface or moving interface is updated.

11.3.8. Additional Hints

11.3.8.1. Mesh Resolution

For integral viscoelastic calculations, the flow pathlines are computed by integrating the velocity field. In order to avoid tracking problems (e.g., “out of mesh by a wall" error message, which causes a reduction of the *S* step), you must ensure that the mesh is adequately resolved (i.e., not too coarse).

11.3.8.2. Performance on Vector Machines

When solving an integral viscoelastic flow problem, the viscoelastic extra stresses are uncoupled from the pressure and velocity variables, and most of the CPU time (more than 95%) is spent tracking the deformation history at each Gauss point. An uncoupled scheme means that convergence cannot be quadratic, as is the case for Newton-Raphson schemes. Since tracking the past deformation history involves many element-configuration checks, integral viscoelastic flow problems will not vectorize. Performance on vector machines will therefore not be as good as that achieved for differential viscoelastic calculations. Performance on parallel machines will be similar to parallel performance for differential viscoelastic calculations.

11.4. Simplified Viscoelastic Model

One of the interesting features of viscoelastic flow simulations is the prediction of extrudate swelling, which can be larger than its Newtonian counterpart. However, running complex 3D flows with a rheologically sophisticated model can sometimes be computationally expensive. Hence, we propose hereafter a simplified approach, which should enable the qualitative prediction of the extrudate free surface. This approach can be referred to as the “simplified viscoelastic model", or “light viscoelastic model". The suggested model is also sometimes referred to as the acronym “ScaFTen", which is the name of a research project that stands for “Scalar For Tensor".

11.4.1. Theory and Equations

It is known that the first normal stress difference is mainly responsible for enhanced extrudate swelling in extrusion flow. This is typically a viscoelastic property. With respect to this fact, the simplified viscoelastic model is an extension of existing Newtonian fluid models, in which a normal stress difference has been incorporated into the force balance. In other words, in simple shear flow along the first axis and with a shear rate $\dot{\gamma}$, the total extra-stress tensor $\mathbf{T}$ is given by:

$$\mathbf{T} = \begin{pmatrix} \psi\mu(\dot{\chi})\dot{\chi} & \eta(\dot{\gamma})\dot{\gamma} & \cdot \\ \eta(\dot{\gamma})\dot{\gamma} & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix} \tag{11–66}$$

In this tensor, the shear stress component is given by $\eta(\dot{\gamma})\ \dot{\gamma}$, which involves the shear rate dependent viscosity $\eta(\dot{\gamma})$. Several algebraic relationships are available for describing the shear viscosity, and can be found in *Generalized Newtonian Flow* (p. 207). In the present context, the following laws can be considered:

- constant viscosity (*Equation 10–5* (p. 208))
- Bird-Carreau law (*Equation 10–7* (p. 209))
- power law (*Equation 10–6* (p. 209))
- Cross law (*Equation 10–8* (p. 209))
- Carreau-Yasuda law (*Equation 10–15* (p. 212))
- modified Cross law (*Equation 10–9* (p. 210))

The first normal stress is given by the quantity $\psi\mu(\dot{\chi})\dot{\chi}$. This quantity involves the viscoelastic variable $\dot{\chi}$, a quantity $\mu(\dot{\chi})$ (which can be referred to as normal viscosity), and a weighting factor ψ.

The viscoelastic variable $\dot{\chi}$ obeys a transport equation involving a characteristic or relaxation time $\lambda(\dot{\gamma})$, which is given by:

$$\lambda(\dot{\gamma})\frac{D\dot{\chi}}{Dt} + \dot{\chi} = \dot{\gamma} \tag{11–67}$$

The equation is such that the solution $\dot{\chi} = \dot{\gamma}$ is recovered in simple shear flow. The normal viscosity $\mu(\dot{\chi})$ found in *Equation 11–66* (p. 265) is described by means of functions similar to those available for the shear viscosity $\eta(\dot{\gamma})$, where $\dot{\gamma}$ is replaced by $\dot{\chi}$. In order to facilitate the setup of a flow simulation involving the simplified viscoelastic model, identical dependences for $\mu(\dot{\chi})$ and $\eta(\dot{\gamma})$ are considered by default. However, it is important to note that different functions can be selected for the shear and normal viscosities.

Finally, for nonisothermal flows, temperature dependence can be selected for the shear and normal viscosities. Here, the Arrhenius law (*Equation 10–17* (p. 213)), the approximate Arrhenius law (*Equation 10–19* (p. 214)), and the WLF law (*Equation 10–21* (p. 214)) can be selected. When defining a nonisothermal case, a single function is used for describing the temperature dependence of the material functions $\eta(\dot{\gamma})$, $\mu(\dot{\chi})$, ψ, and optionally $\lambda(\dot{\gamma})$.

As far as the other features are concerned, the situation does not differ from that available for Newtonian or viscoelastic flows. In particular, 2D and 3D flows can be defined, and may include attributes such as inertia, gravity, viscous heating, etc. Also, in terms of boundary conditions, the situation does not differ from general viscoelastic flows. Free surfaces can be defined, as this is the primary motivation for the simplified viscoelastic model. The transport equation (*Equation 11–67* (p. 265)) for the viscoelastic variable $\dot{\chi}$ requires boundary conditions at the inlet of the computational domain. They are automatically selected and imposed along boundary sides where the inflow condition is selected.

11.4.2. Considerations

The following considerations should be noted when using the simplified viscoelastic model:

- By examining the previous equations that describe the material functions, it becomes clear that the model may involve a relatively large number of parameters. Therefore, it is generally suggested that you select identical algebraic functions for both shear and normal viscosities. This is also the default option.
- Very large swelling ratios are sometimes observed. This can be adjusted with an appropriate value of the weighting factor ψ.
- When considering non-constant material properties, the power law should be avoided if you expect large regions in which there is no deformation. Indeed, the power law exhibits unbounded values under zero deformation. Instead, functions that exhibit a plateau should be preferred, such as the Bird-Carreau law.
- It is recommended that you start a calculation without the additional contribution from the simplified viscoelastic model, and hence define an evolution strategy for the parameter ψ. A ramp function that increases quickly is a good choice.

11.4.3. Identifying Model Parameters and Functions

The simplified viscoelastic model is mainly an empirical construction. The key ingredient is the normal stress property that is introduced for the prediction of swelling. Although it is possible to qualitatively relate the swelling and the first normal stress difference, a quantitative relationship is not obvious. Methodologies have to be identified and developed for the determination of material functions and parameters. A stepwise technique is recommended for this purpose.

As a preliminary remark, it is worth mentioning that the simplified viscoelastic model has been developed and implemented mainly for the simulation of 3D extrusion flows, therefore including the prediction of extrudate swelling. In view of this, it should be considered acceptable to use cylindrical extrudate swelling data for the identification of the specific model properties.

As seen previously in *Theory and Equations* (p. 264), the simplified viscoelastic model involves three material functions and a parameter: the shear viscosity $\eta\ (\dot{\gamma})$, the normal viscosity $\mu\ (\dot{\chi})$, the relaxation time $\lambda\ (\dot{\gamma})$, and a weighting coefficient ψ. Typically, standard viscosity data (shear viscosity vs. shear rate) should be used for identifying the shear viscosity function. In most situations, shear thinning is experimentally observed, and algebraic relationships such as the power law, the Bird-Carreau law, or the Cross law will be good candidates. However, you should consider a law that exhibits a zero-shear plateau if you expect regions in which there is no deformation in the flow domain.

Next, a function and material parameters should be selected for the normal viscosity $\mu\ (\dot{\chi})$. By default, a relationship identical to the selected shear viscosity $\eta\ (\dot{\gamma})$ is considered, as this appears to be a reasonable initial choice. Of course, this default selection can be revised subsequently.

When eventually selecting the relaxation time and the weighting factor, it is recommended that you perform a fast 2D simulation of the axisymmetric extrudate swelling, to examine the effects of the remaining degrees of freedom. Typically, the weighting coefficient ψ will control the swelling intensity vs. flow rate, while the relaxation time function $\lambda\ (\dot{\gamma})$ will control the development of the extrudate diameter along the jet, and may also have an influence on the developed extrudate geometric attributes. Usually, a constant value or a Bird-Carreau law can be selected for the relaxation time. The value or zero-shear value should preferably be in agreement with the typical times involved in the flow. On the other hand, a series of calculations should be performed with various values of the weighting coefficient ψ via an evolution scheme, to examine the development of extrudate vs. flow rate. A comparison with experimental data on swelling should enable the selection of an appropriate numerical value for the weighting coefficient ψ.

11.4.4. Selecting the Interpolation

Unlike differential viscoelastic models, the simplified viscoelastic model introduces only one additional unknown field $\dot{\chi}$, next to velocity and pressure. Considering the fact that the simplified model is aimed at predicting viscoelastic effects at reasonable computational cost, it has been decided to select the interpolation accordingly. Hence, by default, the lowest interpolation functions that still verify the LBB conditions have been selected. In particular, the linear interpolation is selected for the velocity, together with a constant interpolation for the pressure. The additional unknown field $\dot{\chi}$ is interpolated by means of linear functions. Of course, you have the ability to change these selections, and to consider another interpolation (e.g., the mini-element).

11.4.5. Problem Setup for Simplified Viscoelastic Model

The basic steps for setting up a calculation with the simplified viscoelastic model are as follows:

1. Create a sub-task for the simplified viscoelastic flow problem.

   Create a sub-task

   a. Select the appropriate problem from the **Create a sub-task** menu.

      Simplified viscoelastic isothermal flow problem

      or

      Simplified viscoelastic non-isothermal flow problem

   b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. Define the material properties.

   Material data

   a. Define the shear rate dependence of the viscosity.

      See the corresponding step 3.(a) in *General Procedure* (p. 215) for the definition of the shear rate dependence of the viscosity.

   b. For nonisothermal flows, define the temperature dependence of the viscosity.

      See the corresponding step 3.(b) in *General Procedure* (p. 215) for the definition of the temperature dependence of the viscosity.

   c. If inertia, heat convection, or natural convection are to be taken into account in the calculation, define the density, inertia terms, and gravity.

See the corresponding step 3.(c) in *General Procedure* (p. 215) for the definition of density, inertia terms, and gravity.

d. Set any other relevant material properties (such as thermal conductivity, heat capacity, or the thermal expansion coefficient).

   See the corresponding step 3. in *General Procedure* (p. 286) for the definition of nonisothermal parameters.

e. Define the simplified viscoelastic model.

Simplified viscoelastic model

i. Define the first normal viscosity.

First normal viscosity

By default, the first normal viscosity is described with the same function as the shear viscosity, and involves the same parameters. Select **Switch to “Not linked to shear rate viscosity"** if you want to specify another function for the normal viscosity, and select the required function with the numerical parameters accordingly.

ii. Define the weighting coefficient.

Weighting coefficient

It may be appropriate to apply an evolution strategy on the weighting coefficient. A ramp function can be a good choice.

iii. Define the shear rate dependence of the relaxation time.

Shear rate dependence of relaxation time

By default, the relaxation time is vanishing. A Bird-Carreau or a power law can be selected for describing the shear rate dependence. For flow simulations involving the prediction of extrudate, you should consider a Bird-Carreau law, as it is bounded at vanishing shear rates.

iv. Specify the temperature dependence for the relaxation time.

Switch to H(T) applied on relaxation time

By default, the relaxation time does not depend on the temperature for nonisothermal flows. Select **Switch to “H(T) applied on relaxation time"** if you want the same temperature dependence as defined for the shear and normal viscosities.

4. Define the flow boundary conditions.

Flow boundary conditions

See *Boundary Conditions* (p. 173) for details, and *Using Evolution to Compute Generalized Newtonian Flow* (p. 222) for suggestions about using evolution on boundary conditions.

Important

Inflow boundary conditions should be selected at inlet(s) with the flow rate, as this is the only way of imposing a relevant boundary condition for the viscoelastic variable.

5. For nonisothermal flows, define the thermal boundary conditions.

   Thermal boundary conditions

   See *Boundary Conditions* (p. 173) and *Problem Setup* (p. 286) for details.

6. For free surface flows, define the remeshing method.

   Global remeshing

   See *Free Surfaces and Extrusion* (p. 311) for details.

7. Define the interpolation.

   Interpolation

   The motivation for using the simplified viscoelastic model is to quickly provide (qualitative) results on extrudate swelling, for both 2D and 3D flow simulations. Hence it is reasonable to select the most computationally inexpensive interpolation for the calculated fields (velocity, pressure, viscoelastic variable, temperature). By default, linear interpolation is used for the velocity and the viscoelastic variable within the context of the simplified viscoelastic model. While other interpolations are available and can be invoked, they will lead to an increase in the calculation time.

Chapter 12: Flow Induced Crystallization (FIC)

ANSYS POLYFLOW can be used to solve several types of viscoelastic flows. Features described in *Viscoelastic Flows* (p. 225) are interlinked with fluids whose structure is not affected by the flow. In some processes, the fluid structure is altered by the flow. This is especially true in fiber spinning processes, where crystallization takes place. For this purpose, a model has been developed by Doufas, McHugh, and Miller (see [*9*] (p. 715)).

This chapter introduces the basic equations for a single mode viscoelastic model that incorporates a flow induced crystallization (FIC) mechanism. It consists of the following sections:

12.1. Introduction

ANSYS POLYFLOW solves a broad range of applications where the fluid exhibits complex deformation properties. Various differential and integral models are presented in *Viscoelastic Flows* (p. 225), with a detailed description of problem setups.

For the models presented in *Viscoelastic Flows* (p. 225), and the generalized Newtonian models introduced in *Generalized Newtonian Flow* (p. 207), we assume that the morphology of the fluid remains unchanged. In some circumstances and for some fluids such as Nylon, the fluid undergoes some structural changes, because of the flow induced crystallization (FIC) mechanism.

A model has been recently proposed for describing the FIC [*9*] (p. 715). It involves a combined description of both amorphous and semicrystalline phases. It is based on a tensorial description of the configuration for the amorphous phase and of the macro-molecular orientation for the semicrystalline phase.

An understanding of the following features of the model may help a few to obtain the best possible results from this model, without unrealistic expectations.

- This model ([*9*] (p. 715)) is new, and may be improved in the future.
- Some alterations have been made to the model for a convenient software implementation and for controlling the solution under no flow (rest state). Hence, there will be slight deviations when you compare the original paper to this chapter.
- The model is based on a tensorial description with appropriate transport equations for crystallization. The strongly coupled, highly nonlinear resulting system can affect the calculation time and the convergence of the solver. Hence it requires a careful problem setup with appropriate evolution schemes.
- The paper (see [*9*] (p. 715)) proposes numerical values for all model parameters required for simulating FIC occurring in Nylon. An experimental protocol does not exist for the determination of numerical parameters. Hence, a possible user's database is the best source of information.

12.2. Crystallization Model

12.2.1. Qualitative Description

Consider an incompressible polymer melt, with coarse grains as a concentrated suspension of *n* nonlinear elastic dumbbell molecules per unit volume. In the flow, the total-viscoelastic, extra-stress tensor $\mathbf{T}$ can be written as:

$$\mathbf{T} = \mathbf{T}_a + \mathbf{T}_{sc} \tag{12–1}$$

where $\mathbf{T}_a$ and $\mathbf{T}_{sc}$ are the extra-stress contributions from the amorphous and semicrystalline phases, respectively. A purely Newtonian stress component can be added, which is given by

$$\mathbf{T}_n = 2\eta \mathbf{D} \tag{12–2}$$

where η is the shear viscosity, and $\mathbf{D}$ is the rate of deformation tensor.

The extra-stress tensor $\mathbf{T}$ must obey the momentum equation, which for a steady-state flow in the absence of volume forces, reduces to

$$-\nabla p + \nabla \cdot \mathbf{T} + 2\eta \nabla \cdot \mathbf{D} = 0$$

or

$$-\nabla p + \nabla \cdot \mathbf{T}_a + \nabla \cdot \mathbf{T}_{sc} + 2\eta \nabla \cdot \mathbf{D} = 0 \tag{12–3}$$

where *p* stands for the pressure. Constitutive equations are required for calculating the stress contributions from both amorphous and semicrystalline phases. Individual macromolecules may undergo a conversion process from amorphous to semicrystalline phases in this flow. This conversion is governed by a degree of transformation (*x*) that may depend on the current stress state in the flow.

The temperature *T* plays a significant role in the crystallization mechanism, and the associated energy equation incorporates contributions from crystallization (such as enthalpy). These four distinct components are discussed in the following sections.

12.2.2. Extra-Stress Contribution from the Amorphous Phase

Let $\mathbf{C}$ denote a dimensionless configuration tensor for the amorphous phase. The stress in the amorphous phase is obtained from $\mathbf{C}$ as follows:

$$\mathbf{T}_a = GE\left[\frac{3}{(1-x)}\mathbf{C} - \mathbf{I}\right] \tag{12–4}$$

where

$\mathbf{I}$ = unit tensor

G = zero shear modulus

x = degree of transformation

E = nonlinear spring force factor

E depends on the fractional extension of the chains, and is defined as:

$$E = \frac{1}{1 - \frac{\mathrm{tr}\,(\mathbf{C})}{N_0}} \tag{12–5}$$

where N_0 is the number of statistical links. The constitutive equation required for the configuration tensor $\mathbf{C}$ is given by:

$$\lambda_a \overset{\nabla}{\mathbf{C}} + \frac{1}{3\,(1-x)}\left[(1-\alpha)\,\mathbf{I} + \alpha\frac{3}{(1-x)}E\mathbf{C}\right] E\left[\frac{3}{(1-x)}\mathbf{C} - \mathbf{I}\right] = 0 \tag{12–6}$$

where ∇ stands for the upper-convected derivative used for objectivity reasons.

Equation 12–6 (p. 273) depends on the degree of transformation *x*. Two material parameters are invoked in the equation; the mobility factor α and a relaxation time λ_a. The latter depends on temperature *T* for nonisothermal flows. In this case, the recommended temperature dependence obeys the Arrhenius law, and is given by:

$$\lambda_a(T) = \lambda_{a,0} exp\left[\alpha_\lambda\left(\frac{1}{T - T_0} - \left(\frac{1}{T_a - T_0}\right)\right)\right] \tag{12–7}$$

where α_λ = coefficient that depends on the activation energy

T_a = reference temperature

T_0 = scaling factor required for a non-absolute temperature scale

In [*9*] (p. 715) the authors have suggested a modified Giesekus model to incorporate the concept of finite extensibility of the macro-molecular chains included in the definition of *E*.

It involves the evaluation of the reciprocal *Langevin function L(z)* which may give raise to numerical problems. The Langevin function is defined as $L(z) = coth(z) - \frac{1}{z}$ and the evaluation of its reciprocal is very tricky. Hence it is replaced by *Equation 12–5* (p. 273) which exhibits similar features.

Equation 12–6 (p. 273) for $\mathbf{C}$ is also slightly altered to directly obtain the result $\mathbf{C} = \frac{\mathbf{I}}{3}$ before developing crystals (at rest). In addition, the orientation tensor $\mathbf{C}$ presently used is dimensionless and is independent with respect to the number of statistical links N_0 and their length.

Simplifications have allowed deletion of the term *(1-x)* in the constitutive equation.

12.2.3. Extra-Stress Contribution from the Semicrystalline Phase

Let $\mathbf{Z}$ denote a dimensionless orientation tensor for the semicrystalline phase. In the absence of crystallization it has the value of $\frac{\mathbf{I}}{3}$. The stress in the semicrystalline phase is obtained from $\mathbf{Z}$ as follows:

$$\mathbf{T}_{sc} = 3G\left(\mathbf{Z} - \frac{\mathbf{I}}{3} + 2\lambda_{sc}\,(\nabla\;\mathbf{v})^{T} : <\mathbf{uuuu}>\right) \tag{12–8}$$

where G = shear modulus

$\mathbf{v}$ = velocity field

$\mathbf{u}$ = a unit vector along the axis of the crystallized rod

λ_{sc} = time constant

The shear modulus *G* is assumed to keep the same value as for the amorphous phase. The time constant λ_{sc} depends on the degree of transformation *x* and temperature *T*.

It is connected to the relaxation time λ_{a} of the amorphous phase with:

$$\lambda_{sc}(x, T) = \lambda_{sc,0}(T)\, exp\,(Fx) \cong c\lambda_{a}(T)\, exp\,(Fx) \tag{12–9}$$

where *c* and *F* are additional model parameters that have to be determined from experimental data. As observed λ_{sc} exhibits the same temperature dependence as λ_{a}.

The second equality in *Equation 12–9* (p. 274) is an approximation that is introduced in [*9*] (p. 715). It is motivated by the need to have a nearly vanishing relaxation time λ_{sc} for a vanishing degree of transformation, and that can be interpreted as the relaxation time of infinitesimal crystals. As you can see, λ_{sc} grows rapidly with the crystallization process.

The constitutive equation for the orientation tensor $\mathbf{Z}$ is given by:

$$\overset{\nabla}{\mathbf{Z}} = \frac{\sigma}{\lambda_{sc}(x, T)}\left(\mathbf{Z} - \frac{\mathbf{I}}{3}\right) - 2\,(\nabla\mathbf{v})^{T} : <\mathbf{uuuu}> \tag{12–10}$$

where σ is an anisotropic drag parameter that lies in the range $0 \leq \sigma \leq 1$. *Equation 12–10* (p. 274) is similar to the upper-convected Maxwell fluid model and hence may exhibit a similar numerical behavior. This equation depends on the degree of transformation *x* through the time constant λ_{sc} given by *Equation 12–9* (p. 274).

Equation 12–8 (p. 274) and *Equation 12–10* (p. 274) involve a complicated source term $(\nabla\mathbf{v})^{T} : <\mathbf{uuuu}>$. The quantity $<\mathbf{uuuu}>$ is a fourth-order tensor that would require solving 81 components if no further approximation is introduced. In [*9*] (p. 715), the following approximation is used:

$$(\nabla \mathbf{v})^T : <\mathbf{uuuu}> = (1-\omega)\left[\frac{-1}{35}\left(\nabla \mathbf{v} + (\nabla \mathbf{v})^T\right) + \frac{I}{7}\left(\left[(\nabla \mathbf{v})^T : \mathbf{Z}\right]\mathbf{I} + \mathbf{Z}\cdot\left(\nabla \mathbf{v} + (\nabla \mathbf{v})^T\right) + \left(\nabla \mathbf{v} + (\nabla \mathbf{v})^T\right)\cdot \mathbf{Z}\right)\right] + \omega\left[(\nabla \mathbf{v})^T : \mathbf{Z}\right]\mathbf{Z} \tag{12–11}$$

where

$$\omega = 1 - 27 det\,(\mathbf{Z}) \tag{12–12}$$

Due to the complexity of *Equation 12–11* (p. 275), an auxiliary second order tensor is introduced for evaluating the quantity $(\nabla \mathbf{v})^T : <\mathbf{uuuu}>$. This prevents multiple evaluations required in *Equation 12–8* (p. 274) for calculating $\mathbf{T}_{sc}$ and in *Equation 12–10* (p. 274) for calculating $\mathbf{Z}$.

In [*9*] (p. 715), the authors wrote the stresses in the semicrystalline phase in terms of a traceless orientation tensor $\mathbf{S}$. We have used an orientation tensor with a unit trace, connected to $\mathbf{S}$ with $\mathbf{Z} = \mathbf{S} + \frac{\mathbf{I}}{3}$ for convenience in software implementation. Some further simplifications in connection with the incompressibility assumption have been introduced.

12.2.4. Degree of Transformation

A constitutive equation is needed for the degree of transformation *x*. The following equation is suggested in [*9*] (p. 715):

$$\lambda_{av}\frac{Dx}{Dt} = m\,[-ln\,(1-x)\,]^{\frac{m-1}{m}}\,(R_{max} - x)\,exp\left(\xi\frac{\mathrm{tr}\,(\mathbf{T})}{G}\right) \tag{12–13}$$

where
ξ = dimensionless model parameter (usually equal to 0.06)

λ_{av} = a time constant
G = melt shear modulus
m = a constant (usually equal to 1)

This transport equation is strongly coupled with both amorphous and semicrystalline phases through the trace of $\mathbf{T}$.

The degree of transformation *x* is governed by a highly nonlinear advection equation. To reduce the level of nonlinearity in *Equation 12–13* (p. 275), select a unit value for *m*. At rest (quiescent condition), this equation indicates that *x* evolves towards R_{max}, which is seen as the upper limit of crystallization. To facilitate the convergence of the solver, you can apply an evolution scheme on the parameter R_{max}.

The time constant λ_{av} is the inverse of the Avrami constant under quiescent conditions and depends on the temperature in nonisothermal flows. In [*9*] (p. 715), the temperature dependence of the Avrami constant is described using a Gaussian function.

You can interpret the Avrami constant as an indication of the maximum crystallization rate. Consequently, an inverse Gaussian function is used for the temperature dependence of λ_{av}:

$$\lambda_{av}(T) = \lambda_{av,0} exp\left[-f \left(\frac{\left(T - T_\beta\right)}{\beta} \right)^2 \right] \tag{12–14}$$

where T_β is the temperature that corresponds to the peak in the Gaussian function, β is its width around T_β. The factor f takes the value of -1.

A large value of time constant means that the information is transported without any change. To prevent the sudden use of a large time constant in the calculation, an evolution scheme can be defined on f. The degree of transformation can change only when the temperature is close to T_β.

12.2.5. Energy Equation

For the simulation of flow induced crystallization under nonisothermal conditions, the energy equation must be added to the others. In the present context, it is given by the following equation where $\frac{DT}{Dt}$ is the material derivative:

$$\rho C_p \frac{DT}{Dt} = \nabla \, . \, [k \nabla T] + \mathbf{T}:\mathbf{D} + \rho \Delta H_f \phi_\infty \frac{Dx}{Dt} \tag{12–15}$$

In this equation, the left-hand side is the internal energy depending on the melt density ρ and heat capacity C_p. The right-hand side of the equation has the diffusion term characterized by the thermal conductivity *k*, the dissipation term, and a term related to the release of latent heat. The term related to the latent heat involves the heat of crystallization per unit mass ΔH_f and the average absolute degree of crystallinity of the system ϕ_∞.

It is assumed that the thermal conductivity *k* depends only on the temperature and is independent of the degree of transformation *x*. A third order polynomial expression is enabled for this temperature dependence:

$$k(T) = k_0 + k_1(T - T_0) + k_2(T - T_0)^2 + k_3(T - T_0)^3 \tag{12–16}$$

where, T_0 is a scaling temperature factor. For the heat capacity C_p, a significant dependence with respect to *x* is reported next to temperature dependence. Hence, you get the following expression for the mixed dependence of C_p:

$$C_p(T) = \left(1 - x\phi_\infty\right)(C_{a0} + C_{a1}(T - T_0) + C_{a2}(T - T_0)^2 + C_{a3}(T - T_0)^3)$$

$$+x\phi_{\infty}\left(C_{sc0}+C_{sc1}\left(T-T_0\right)+C_{sc2}\left(T-T_0\right)^2+C_{sc3}\left(T-T_0\right)^3\right) \tag{12–17}$$

The heat of crystallization per unit mass ΔH_f is obtained as:

$$\Delta H_f(T) = \Delta H_{f0} + \left(C_{a0}-C_{sc0}\right)\left(T-T_0\right) + \left(C_{a1}-C_{sc1}\right)\frac{\left(T-T_0\right)^2}{2}$$
$$+\left(C_{a2}-C_{sc2}\right)\frac{\left(T-T_0\right)^3}{3}+\left(C_{a3}-C_{sc3}\right)\frac{\left(T-T_0\right)^4}{4} \tag{12–18}$$

where ΔH_{f0} is the reference heat of fusion and T_0 is a scaling factor for the temperature. The energy equation is coupled with the degree of transformation *x* through the release of latent heat and its material properties. It is also coupled with the stresses in both amorphous and semicrystalline phases through the dissipation term $\mathbf{T}:\mathbf{D}$.

12.2.6. Boundary Conditions

Under nonisothermal conditions, besides the calculation of velocity and pressure, the simulation of FIC requires the evaluation of:

- two tensorial quantities ($\mathbf{C}$ and $\mathbf{Z}$)
- two scalar quantities (*x* and *T*)

These four quantities ($\mathbf{C}$, $\mathbf{Z}$, *x*, and *T*) obey partial differential equations and require appropriate boundary conditions. Under isothermal assumption, ignore temperature *T*. Presently we also ignore the quantity $\left(\nabla\mathbf{v}\right)^T:<\mathbf{uuuu}>$ since it is evaluated on the basis of an algebraic equation.

The equations that respectively govern $\mathbf{C}$, $\mathbf{Z}$, and *x* are hyperbolic of the first order. They require boundary conditions only at the inlet of the computational domain. Those inlet boundary conditions are automatically imposed when you select the inflow boundary condition. In particular, vanishing inlet conditions are imposed for *x*.

For the simulation of a flow involving FIC under nonisothermal conditions, evaluate the temperature field *T*. It obeys the energy equation, which is of the second order, so that conditions are required on all boundaries Within the present context of FIC, the most appropriate thermal boundary conditions are as follows:

- Temperature distribution at the inlet of the computational domain
- Heat flux along solid walls (possibly vanishing)
- Cooling conditions along free surfaces
- Zero flux along symmetry lines and planes
- Outflow conditions at the exit of the computational domain

Considering the expected temperature convection involved in the flow, do not assign a temperature at the exit of the computational domain. The equations are coupled to the momentum and incompressibility (mass conservation) equations. The momentum equation that governs the velocity field also requires conditions on all boundaries. They are expressed in terms of contact forces or velocity components.

Within the present context of FIC, the most appropriate flow boundary conditions are:

- Velocity distribution at the inlet of the computational domain (based on a flow rate)
- Zero velocities or slipping along solid walls
- Usual conditions along free surface
- Usual conditions along symmetry lines and planes
- Take-up velocity or force at the exit of the computational domain

12.2.7. Evolution Schemes

Under either isothermal or nonisothermal conditions, the equations involved in the simulation of FIC are strongly coupled and highly nonlinear. When considering the various equations and their boundary conditions, we can identify possible sources of nonlinearities.

In terms of boundary conditions, the most probable source of nonlinearity originates from the inlet flow rate and from the take-up velocity or force for the simulation of FIC in fiber spinning. Here, you can select appropriate evolution schemes, such as increasing ramps. Throughout the calculation, the increasing flow rate will lead to an increase of other nonlinearities, such as stress development, convection, and dissipation.

The flow rate should not vanish in the simulation of a steady free surface flow. In terms of flow and FIC governing equations, nonlinearities originates from the values of time constants (λ_a, λ_{sc}, λ_{av}). Appropriate evolution schemes with increasing functions should be selected. Nonlinearities may also originate from the temperature dependence of these parameters.

Appropriate evolution schemes with increasing functions can be invoked for the parameter α_λ in *Equation 12–7* (p. 273) and *f* in *Equation 12–14* (p. 276). It is also advised to apply an evolution scheme on the parameter R_{max} appearing in *Equation 12–13* (p. 275), to bound the development of the degree of transformation *x*, especially when λ_{av} is small.

12.2.8. Rheological Model and Properties

The model for flow-induced crystallization suggested here involves a long series of material parameters. They are currently available for a limited range of materials, while protocol measurements and techniques for parameter identifications still have to be developed.

The FIC model actually involves the evaluation of three tensors: $\mathbf{C}$, $\mathbf{Z}$, and an auxiliary tensor for $(\nabla \mathbf{v})^T : \langle \mathbf{uuuu} \rangle$. This can be computationally expensive, especially for 3D flows. Hence the present crystallization model is available in single-mode only. The use of a modified Giesekus model with finite chain extensibility is endowed with realistic rheological properties, such as shear thinning and normal stress differences.

12.2.9. Total Extra-stress Postprocessor

The primary unknowns calculated in a FIC simulation case do not explicitly involve stresses. The following quantities are calculated:

- configuration tensor ($\mathbf{C}$) for the amorphous phase
- orientation tensor ($\mathbf{Z}$) for the semicrystalline phase

- degree of transformation (x)
- temperature (T) in a nonisothermal simulation
- velocity ($\mathbf{v}$)
- pressure (p)

The quantity $(\nabla\mathbf{v})^T : <\mathbf{uuuu}>$ given by *Equation 12–11* (p. 275) is evaluated as an auxiliary tensor. Based on these fields, it is possible to calculate the total extra-stress tensor $\mathbf{T}$ developing in the flow.

By summing the various contributions from *Equation 12–2* (p. 272), *Equation 12–4* (p. 272), and *Equation 12–8* (p. 274), we obtain:

$$\mathbf{T} = \frac{G}{1 - \frac{\mathrm{tr}\,(\mathbf{C})}{N_0}}\left[\frac{3}{1-x}\mathbf{C} - \mathbf{I}\right] + 3G\left(\mathbf{Z} - \frac{\mathbf{I}}{3} + 2c\lambda_a(T)\,e^{Fx}(\nabla\mathbf{v})^T : <\mathbf{uuuu}>\right) + \eta\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right) \tag{12–19}$$

In the right side of the equation, you can identify all three contributions to the total extra-stress tensor. Successively, you can find the contribution of the amorphous phase, the contribution of the semicrystalline phase, and the purely Newtonian contribution.

That extra-stress tensor $\mathbf{T}$ has the same number of components as the other tensorial quantities, except for planar cases, where only three components are evaluated. The fourth one, perpendicular to the flow domain, can be evaluated by creating a function within the graphical postprocessor.

12.3. Problem Setup

The basic steps for setting up a flow simulation involving FIC features are as follows.

1. Create a sub-task for crystallization flow simulation (isothermal or nonisothermal)

   Create a sub-task

   a. Select the appropriate problem type.

      Isothermal crystallization

      or

      Non-isothermal crystallization

   b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. Define the material properties.

Material data

a. Select appropriate choices which are independent of the crystallization.

- For isothermal and nonisothermal FIC simulations, select the relevant menu item:

  Density

  Inertia terms

  Gravity

- For nonisothermal FIC simulations, select the relevant menu item:

  Thermal conductivity including the scaling temperature **t0** (T).

  Viscous heating

  Average temperature used for initializing the temperature field.

  Heat source per unit volume

b. Specify data for the crystallization mechanism:

Mat. Data for crystallization

i. For either isothermal or nonisothermal FIC simulation cases, successively, select the following:

G0 Shear modulus (G)

Visc Newtonian viscosity (η)

relax_a relaxation. time (amorphous) ($\lambda_{a,0}$)

relax_r inverse of Avrami constant ($\lambda_{av,0}$)

alpha mobility factor (α)

N0 nber of statistical links (N_0)

F growth factor for relax_sc (*F*)

c ratio of relax_sc to relax_a (*c*)

sigma anisotropic drag parameter (σ)

xsi model parameter for FIC (ξ)

m Avrami constant (*m*)

Rmax maximum degree of transformation (R_{max})

ii. For nonisothermal FIC simulations, specify a few additional data:

- For specifying the temperature dependence of λ_a, select:

  Temperature dependence for relax_a

  Select the appropriate temperature dependence, Arrhenius approximate law, or Arrhenius law. Arrhenius law is recommended with the corresponding parameters **alfa** (α), **talfa** (T_α) and **t0** (T_0).

- For specifying the temperature dependence of λ_{av}, select:

  Temperature dependence for relax_r

  and enter the parameters for the Gaussian function, **Tbeta** (T_β) and **beta** (β). Also specify the possible evolution function on **factor** *f*. This parameter **factor** should be negative and must not be smaller than -1.

- For specifying the maximum degree of crystallinity (ϕ_∞), select:

  phi degree of crystallinity

  Enter the numerical value of **phi** (ϕ_∞).

- For specifying the heat of fusion ΔH_{f0}, select:

  Hf heat of fusion

  Enter the numerical value of **Hf** (ΔH_{f0}).

- For specifying the heat capacity C_p, select:

  Heat capacity

  – Enter the numerical value of **Cpa0** to **Cpa3** (C_{pa0} to C_{pa3}) for the amorphous phase.

  – Enter the numerical value of **Cpsc0** to **Cpsc3** (C_{psc0} to C_{psc3}) for the semicrystalline phase.

  – Enter the value for the scaling temperature **t0** (T_0).

4. Define the flow boundary conditions.

Flow boundary conditions

Boundary conditions are required for the unknowns $\mathbf{C}$, $\mathbf{Z}$, and x, obeying partial differential equations of the first order. For a steady state flow, boundary conditions must be assigned at the inlet of the calculation domain. This is automatically done when selecting inflow boundary conditions at the inlet. For information see *Boundary Conditions* (p. 173).

5. For nonisothermal flows, define the temperature boundary conditions.

Thermal boundary conditions

Thermal boundary conditions can be given in terms of temperature or heat flux (possibly vanishing). At least one boundary condition must be given in terms of temperature for a steady state flow. For details refer to step 5 in *Using the Model* (p. 306).

6. Modify the interpolation scheme used for the stresses (optional).

Interpolation

For details see *Selecting the Interpolation* (p. 246). In the present context of isothermal or nonisothermal FIC simulations, quadratic interpolation is selected by default for $\mathbf{C}$ and $\mathbf{Z}$.

DEVSS scheme, with or without streamline upwinding method, can be selected in combination with a linear interpolation for the tensorial unknowns $\mathbf{C}$ and $\mathbf{Z}$.

12.3.1. Names of Variables in ANSYS POLYFLOW

Most of the calculated fields, such as $\mathbf{v}$ or p, have explicit names that do not require any explanations. The specific fields invoked in a FIC simulation have the following names:

- **Configuration** stands for $\mathbf{C}$
- **Orientation** stands for $\mathbf{Z}$
- **DEGREE_TRANSFORM** stands for x
- **GradV_UUUU** stands for the auxiliary tensor $(\nabla \mathbf{v})^T : <\mathbf{uuuu}>$

Chapter 13: Heat Transfer

ANSYS POLYFLOW can model nonisothermal flow, heat conduction in solids, and internal radiation. This chapter provides information about heat transfer in both fluid and solid regions.

13.1. Conduction and Convection

ANSYS POLYFLOW allows you to include the transfer of heat via conduction and convection within the fluid and/or solid regions in your model. In die design, for example, rheological and thermophysical properties of the melt and the thermal settings in the die can be important in obtaining a geometrically well-defined polymer product. The heat transfer calculation ensures that temperature-dependent material properties are accurately resolved; this is especially important when the shape and surface quality of the product are of critical importance. The temperature field at the die exit can also influence the swelling and drawing behavior of the product.

13.1.1. Theory

13.1.1.1. Basic Equations

For heat conduction in a solid region, ANSYS POLYFLOW solves the energy equation in the following form:

$$\rho c_p \frac{\partial T}{\partial t} = r - \nabla \cdot \mathbf{q} \qquad \textbf{(13–1)}$$

where

$\frac{\partial T}{\partial t}$ = time derivative of the temperature

ρ = density

c_p = specific heat capacity

r = heat generated per unit volume by external sources

$\mathbf{q}$ = heat flux

ANSYS POLYFLOW assumes that heat conduction is governed by Fourier's law:

$$\mathbf{q} = -k \nabla T \qquad \textbf{(13–2)}$$

where k is the thermal conductivity, which can be constant or temperature dependent. c_p can also be constant or temperature dependent.

When solving steady problems in the absence of heat transport in the solid,

$$\frac{\partial T}{\partial t} = 0 \tag{13–3}$$

so the heat conduction model only requires the single parameter k. For unsteady problems, you also need to supply ρ and c_p.

For nonisothermal generalized Newtonian flows, additional terms are included in the energy equation:

$$\rho c_p \frac{DT}{Dt} = r - \nabla \cdot \mathbf{q} + \mathrm{tr}\,(\sigma \mathbf{D}) \tag{13–4}$$

where σ = Cauchy stress tensor

$\mathbf{D}$ = rate-of-deformation tensor

and $\mathrm{tr}\,(\sigma \mathbf{D})$ is the sum of the diagonal terms of $\sigma \mathbf{D}$ (i.e., the trace operator).

$\frac{DT}{Dt}$ is the material derivative of the temperature:

$$\frac{DT}{Dt} = \frac{dT}{dt} + \mathbf{v} \cdot \nabla T \tag{13–5}$$

The energy equation for viscoelastic flow takes the following form:

$$\rho c_p \frac{DT}{Dt} = \mathbf{T} : \nabla \mathbf{v} + r - \nabla \cdot \mathbf{q} \tag{13–6}$$

where $\mathbf{T}$ is the extra-stress tensor and $\mathbf{v}$ is the velocity.

13.1.1.2. Heat Flux Boundary Conditions

A heat flux condition can be imposed on external boundaries of the domain. ANSYS POLYFLOW uses the following equation to compute the heat flux:

$$q = q_c + \alpha\,(T - T_\alpha) + \sigma\,(\,(T + T_0)^{4} - (T_\sigma + T_0)^{4}) \tag{13–7}$$

where q_c is a temperature-independent heat flux and α is the heat convection coefficient. The term q_c allows the heat flux to be defined as a constant.

The second term stands for heat exchange by convection; T is the temperature at the boundary and T_α is the reference temperature for the convective heat exchange. The last term represents the heat exchange by radiation as given by the Stefan-Boltzmann law:

$$q = \sigma \left((T + T_0)^4 - (T_\sigma + T_0)^4 \right) \tag{13–8}$$

Here, σ is the coefficient of radiation, which is equal to the Stefan-Boltzmann constant in the case of black bodies (i.e., 5.6704 × 10^{-8} W/m^2-K^4). For non-black bodies, σ is equal to the product of the Boltzmann constant and an emissivity. Currently in ANSYS POLYFLOW σ is assumed constant; the variation of emissivity with temperature is not available. T_σ is the reference temperature for the radiative heat exchange.

In its original formulation, the Stefan-Boltzmann law describes a relation between the heat flux and the absolute temperature. The temperature can also be specified relative to a nonzero reference temperature T_0.

13.1.1.3. Boussinesq Approximation for Density in Nonisothermal Flows

The Boussinesq approximation can be used instead of a constant density. This model treats density as a constant value in all solved equations, except for the buoyancy term in the momentum equation:

$$\left(\rho - \rho_0\right) \mathbf{g} \cong -\rho_0 \beta \left(T - T_\beta\right) \mathbf{g} \tag{13–9}$$

where ρ_0 is the (constant) density of the fluid, T_β is a reference temperature, and β is the thermal expansion coefficient. *Equation 13–9* (p. 285) is obtained by using the Boussinesq approximation $\rho = \rho_0 (1 - \beta \Delta T)$ to eliminate ρ from the buoyancy term. This approximation is accurate as long as changes in actual density are small.

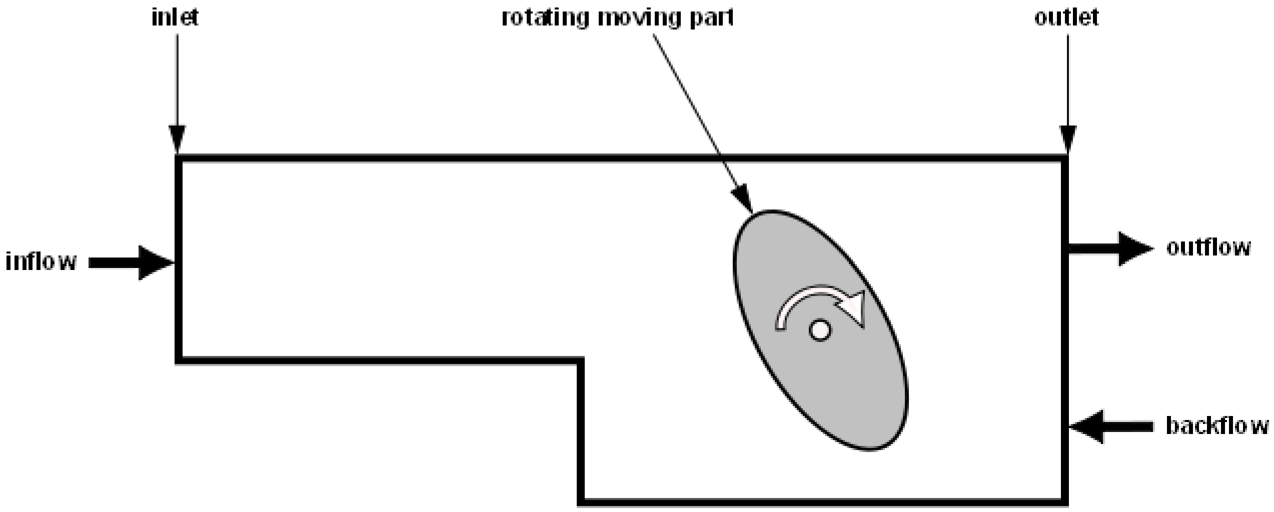
A typical application of the Boussinesq approximation is a glass furnace model with natural-convection-induced vortices.

13.1.1.4. Boundaries with Incoming and Outgoing Flows

Special consideration must be given to boundaries that experience both incoming and outgoing flows as part of a nonisothermal simulation. Typically, such circumstances arise in outlets of the flow domain through which some fluid enters as backflow, although it is possible that you may observe an inlet section that has a mix of entering and exiting velocities. In such circumstances you must impose a temperature on the incoming flow; this is necessary because the temperature must be fixed at the start of every pathline that begins on the border of a flow domain in order to maintain the stability of the calculation.

Outlets may experience backflow as a result of rotating moving parts (such as the screws in extruders), especially if only a part of the full flow domain is simulated, e.g., a few pitches of a full extruder. However, backflow can occur in cases that have no moving parts as well.

Figure 13.1 An Outlet with Backflow



The temperature is imposed on the incoming flow via a penalty formulation. The energy equation (*Equation 13–4* (p. 284) or *Equation 13–6* (p. 284)) for the nodes on the boundary is modified as follows:

$$\rho c_p \frac{DT}{Dt} = r - \nabla \cdot \mathbf{q} + (\sigma \mathbf{D}) + \alpha H(v_n - v_{n,\min})(T - T_{\text{imp}}) \tag{13–10}$$

where α = the penalty coefficient

v_n = the local normal velocity (positive if entering the flow domain)

$v_{n,\min}$ = the minimum normal velocity

T_{imp} = the imposed temperature

H = the Heaviside step function

The Heaviside step function is assigned a value based on the values of the local and minimum normal velocities:

$$H(v_n - v_{n,\min}) = \begin{cases} 0, & v_n < v_{n,\min} \\ 1, & v_n \geq v_{n,\min} \end{cases} \tag{13–11}$$

Note that no condition is imposed on the outgoing flow.

13.1.2. Problem Setup

13.1.2.1. General Procedure

The procedure for setting up a heat transfer problem is described below.

1. Create a sub-task for the heat transfer problem.

Create a sub-task

a. Select the appropriate problem type from the **Create a sub-task** menu.

   - For heat transfer in a solid region, select **Heat conduction problem**.

     Heat conduction problem

   - For a nonisothermal flow problem, select one of the following, depending on the type of flow being modeled:

     Generalized Newtonian non-isothermal flow problem

     Differential viscoelastic non-isothermal flow problem

     Integral viscoelastic non-isothermal flow problem

     Darcy non-isothermal flow problem

     Film model: Gen. Newtonian non-isothermal

     Film model: Viscoelastic non-isothermal

     Shell model: Gen. Newtonian non-isothermal

     Shell model: Viscoelastic non-isothermal

b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. Define the material properties.

   Material data

   - For a heat conduction problem, the material data that must be entered are the thermal conductivity (k in *Equation 13–2* (p. 283)) and the heat source per unit volume (r in *Equation 13–1* (p. 283)).

     Thermal conductivity

     Heat source per unit volume

     For time-dependent models, you will also need to define the specific heat capacity and density (c_p and ρ in *Equation 13–1* (p. 283)).

     Heat capacity per unit mass

Density

In addition, an average temperature can be entered to improve convergence in nonlinear problems.

Average temperature

The closer this temperature is to the solution, the better the convergence will be. If this temperature is too far from the solution, the numerical scheme might diverge. Note that, for time-dependent problems, this value is used as an initial temperature condition.

- For a nonisothermal flow problem, in addition to the flow properties (e.g., viscosity and density), you will need to define the thermal conductivity (k in *Equation 13–2* (p. 283)) and the specific heat capacity and heat source per unit volume (c_p and r, respectively, in *Equation 13–4* (p. 284) or *Equation 13–6* (p. 284)).

Thermal conductivity

Heat capacity per unit mass

Heat source per unit volume

By default, viscous dissipation is neglected in the energy equation. For some nonisothermal flows, it is preferable to solve the flow first without viscous dissipation, then add viscous dissipation and continue the calculation. To add viscous dissipation to the energy equation, select **Viscous heating** and choose the **Viscous heating will be taken into account** menu item.

Viscous heating

If you choose to use the Boussinesq approximation for density (see *Boussinesq Approximation for Density in Nonisothermal Flows* (p. 285)), you will need to specify the thermal expansion coefficient (β in *Equation 13–9* (p. 285)).

Coefficient of thermal expansion

You will also be prompted here to enter the reference temperature (T_β in *Equation 13–9* (p. 285)).

Important

For information about setting other flow properties (such as viscosity), see the procedure for the type of flow you are modeling (*Problem Setup* (p. 215) for generalized Newtonian flow, *Problem Setup for Differential Viscoelastic Flows* (p. 237) for differential viscoelastic flow, *Problem Setup for Integral Viscoelastic Flows* (p. 256) for integral viscoelastic flow).

4. Define the flow boundary conditions (nonisothermal flows only).

Flow boundary conditions

See the procedure for the type of flow you are modeling (*Problem Setup* (p. 215) for generalized Newtonian flow, *Problem Setup for Differential Viscoelastic Flows* (p. 237) for differential viscoelastic flow, *Problem Setup for Integral Viscoelastic Flows* (p. 256) for integral viscoelastic flow) for details about setting flow boundary conditions.

5. Define the thermal boundary conditions.

Thermal boundary conditions

a. Select the boundary for which you want to set thermal conditions.

b. Click **Modify**.

c. Select the boundary condition type you want to impose. For each external boundary, there are four possible conditions for a heat conduction problem, and five or six for a nonisothermal flow. For interface boundaries, the temperature and heat flux can be continuous or discontinuous across the boundary. By default, ANSYS POLYDATA imposes a zero temperature on all external boundaries, and continuous temperature and heat flux across interfaces.

- Choose **Temperature imposed** to specify the temperature on the boundary.

  Temperature imposed

  Select the appropriate specification method:

  – Select **Constant** to set a constant value for temperature.

  – Select **Linear function of coordinates** to specify a linear function of the form $T = A + Bx + Cy + Dz$ for the temperature.

  – Select **Map from CSV (Excel) file** to impose a temperature profile contained in a CSV file. Note that boundary conditions imposed via a CSV file are evaluated only once. In other words, if the CSV file contains a temperature distribution in space, the evaluation will be based on the geometry that is known at the beginning of the calculation, even if the boundary is moving.

  – Select **User Defined Function** to impose a temperature profile using a UDF.

- Choose **Flux density imposed** to specify the heat flux on the boundary.

  Flux density imposed

  The inputs are q_c (which can be constant or a linear function of the form $q_c = A + Bx + Cy + Dz$), α, T_α, σ, T_σ, and T_0 in *Equation 13–7* (p. 284). To specify T_σ as an absolute temperature, set the reference temperature T_0 to 0. Otherwise, set T_0 to the appropriate reference temperature and then specify T_σ relative to T_0. For example, to work in degrees Celsius, T_0 should be equal to 273.15; T_σ is the ambient temperature.

  Note that T_α and T_σ do not need to be the same. For example, radiation may be controlled by a far-field temperature T_σ, while natural convection occurs at the boundary with a fluid whose temperature is different from the far-field temperature.

  Since the Stefan-Boltzmann Law is a nonlinear function of temperature, it may require an evolution scheme. In this case, you can define an evolution scheme on the ambient temperature T_σ in such a way that T_σ changes progressively from the average temperature to its actual value. See *Evolution* (p. 517)

- Choose **Insulated boundary / symmetry** to specify an insulated boundary.

Insulated boundary / symmetry

No further inputs are required.

- Choose **Inflow** (available only for nonisothermal flows when the same boundary has been defined as an inflow boundary in the **Flow boundary conditions** menu) to assign a fully-developed inlet temperature profile for the boundary.

Inflow

ANSYS POLYFLOW will compute the profile automatically, so no further inputs are required.

- Choose **Outflow** (available for nonisothermal flow only; not available for heat conduction problems) for a vanishing conductive heat flux.

Outflow

No further inputs are required.

- Choose **Rosseland Correction** to use the Rosseland approximation for radiative heat transfer.

Rosseland Correction

This option is used primarily in glass melting problems. See *Radiative (Rosseland) Correction* (p. 487) for details. Note that the Rosseland approximation is a more computationally inexpensive alternative to the use of an internal radiation sub-task (as described in *Internal Radiation* (p. 293)), and should not be invoked for boundaries of a domain in which an internal radiation sub-task is defined.

- Choose **Inlet of periodic condition** to specify a periodic boundary for a periodic heat transfer problem.

Inlet of periodic condition

Periodic boundary conditions are used when the physical geometry of interest and the expected temperature-field pattern have a periodically repeating nature. This means that the heat fluxes across two opposite planes in your computational model are identical. Periodic boundary conditions can be applied to a pair of boundary sections, which are referred to as the inlet and outlet of the periodic condition. The temperature field and heat flux are continuous between the inlet and outlet.

The inputs for the periodic inlet are as follows:

i. In the resulting **Inlet of periodic cond.** panel, select the corresponding periodic boundary section (the outlet of the periodic condition) and click **Select**.

ii. Specify the transformation of coordinates and temperature field on the inlet to those along the outlet. There are two ways to do this:

   - Specify a rotation matrix and/or translation vector explicitly. For rotational periodicity, specify the rotation matrix.

Direct modification of rotation matrix

For translational periodicity, specify the translation vector.

Direct modification of translation vector

The rotation matrix and/or translation vector, once applied to the coordinates of a node along the inlet, will produce the coordinates of the corresponding node along the outlet.

- Specify pairs of corresponding points on the inlet and outlet: two points on the inlet and two points on the outlet for 2D, or three points on the inlet and three points on the outlet for 3D.

  Definition by source-target points

  ANSYS POLYFLOW will determine the proper rotation matrix and translation vector based on these sets of points, and then apply it to the rest of the points on the inlet to obtain the points on the outlet.

Note that, for axisymmetric problems, only translations along the z axis are allowed; no rotation or other translations are possible.

iii. Check the transformation to be sure it is consistent with the mesh.

Test of the transformation

iv. When you are satisfied with the inputs, accept the settings.

Accept current parameters

v. Specify the difference in heat flux between the inlet and the outlet.

Note that there are no inputs for the outlet of the periodic condition.

- Choose **Source of the connected condition** to specify the “source" boundary in a pair of non-conformal boundaries that need to be connected.

Source of the connected condition

Non-conformal boundary conditions are described in *Non-Conformal Boundaries* (p. 190). The inputs for non-conformal thermal boundary conditions are the same as those for flow boundary conditions, as described in *Connecting Non-Conformal Boundaries* (p. 191), except for the numerical parameters. The first two numerical parameters (element dilatation and amplitude of volume generation) are the same as for flow boundary conditions, but the stabilization factor for flow conditions is replaced by a smoothing factor for thermal conditions.

The smoothing factor controls the amount of heat flux that is allowed to be conducted through faces that are connected to only one of the connected boundaries (i.e., just to the source boundary or just to the target boundary). Ideally, there should be no conductive heat loss through these faces. The smoothing factor is therefore set to a very large value (causing the faces to become insulated walls with a uniform temperature), to avoid conductive heat loss.

Important

Changing the value of the smoothing factor from the default value is not recommended, except on the advice of your support engineer or ANSYS POLYDIAG. Note that setting this value to zero will destabilize the solution.

- Choose **Interface** (available only for interfaces between subdomains) to indicate whether the heat flux across the boundary should be continuous or discontinuous. (The temperature across the boundary is always continuous.)

  Interface

  By default, the heat flux is continuous. To specify discontinuous heat flux across the interface, you can define a nonzero value for the heat flux jump dq, which is computed the same way q is computed in *Equation 13–7* (p. 284). This can be used, for example, to simulate internal radiation. The inputs for a nonzero dq are the same as those listed above for the **Flux density imposed** condition.

- You can choose **Incoming fluid temperature** for boundaries that experience both incoming and outgoing flows (e.g., outlets of the flow domain through which some fluid enters as backflow). This option is only available for nonisothermal flow simulations, and imposes a temperature on the flow that enters the domain in order to maintain the stability of the simulation (as described in *Boundaries with Incoming and Outgoing Flows* (p. 285)).

  Incoming fluid temperature

  In the menu that opens, you can set parameters used in *Equation 13–10* (p. 286): the imposed temperature (T_{imp}), the minimum normal velocity ($v_{n,\mathrm{min}}$), and the penalty coefficient (α).

  A local temperature will be applied to the nodes of the boundary where the local normal velocity of the flow entering the domain is greater than or equal to the specified minimum normal velocity. Note the following when setting the parameters:

  - It is necessary to set $v_{n,\mathrm{min}}$ to a value that is greater than zero, in order to avoid a conflict with the boundary condition of the boundary. In general, it is recommended that you set $v_{n,\mathrm{min}}$ to a very small value relative to the dimension scale of your problem, so that it is close to zero; however, if you have concerns about either the assumptions inherent in simulating backflow or the results that you are obtaining, you have the option of setting $v_{n,\mathrm{min}}$ to the highest velocity that yet produces a stable calculation, thus minimizing the extent of the incoming flow on which temperatures are imposed.
  - The local temperature is calculated via a penalty formulation, so higher values for the penalty coefficient cause the local temperature to be closer to your specified imposed temperature. Note that if the penalty coefficient is too high, the simulation can become unstable. For this reason, it is recommended that you apply evolution on the penalty coefficient.
  - Applying evolution on the parameter(s) generating the incoming flow can be helpful.

For more information on setting boundary conditions, see *Boundary Conditions* (p. 173).

6. (optional) Modify the interpolation scheme for temperature.

   Interpolation

   See *Interpolation for Nonisothermal Flows* (p. 221) for interpolation guidelines for generalized Newtonian nonisothermal flows. (These guidelines apply to differential viscoelastic flow as well. For integral viscoelastic flow, you will not be able to modify the interpolation scheme for temperature.)

   It is also possible to define different interpolation types for temperature in different parts of the domain of the sub-task:

   a. Select **Sub-interpolation** at the bottom of the **Interpolation** menu.

      Sub-interpolation

   b. In the **Sub-interpolation** menu, select the subdomain on which you want to modify the interpolation

   c. Choose the type of interpolation you want on this subdomain.

   ANSYS POLYFLOW will automatically apply conformity constraints, so as to maintain the continuity of the temperature field across the interface together with continuity of the heat flux.

   Subdivided elements are mostly useful for solving flow problems in which advection takes place.

13.1.2.2. Using Evolution in Heat Conduction and Nonisothermal Flow Calculations

When modeling heat conduction in a solid, problems are usually linear and so it is generally not necessary to use an evolution scheme. Nonlinearities arise when a nonlinear temperature law is used for the conductivity or when radiation boundary conditions are used.

If there are convergence difficulties related to radiation, the use of an evolution scheme that increments the reference temperature T_σ with the evolution parameter S is recommended. See *Evolution* (p. 517) for information on evolution. Note that a convective term defined in a boundary condition does not make the problem nonlinear.

See *Using Evolution to Compute Generalized Newtonian Flow* (p. 222) for suggestions on using evolution for generalized Newtonian nonisothermal flow calculations. These guidelines are also appropriate for other types of flow.

13.2. Internal Radiation

When modeling a semitransparent medium, it can be important to account for radiation that is internal to the fluid. ANSYS POLYFLOW allows you to model internal radiation using the discrete ordinates (DO) model. The DO model requires a fixed geometry domain, in which the mesh encapsulates the radiative domain. Note that the DO model does not perform ray tracing.

The DO model can achieve more accurate results than those achieved by specifying the Rosseland correction at the boundary (see *Radiative (Rosseland) Correction* (p. 487)), but it is more expensive computationally. When you apply the DO model on a domain via an internal radiation sub-task, you should not apply the Rosseland correction to any of the boundaries.

13.2.1. Theory

Radiation equations are integral equations, and hence are more complex and expensive to solve than differential equations. However, if the angular dependency is assumed constant in the integral equation, radiation equations for a given direction ($\vec{s}$) reduce to a transport differential equation whose sole differential term represents the transport of the radiative flux along $\vec{s}$; the only other terms are source or sink terms. In order to take advantage of this reduction, ANSYS POLYFLOW allows you to discretize the space of the domain into a defined number of solid or planar angles, which then act as radiative directions. A partial differential equation (PDE) is then solved for each radiative direction. Specifying more radiative directions results in better discretization, but also makes the calculation more expensive. For example, if a 3D domain is discretized into six directions (which only provides three directions in each hemisphere), the simulation will basically require as many variables as a single-mode viscoelastic model.

Each of the defined angles in the discretized domain has a separate variable field, which can be referred to as a directional mode. The directional modes are generally uncoupled; however, there are two mechanisms that can cause them to be coupled:

- When scattering is taken into account, all of the directional modes are coupled.
- Boundary conditions (such as those for radiation) can couple otherwise unrelated directional modes. Radiation boundary conditions make the radiative flux change direction in cases where the emissivity of the boundary is not set to 1, as a part of the radiative flux is reemitted as diffuse radiation.

For the DO model, the radiative transfer equation is written as

$$\nabla \cdot (I(\vec{r},\vec{s})\,\vec{s}) + (a+\sigma_s)\,I(\vec{r},\vec{s}) = an^2\frac{\sigma T^4}{\pi} + \frac{\sigma_s}{4\pi}\int_0^{4\pi} I(\vec{r},\vec{s}\,')\,\Phi(\vec{s}\cdot\vec{s}\,')\,d\Omega' \tag{13–12}$$

where $\vec{r}$ = position vector

$\vec{s}$ = direction vector

$\vec{s}\,'$ = scattering direction vector

a = absorption coefficient (in units of L^{-1}, where 0 corresponds to empty space)

n = non-dimensional refractive index (where 1 corresponds to a vacuum)

σ_s = scattering coefficient (in units of L^{-1}, where 0 corresponds to no scattering)

σ = Stefan-Boltzmann constant (5.6704 × 10^{-8} W/m^2-K^4)

I = radiation intensity, which depends on position ($\vec{r}$) and direction ($\vec{s}$ or $\vec{s}\,'$)

T = local temperature

Φ = phase function

Ω' = solid or planar angle

The scattering function is represented by $\Phi(\vec{s}\cdot\vec{s}\,')$, which is only available in the Delta-Eddington form:

$$\Phi(\vec{s} \cdot \vec{s}\,') = 2f\delta(\vec{s} \cdot \vec{s}\,') + (1-f)(1 + C\vec{s} \cdot \vec{s}\,') \qquad \textbf{(13–13)}$$

In the previous equation, f is the forward-scattering factor and $\delta(\vec{s} \cdot \vec{s}\,')$ is the Dirac delta function. The f term essentially cancels a fraction f of the out-scattering; thus, for $f = 1$, the Delta-Eddington phase function will cause the radiation intensity to behave as if there is no scattering at all. C is the asymmetry factor. When the Delta-Eddington phase function is used, you will specify nondimensional values for f and C.

13.2.1.1. Angular Discretization

For 3D simulations, ANSYS POLYFLOW selects solid angles in a pseudo-optimal way. The software attempts to best cover all the angles of the domain with (nearly) equally spaced radiative directions. As the problem of optimally placing an arbitrary number of points on a sphere is unsolved in three dimensions, ANSYS POLYFLOW has adopted distributions that correspond to either the vertices, the mid-edge nodes, or the nodes central to the faces of two Platonic solids: a cube and an icosahedron. Such distributions allow you to divide your 3D domain into 6, 8, 12, 14, 20, or 30 discrete radiative directions; you are only required to select which one of these numbers you would like to use for the discretization (which is then designated as N_d).

To be clear, the method employed by ANSYS POLYFLOW to discretize the solid angles does not correspond to simply dividing the latitude and longitude into equal angular slices, as such a scheme would bias the radiative directions toward the “north" and “south" poles (i.e., toward the positive and negative z directions).

Because of scattering and boundary conditions, a coupled resolution approach is generally used in ANSYS POLYFLOW. By default, the number of radiative directions is set to 8 in order to limit the problem size.

For 2D simulations, the angular discretization is more straightforward: ANSYS POLYFLOW divides the planar domain into equal angular sectors, in accordance with the number of directions you specify. For 2D flows, the number of directions must be an even integer between 4–30.

13.2.1.2. Domain Boundaries

At domain boundaries, *Equation 13–12* (p. 294) requires that the incident radiation intensity (I_{in}) be calculated specifically for each incident radiative direction $\vec{s}$. The radiative direction is considered to be incident if $\vec{s} \cdot \vec{n} < 0$, where $\vec{n}$ represents the outward normal vector for the boundary. The discontinuous behavior of the boundary conditions as a function of the normal is one of the major reasons why radiation domains need to be fixed.

ANSYS POLYFLOW allows you to specify the following types of boundary conditions at domain boundaries:

- diffuse gray wall (DGW) You can specify that the boundary behaves as a diffuse gray wall of temperature T_w, where the incident radiative heat flux (q_{in}) is treated as follows: a fraction of the flux is absorbed by the wall and converted to heat; a fraction is transmitted through the wall; and the rest is reemitted within the domain, contributing to the radiative heat flux q_{out} that is emitted away from the wall in all directions as an outward radiation intensity I_0. The following are the equations used for the DGW boundary condition:

$$q_{in} = \int_{\vec{s} \cdot \vec{n} < 0} I_{in} \vec{s} \cdot \vec{n} \, d\Omega' \tag{13–14}$$

$$q_{out} = (1 - \tau_w) \left((1 - \varepsilon_w) q_{in} + n^2 \varepsilon_w \sigma T_w^4 \right) \tag{13–15}$$

$$I_0 = \frac{q_{out}}{\pi} \tag{13–16}$$

In the previous equations, you can specify the values for the following parameters:

- τ_w represents the transmittance of the wall, and is defined for individual boundaries of the domain. The transmittance determines the fraction of the original incident radiative heat flux that passes through the boundary (and is thereby lost to the sub-task), and ranges from 0 (for a completely opaque boundary) to 1 (for a completely transparent boundary, where the directional radiative heat flux is continuous in every direction). Physically, τ_w corresponds to the ratio of the area / length of “holes" versus the total area / length of the boundary.
- ε_w represents the emissivity of the wall, and is defined for individual boundaries of the domain. The emissivity determines the fraction of the remaining radiative heat flux (i.e., the flux that has not been transmitted through the boundary) that is absorbed by the boundary. The emissivity value ranges from 0 (for perfect reflection) to 1 (for a black body).
- n represents the refractive index of the medium adjacent to the wall (i.e., the ratio of the velocity of radiation in a vacuum to the velocity of radiation in the medium). The defined value is applied to all of the domain boundaries, and affects how much radiation is emitted by the boundaries as a result of absorption.

- insulated / symmetry

  You can set the boundary to act as an insulated wall. Such a boundary condition is the equivalent of the DGW boundary condition where the emissivity is set to 0 (i.e., no heating of the boundary) and the transmittance is set to 0 (i.e., completely opaque). All of the incident radiative energy is reflected away from the insulated wall in all directions.

  A boundary that behaves like an insulated wall is the equivalent of a symmetry boundary. Note that such a symmetry condition is only with respect to internal radiation, and does not apply to the flow.

- interface

  The interface boundary condition imposes continuity of the “irradiance" field through the boundary. In ANSYS POLYDATA, each radiative direction $\vec{s}$ has an associated scalar field of radiation intensity values; the entire set of these fields constitute the irradiance field. The interface condition allows you to connect different sub-tasks that model internal radiation without any losses due to absorption or reflection. The sub-tasks on either side of an interface boundary must have same number of radiative directions $\vec{s}$.

Important

ANSYS POLYFLOW does not model specular radiation, so all reflected radiation is diffuse.

13.2.1.3. Boundaries Internal to a Domain

A diffuse gray wall boundary can be modeled inside a domain for any boundary that is explicitly designated as a d dimensional PMesh, where d is one less than the number of dimensions for the domain (e.g., the PMesh must be 2D for a 3D domain). See *PMeshes* (p. 134) for information about PMeshes.

As is the case for the DGW boundaries described in the previous section, you can specify values for the transmittance, emissivity, and refractive index for the internal boundary. Internal boundaries are different than domain boundaries, however, in the following ways:

- You have the option of setting different emissivity values for the faces on each of the two sides of the internal boundary (i.e., the positive normal face and the opposing face), so that you can model a boundary where the absorption loss can vary depending on the direction of the incident radiative flux. The **Graphics Display** window identifies the positive normal face of the boundary via direction darts.
- You have the option of setting different refractive indices for the faces on each of the two sides of the internal boundary (i.e., the positive normal face and the opposing face), so that you can model the intersection of different media. As a consequence, different amounts of radiative energy (due to the absorption) will be emitted on either side of the boundary. The **Graphics Display** window identifies the positive normal face of the boundary via direction darts.
- The transmittance of the internal boundary does not represent a loss for the sub-task as a whole, as this fraction of the incident radiative heat flux is continuous on both sides of the boundary.

When the transmittance is less than 1, the values of the radiative heat flux on either side of DGW internal boundary will be discontinuous (though the discontinuities are smoothed out for graphic postprocessing). If you have a selected a PMesh as part of a DGW sub-model, it will exhibit this discontinuity by default; if transparent behavior is expected, you must be sure to explicitly set the transmittance value to 1.

13.2.2. User Inputs for Internal Radiation Model

In order to model internal radiation in your simulation, perform the following steps:

1. Generate an appropriate mesh for your internal radiation simulation, being sure to address the following considerations:

   - You must create separate subdomains for regions that have different radiation material data parameters.
   - If you intend to model diffuse gray wall (DGW) boundaries internal to the domain of an internal radiation sub-task, make sure that you have defined these boundaries as PMeshes (as described in *PMeshes* (p. 134)). The PMeshes must be d dimensional, where d is one less than the number of dimensions for the domain (e.g., the PMeshes must be 2D for a 3D domain). Note that the faces of PMeshes do not have an assumed or defined thickness, and hence are treated as perfectly thin.

   Important

   You must make sure that no PMeshes are on or extending through the exterior boundaries of the domain in which the internal radiation sub-tasks are defined. You can have a PMesh on (i.e., coincident with) a boundary that is an interface between two internal radiation subdomains, but if the PMesh extends through the interface it should be split into two separate PMeshes.

2. Launch ANSYS POLYDATA and create a task for the mesh created in the previous step.
3. Define a nonisothermal sub-task to model the flow and/or heat conduction in your problem.
4. Create a separate sub-task to calculate internal radiation, by clicking **Create a sub-task** in the task menu.

   Create a sub-task

   The **Create a sub-task** menu will open.

   a. Click **Internal radiation** for the sub-task type.

      Internal radiation

   b. When prompted, enter a name for the sub-task and click **OK**. The internal radiation sub-task menu will open.

5. Define the domain where the internal radiation sub-task applies (i.e., the domain for *Equation 13–12* (p. 294)), by clicking **Domain of the sub-task** in the sub-task menu.

   Domain of the sub-task

   The **Domain of the sub-task** menu will open.

   a. Select subdomains from the upper list and click **Remove** until it displays a group of subdomains that all share a common set of internal radiation parameters.
   b. Click **Upper level menu** to return to the internal radiation sub-task menu.

6. Specify the material data parameters for the internal radiation sub-task, by clicking **Material data** in the sub-task menu.

   Material data

   The **Material Data** menu will open.

   a. Click **Data for Internal Radiation Model** to open the menus that allow you to define the units and parameters.

      Data for Internal Radiation Model

   b. A panel will open to warn you if no system of units is known. Click **OK** to open the **Current System of Units** menu, where you can use the menu items and related panels to define the unit system for simulation. Your inputs allow ANSYS POLYDATA to automatically convert the Stefan-Boltzmann constant into the appropriate units.

      After you have completed your inputs, click **Upper level menu** to open the **Material data for internal radiation** menu.

   c. Define the parameters for angular discretization and the properties of the medium (see *Equation 13–12* (p. 294), *Equation 13–13* (p. 295), and *Equation 13–15* (p. 296)), by performing the following steps in the **Material data for internal radiation** menu:

i. Define the number of discrete radiative directions (i.e., N_d, the number of solid or planar angles into which the domain is discretized). The model will be more accurate if you specify higher values, but the calculation will be more expensive in terms of memory and CPU.

Modify Nd

Enter an even integer for **New value** in the panel that opens. For 2D simulations, you must enter an even integer between 4–30. For 3D simulations, enter one of the following integers: 6, 8, 12, 14, 20, or 30. Then click **OK**.

ii. Define the absorption coefficient of the medium.

Modify a

Enter a **New value** in the panel that opens and click **OK**.

iii. Define the refractive index of the medium.

Modify n

Enter a **New value** in the panel that opens and click **OK**.

iv. Define the scattering coefficient (σ_s).

Modify sigma_s

Enter a **New value** in the panel that opens and click **OK**.

v. If you do not want to use an absolute system for temperature, define the value for absolute zero in the current system for temperature.

Modify T0

Enter a **New value** in the panel that opens; for example, if you want to use the Celsius temperature scale instead of Kelvin, enter `-273.15`. Then click **OK**.

vi. Define the forward-scattering factor for the Delta-Eddington phase function.

Modify f

Enter a **New value** in the panel that opens and click **OK**.

vii. Define the asymmetry factor for the Delta-Eddington phase function.

Modify C

Enter a **New value** in the panel that opens and click **OK**.

d. Click **Upper menu level** repeatedly to return to the internal radiation sub-task menu.

7. Define the radiation boundary conditions on the boundary sets (or along intersections of subdomains) that form the boundary of the domain of the internal radiation sub-task, by clicking **Radiation boundary conditions** in the sub-task menu.

Radiation boundary conditions

The **Radiation boundary conditions** menu will open and allow you to perform the following steps:

a. Select a boundary from the list for which you want to modify the default radiation boundary condition.

b. Click **Modify** to open the **Radiation boundary condition along boundary <ID>** menu.

   i. Select the boundary condition type you want to impose from the following options:

      - To guarantee continuity of the irradiance field across the boundary, click **Interface**.

        ### Interface

        Note that this option is only available for boundaries that border the domain of another sub-task. You must make sure that the sub-tasks on either side of the boundary define it as an **Interface** and have the same number of radiative directions (defined using the **Modify Nd** menu item in step 6.(c)i.).

      - To specify that the boundary behaves like an insulated wall (which is equivalent to a radiative symmetry condition), click **Insulated boundary / symmetry**. This is the default selection for the axis of symmetry in axisymmetric tasks, and corresponds to a **Diffuse gray wall** condition where the values for emissivity and transmittance are set to 0, such that all incident radiative energy is diffusely reflected.

        ### Insulated boundary / symmetry

      - To specify a condition where a fraction of incident radiative energy is absorbed by the boundary and converted to heat, another fraction is lost through the boundary, and the rest is reflected, click **Diffuse gray wall**. This is the default selection, except as noted in the previous description.

        ### Diffuse gray wall

        The **Diffuse gray wall along boundary <ID>** menu will open. Click **Modify emissivity**.

        ### Modify emissivity

        A panel will open in which you can revise the **New value**, which defines the local value of the emissivity. The emissivity represents the fraction of radiative energy that is absorbed by the boundary. A value of 0 means that the incident energy that has not passed through the boundary due to the transmittance (as described in the description that follows) is entirely reflected in all directions, whereas a value of 1 means that it is all converted into heat. Click **OK** to return to the **Diffuse gray wall along boundary <ID>** menu.

        Click **Modify transmittance**.

        ### Modify transmittance

A panel will open in which you can revise the **New value**, which defines the local value of the transmittance. The transmittance represents the fraction of incident radiative energy that passes through the boundary and escapes from the domain of the internal radiation sub-task. Note that no radiative energy coming from outside the domain can enter through the diffuse gray wall boundary. A value of 1 for transmittance means that all of the incident energy is lost from the domain through transmittance, whereas a value of 0 means that it is all eligible for reflection and absorption. Click **OK** to return to the **Diffuse gray wall along boundary <ID>** menu.

ii. Click **Upper level menu** to return to the **Radiation boundary conditions** menu.

c. Repeat steps 7.(a)–7.(b)ii. for each additional boundary you want to modify.

d. Click **Upper level menu** repeatedly to return to the task menu.

8. Repeat steps 4.–7.(d) to create any additional internal radiation sub-tasks with unique sets of material data parameters.

9. If you want to set conditions on internal boundaries within the domain of an internal radiation sub-task, perform the steps that follow. Internal boundaries allow you to specify that a fraction of the incident radiative energy is absorbed by the boundary and converted to heat, another fraction passes through the boundary, and the rest is reflected.

   a. Click **Define sub-models** in the task menu to open the **Define sub-models** menu.

      Define sub-models

   b. Click **Create a new sub-model** to open the **Create a sub-model** menu.

      Create a new sub-model

   c. Click **Diffuse gray wall imposed**.

      Diffuse gray wall imposed

      Specify a name for the sub-model by revising **New value**in the panel that opens, and click **OK**. The sub-model menu will open.

   d. Specify the PMeshes (created in step 1.) that define the domain of the sub-model by clicking **Domain of the sub-model**.

      Domain of the sub-model

      The **Domain of the sub-model** menu will open.

      i. Select PMeshes from the lower list and click **Add** until the upper list displays all of the internal boundaries that share a common set of diffuse gray wall parameters.

      ii. Click **Upper level menu** to return to the sub-model menu.

   e. Specify the parameters of the sub-model by clicking **Diffuse gray wall parameters**.

      Diffuse gray wall parameters

      The **Diffuse gray wall parameters** menu will open.

i. Click **Modify emissivity (along darts direction)**.

Modify emissivity (along darts direction)

A panel will open in which you can revise the **New value**, which defines value of the emissivity applied to the faces of the internal boundary that have a positive normal (as indicated by the direction darts in the **Graphics Display** window). The emissivity represents the fraction of radiative energy that is absorbed by the boundary. A value of 0 means that the incident energy that has not passed through the boundary due to the transmittance (as described in the description that follows) is entirely reflected in all directions, whereas a value of 1 means that it is all converted into heat. Click **OK** to return to the **Diffuse gray wall parameters** menu.

ii. Click **Modify emissivity (opposite to darts direction)**.

Modify emissivity (opposite to darts direction)

A panel will open in which you can revise the **New value**, which sets the value of the emissivity (as defined in the previous description) applied to the faces of the internal boundary that oppose the positive normal. The opposing faces are those that do not display direction darts in the **Graphics Display** window. Click **OK** to return to the **Diffuse gray wall parameters** menu.

iii. Click **Modify refractive index (along darts direction)**.

Modify refractive index (along darts direction)

A panel will open in which you can revise the **New value**, which defines value of the refractive index applied to the faces of the internal boundary that have a positive normal (as indicated by the direction darts in the **Graphics Display** window). The refractive index represents the ratio of the velocity of radiation in a vacuum to the velocity of radiation in the medium adjacent to these faces of the boundary. A higher value increases the amount of radiation that is emitted by the boundary due to absorption. Click **OK** to return to the **Diffuse gray wall parameters** menu.

iv. Click **Modify refractive index (opposite to darts dir)**.

Modify refractive index (opposite to darts dir)

A panel will open in which you can revise the **New value**, which sets the value of the refractive index (as defined in the previous description) applied to the faces of the internal boundary that oppose the positive normal. The opposing faces are those that do not display direction darts in the **Graphics Display** window. Click **OK** to return to the **Diffuse gray wall parameters** menu.

v. Click **Modify transmittance**.

Modify transmittance

A panel will open in which you can revise the **New value**, which defines the local value of the transmittance. The transmittance represents the fraction of incident radiative energy that passes through the internal boundary to the other side. A value of 1 for transmittance means that all of the incident energy passes through, whereas a value of 0 means that all the incident energy is eligible for reflection and absorption. Click **OK** to return to the **Diffuse gray wall parameters** menu.

f. Click **Upper level menu** repeatedly to return to the **Define sub-models** menu.

g. Repeat steps 9.(b)–9.(e)iii. to create any additional sub-models with unique sets of diffuse gray wall parameters.

10. Click **Upper level menu** repeatedly to return to the task menu, and continue defining the task as necessary.

Chapter 14: Porous Media

ANSYS POLYFLOW allows you to include porous media in your problem. This chapter provides information about modeling porous media.

14.1. Introduction

In ANSYS POLYFLOW, the Darcy model is used to calculate flows involving porous media, such as the flow of oil in sand, or of a polymer through a mat. Sub-tasks of the Darcy type are available in 2D and 3D, in steady-state and time-dependent modes. Although multi-material simulations can be performed by defining several sub-tasks, the description in this chapter is limited to a single sub-task, since the procedure follows the general principle of assembling several sub-tasks in a single task.

Important

Currently in ANSYS POLYFLOW sub-tasks of the Darcy type must be solved in fixed domains. That is, free surfaces and moving interfaces cannot be included in a Darcy flow calculation.

14.2. Theory

A solid porous medium is characterized by two quantities: the volumetric void fraction ε (the ratio of the volume of void to the total volume of the porous media), which can range from 0 to 1, and the permeability factor $\mathbf{K}$. For isotropic porous media, $\mathbf{K}$ is a scalar. For non-isotropic porous media (i.e., when the physical structure of the solid body exhibits a particular orientation) it is a tensor, which indicates a preferred orientation of the pores.

Important

For highly non-isotropic materials, instead of specifying zero values for the permeability coefficient in the transverse direction, it is strongly recommended that you specify very small nonzero values (e.g., 10^{-12}) to improve the convergence of the solver.

The flow of an incompressible fluid with viscosity η in the porous material obeys the following constitutive equation:

$$v = -\frac{1}{\eta}\mathbf{K}\cdot\nabla p \tag{14–1}$$

where v is the fluid velocity and p is the pressure field. The conservation equation can be written as

$$\nabla \cdot \left[\frac{1}{\eta} \mathbf{K} \cdot \nabla p \right] = 0 \tag{14–2}$$

The pressure field is the only unknown associated with this equation. The boundary conditions can therefore be specified in terms of pressure or velocity (according to *Equation 14–1* (p. 305)). *Equation 14–2* (p. 306) holds for both isothermal and nonisothermal flows. When nonisothermal effects are taken into account, however, the energy equation must be solved as well. The energy equation results from the combined thermal effects in both the fluid in motion and the fixed solid porous medium. As a result, other material properties, in addition to ε, $\mathbf{K}$, and η, are required for calculating the temperature field.

In this description, the subscripts s and f identify the solid and the fluid, respectively. The solid and the fluid are characterized by their densities (ρ_s, ρ_f), their thermal conductivities (k_s, k_f), and their heat capacities (c_{p_s}, c_{p_f}). Thermal conductivity and heat capacity can be made temperature-dependent using a third-order polynomial expression.

The energy equation is then

$$\left[(1-\varepsilon)\rho_s c_{p_s} + \varepsilon \rho_f c_{p_f} \right] \frac{\partial T}{\partial t} - \varepsilon \rho_f c_{p_f} \frac{1}{\eta} (\mathbf{K} \cdot \nabla p) \cdot \nabla T = r + \frac{1}{\eta} (\nabla p)^T \cdot \mathbf{K} \cdot \nabla p + \nabla \cdot \left[\left((1-\varepsilon) k_s + \varepsilon k_f \right) \nabla T \right] \tag{14–3}$$

In *Equation 14–3* (p. 306), the contributions of the solid and the fluid are weighted by $1-\varepsilon$ and ε, respectively, since it is assumed that the fluid fills the voids entirely. Also, since the solid is motionless, only the fluid contributes to the convective term and to viscous heating. Finally, a nonzero value of r can be used to introduce a heat source. For nonisothermal flows, the fluid viscosity η can be made to depend on T, according to the Arrhenius law or Arrhenius approximate law. See *Temperature-Dependent Viscosity Laws* (p. 212) for details. The viscosity cannot be made to depend on shear rate because the actual value of the shear rate is unknown.

The unknown field associated with the energy equation is the temperature T. The boundary conditions can therefore be specified in terms of temperature or heat flux.

In flow through porous media, ANSYS POLYFLOW automatically performs the calculation of the velocity field v (as a postprocessor) according to *Equation 14–1* (p. 305) and computes the stream function as given in *Stream Function* (p. 552).

14.3. Using the Model

When your ANSYS POLYFLOW model includes porous media, you need to enable the relevant models, supply boundary conditions, and input material properties. These inputs are described in this section.

14.3.1. General Procedure

The steps for using the Darcy flow model are as follows:

1. Create a sub-task for the Darcy flow model.

   Create a sub-task

   a. Select the appropriate problem type from the **Create a sub-task** menu.

      Darcy isothermal flow problem

      or

      Darcy non-isothermal flow problem

   b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. Set the material properties.

   Material data

   a. Define the parameters for the Darcy flow model.

      Porous medium and fluid viscosity

      i. Specify the permeability.

         Modification of Permeability

         By default, permeability is a scalar (i.e., the porous medium is isotropic), but you can specify a tensor permeability (i.e., non-isotropic porous medium) by selecting **Switch to tensor mode** before selecting **Modification of Permeability**.

         If the default scalar permeability is retained, you can simply enter a constant value for the permeability. If a tensor $\mathbf{K}$ is selected, you will need to enter the Cartesian components of K.

         In particular, for 2D flows, the components $\mathbf{K}_{xx}$, $\mathbf{K}_{yy}$, and $\mathbf{K}_{xy}$ are requested, while, for 3D flows, the components $\mathbf{K}_{xx}$, $\mathbf{K}_{yy}$, $\mathbf{K}_{zz}$, $\mathbf{K}_{xy}$, $\mathbf{K}_{yz}$, and $\mathbf{K}_{zx}$ are requested. Physical considerations require that $\mathbf{K}$ be positive-definite.

      ii. Specify the fluid viscosity.

          Modification of Fluid viscosity

      iii. Specify the void fraction of the porous medium.

           Modification of Void fraction

      iv. For nonisothermal flows, specify the thermal conductivity for the solid (k_s in *Equation 14–3* (p. 306)).

Modification of Solid conductivity

(The thermal conductivity of the fluid, k_f, is specified as described in *General Procedure* (p. 286).)

v. For nonisothermal flows, specify the specific heat capacity for the solid (c_{p_s} in *Equation 14–3* (p. 306)).

Modification of Solid heat capacity

(The heat capacity of the fluid, c_{p_f}, is specified as described in *General Procedure* (p. 286).)

vi. For nonisothermal flows, specify the density of the solid (ρ_s in *Equation 14–3* (p. 306)).

Modification of Solid density

(The heat capacity of the fluid, c_{p_f}, is specified as described in *General Procedure* (p. 286).)

b. Specify any additional material properties required for nonisothermal flows. See *General Procedure* (p. 286) for details.

4. Define the flow boundary conditions.

Pressure boundary conditions

a. Select the boundary for which you want to set flow conditions.

b. Click **Modify**.

c. Select the boundary condition type you want to impose. For each boundary, there are three or four possible conditions. By default, ANSYS POLYDATA imposes a zero pressure on all boundaries.

- Choose **Pressure imposed** to specify the pressure on the boundary.

  Pressure imposed

  Select the appropriate specification method:

  – Select **Constant** to set a constant value for pressure.

  – Select **Linear function of coordinates** to specify a linear function of the form $p = A + Bx + Cy + Dz$ for the pressure.

  – Select **Map from CSV (Excel) file** to impose a pressure profile contained in a CSV file.

  – Select **User Defined Function** to impose a pressure profile using a UDF.

- Choose **Normal velocity imposed** to specify the normal velocity at the boundary.

  Normal velocity imposed

  Select the appropriate specification method:

  – Select **Constant** to set a constant value for normal velocity.

  - Select **Linear function of coordinates** to specify a linear function of the form $v_n = A + Bx + Cy + Dz$ for the normal velocity. A positive normal velocity represents an outflow.
  - Select **Map from CSV (Excel) file** to impose a normal velocity profile contained in a CSV file.
  - Select **User Defined Function** to impose a normal velocity profile using a UDF.
- Choose **Wall** to specify a zero-normal-velocity condition.

  Wall

  This is also the proper choice for an axis of symmetry.
- Choose **Interface between porous media** (available only for interfaces between sub-tasks) to guarantee continuity of the pressure field and normal velocity across the interface

  Interface between porous media

  or choose **Interface with fluid** (also available only for interfaces between sub-tasks) to guarantee continuity of the normal velocity and the normal force across the interface. The tangential velocity will vanish on the fluid side.

  Interface with fluid

  Important

  For a Darcy model, interfaces identify porous media that exhibit different material properties. All interfaces must therefore be fixed in space.

For more information on setting boundary conditions, see *Boundary Conditions* (p. 173).

5. For nonisothermal flows, define the thermal boundary conditions.

   Thermal boundary conditions

   See *Boundary Conditions* (p. 173) and *Problem Setup* (p. 286) for details.

Chapter 15: Free Surfaces and Extrusion

Extrusion is one of the most common applications for ANSYS POLYFLOW. Information about the free surfaces and moving interfaces that are involved in extrusion problems is presented in this chapter, along with information about the remeshing techniques that can be used for the changes to the mesh that occur during extrusion and related processes. See also *Blow Molding and Thermoforming* (p. 379) for information about free surface problems that include contact detection.

Note that Arbitrary Lagrangian-Eulerian (ALE) approach described in this chapter is a combined formulation; this is in contrast to the strictly Eulerian formulation used in the volume of fluid (VOF) model, which is the other means of simulating a free surface in ANSYS POLYFLOW. See *Introduction* (p. 443) for a comparison of the two methods.

This chapter discusses:

15.1. Introduction

Extrusion describes the process by which a polymer melt is pushed across a metal die, which continuously shapes the melt into the desired form. A variety of polymer products, including seals, tires, wires and cables, and tubes and pipes, can be extruded with suitable extruders and shaping methods at different operating conditions.

A wide range of extruders is available. The uniformity of the end product (the extrudate) will depend upon the thermal and physical properties of the polymer, its behavior under given operating conditions (e.g., inlet flow rate), and the die design.

For a given die design, you can use ANSYS POLYFLOW to analyze the extrusion at different operating conditions, in order to determine the optimal conditions at which the extrusion through that die should be performed. This is referred to as direct extrusion. Conversely, you can specify the desired shape of the extrudate, and have ANSYS POLYFLOW determine the shape of the die. This second case is referred to as inverse extrusion.

15.2. Theory and Equations

15.2.1. Free Surfaces

In an extrusion problem, where the shape of the extrudate is not known in advance, a free surface is used to represent the outer surface of the extrudate. A free-surface problem involves a boundary whose position is computed as part of the solution, since it is not known in advance.

Important

You can also model a free surface on a fixed mesh, using the volume of fluid (VOF) model (see *Volume of Fluid (VOF) Model* (p. 443)). The VOF model may be more appropriate for flows where the free surface interface is not well defined, such as injection and filling problems.

Free-surface problems have additional degrees of freedom and additional equations, compared with fixed-boundary flow problems. In a fixed-boundary problem, all components of the velocity vector or the surface-force vector can be prescribed. It is also possible to prescribe one velocity component and the other components of the surface-force vector. Prescribing both the normal surface force *and* the normal velocity, however, would result in an ill-posed problem definition.

For a free-surface problem, the tangential surface force (1 component in 2D, 2 components in 3D), the normal force, and the normal velocity must *all* be prescribed. Two requirements must be satisfied: the dynamic condition and the kinematic condition.

In the absence of surface tension in a steady-state flow, the position of the free surface must be prescribed upstream of the free surface.

15.2.1.1. Dynamic Condition

The normal force $\mathbf{f}$ must be equal to zero, or more generally, be prescribed a given value $\mathbf{g}$:

$$\mathbf{f} = \mathbf{g} \tag{15–1}$$

where $\mathbf{g}$ is nonzero when an external force is applied on the free surface. For example, when an internal pressure is applied in a blow-molding application, $\mathbf{g}$ is not zero. This condition (*Equation 15–1* (p. 312)) is referred to as the dynamic condition. For free surfaces, this condition is prescribed as a Neumann boundary condition for the momentum equation.

15.2.1.2. Kinematic Condition

For steady-state flows, the normal velocity must be equal to zero:

$$\mathbf{v} \cdot \mathbf{n} = 0 \tag{15–2}$$

where $\mathbf{v}$ is the velocity vector and $\mathbf{n}$ is the normal vector to the free surface. *Equation 15–2* (p. 312) states that no mass crosses the free surface, in the case of a steady-state problem. This condition is referred to as the kinematic condition.

For time-dependent problems, the kinematic condition becomes

$$\left(\frac{\partial \mathbf{x}}{\partial t} - \mathbf{v} \right) \cdot \mathbf{n} = 0 \tag{15–3}$$

where $\mathbf{x}$ is the position of a node on the free surface.

Equation 15–3 (p. 312) states that a time-dependent free surface must follow trajectories *in the normal direction*, while the tangential displacement is not restricted. When *Equation 15–3* (p. 312) is implemented in a finite-element code, it is correct to describe it as the requirement that nodes located on a free surface are Lagrangian in the normal direction.

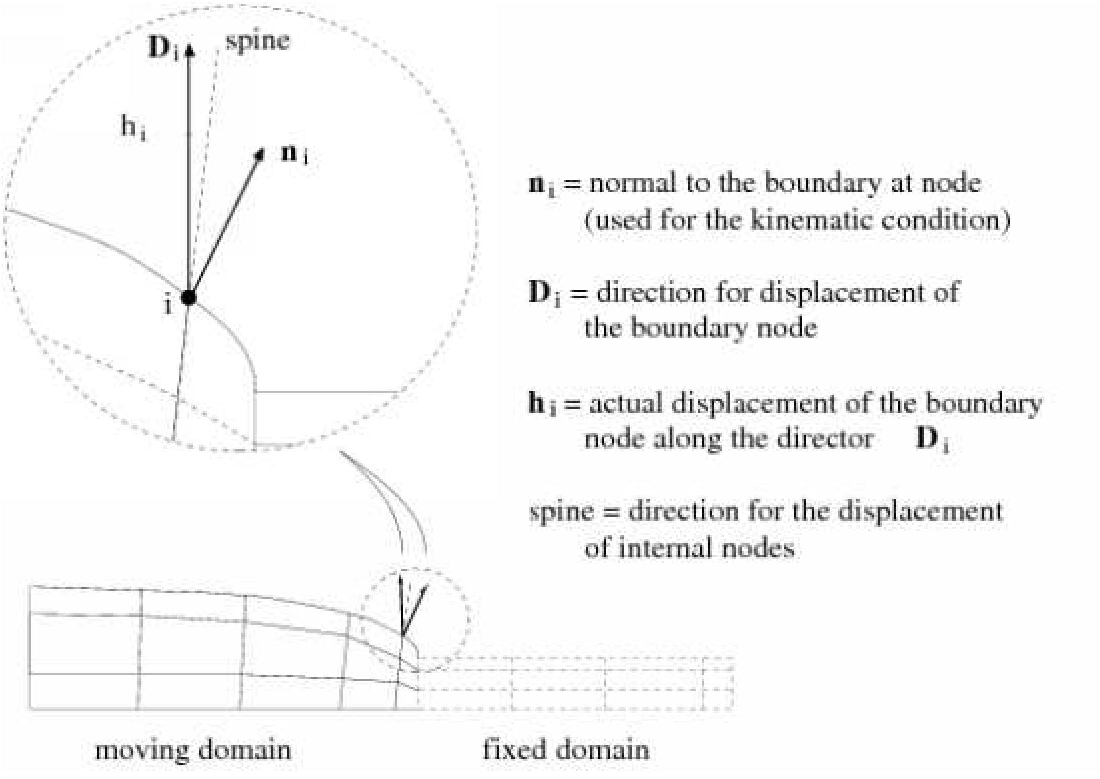
15.2.1.3. Geometrical Degree of Freedom

In order to be well-posed, a free-surface problem requires a new scalar degree of freedom, called the geometrical degree of freedom. The geometrical degree of freedom is denoted by h, which describes the amplitude of the displacement of boundary nodes in the normal direction. This mixture of a Lagrangian description in the normal direction and an Eulerian description elsewhere is sometimes referred to as an Arbitrary Lagrangian-Eulerian (ALE) formulation.

It can be shown that it is possible to satisfy *Equation 15–1* (p. 312) and *Equation 15–2* (p. 312) (for steady-state problems), or *Equation 15–1* (p. 312) and *Equation 15–3* (p. 312) (for time-dependent problems) provided that the direction in which boundary nodes are allowed to move *never becomes tangent* to the moving surface, which is evaluated as a function of the solution itself. If this condition is violated, ANSYS POLYFLOW will print an error message signaling a vanishing pivot for the geometrical degree of freedom. As the violation is approached (i.e., as the direction of the motion approaches the local tangential direction), the calculation may also diverge.

In ANSYS POLYFLOW, the direction of displacement for the boundary nodes ($\mathbf{D}$) is prescribed in advance, and the amplitude of the displacement is the geometrical degree of freedom (h), except for the line kinematic condition. $\mathbf{D}$ is referred to as the director. Although the director $\mathbf{D}$ is a given and not a variable of the problem, each node has its own director, and ANSYS POLYFLOW provides automatic tools for evaluating default directors. Unless otherwise specified, directors are the local normals to the initial mesh. *Figure 15.1* (p. 313) describes the relationship between the $\mathbf{D}$ and $\mathbf{n}$ directions. Note that $\mathbf{D}$ is different from the direction of a spine used for remeshing (see *Remeshing* (p. 343)).

Figure 15.1 Definition of Displacement, Spine, and Normal Directions



$\mathbf{D}_i$
spine
h_i
$\mathbf{n}_i$
i
$\mathbf{n}_i$ = normal to the boundary at node (used for the kinematic condition)
$\mathbf{D}_i$ = direction for displacement of the boundary node
$\mathbf{h}_i$ = actual displacement of the boundary node along the director $\mathbf{D}_i$
spine = direction for the displacement of internal nodes
moving domain
fixed domain

See *Directors* (p. 316) for details about directors.

15.2.2. Moving Interfaces

For coextrusion problems, there will be a moving interface between the two materials, in addition to the free surfaces for the outer shape of the extrudate. A moving interface is similar to a free surface, in that its position is computed as part of the solution. A free-surface condition is prescribed along a boundary set, whereas a moving-interface condition is defined along the topological interface between two subdomains. Flow problems must be defined in both subdomains, but the type of flow problem in each (generalized Newtonian, differential viscoelastic, or integral viscoelastic) can be different.

In the absence of surface tension, the position of the steady-state moving interface must be prescribed upstream of the moving interface.

15.2.2.1. Fixed-Interface Condition

A fixed-interface condition holds along the intersection between subdomain 1 and subdomain 2 when

$$\mathbf{v}_1 = \mathbf{v}_2 \tag{15–4}$$

and

$$\mathbf{f}_1 = \mathbf{f}_2 \tag{15–5}$$

In *Equation 15–4* (p. 314) and *Equation 15–5* (p. 314), neither the velocity vector nor the surface-force vector is prescribed; only the continuity of these quantities is required. These two conditions replace the dynamic condition (*Equation 15–1* (p. 312)). These equations do not guarantee that trajectories will not cross the interface; this condition must be added to the system. These equations define a fixed-interface problem, whereas a moving-interface problem occurs when a kinematic condition (*Equation 15–2* (p. 312) or *Equation 15–3* (p. 312)) is added to the system.

15.2.2.2. Dynamic Condition

For a free surface, the dynamic condition (*Equation 15–1* (p. 312)) is prescribed as a boundary condition for the momentum equation. For a moving interface, *Equation 15–4* (p. 314) and *Equation 15–5* (p. 314) are satisfied by the continuity of the velocity's finite-element interpolation across the interface. The pressure is generally *not* continuous across a moving interface, but the normal force *is* continuous.

15.2.2.3. Kinematic Condition

As for free-surface problems, ANSYS POLYFLOW assigns a director $\mathbf{D}$ to each node located on the interface, and the amplitude of the nodal displacement in the $\mathbf{D}$ direction is called the geometrical degree of freedom (h).

In a finite-element formulation, it is natural to associate the kinematic condition (*Equation 15–2* (p. 312) or *Equation 15–3* (p. 312)) with the variable h. Let ψ_i^h denote the shape function associated with the geometrical degree of freedom. Along the free surface or moving interface, the following equation must be satisfied:

$$\ll \mathbf{v} \cdot \mathbf{n} ;\ \psi'^{h}_{i} \gg\ = 0 \qquad \textbf{(15–6)}$$

for steady-state flows, and

$$\ll \frac{\partial \mathbf{x}}{\partial t} - \mathbf{v} \cdot \mathbf{n} ;\ \psi'^{h}_{i} \gg\ = 0 \qquad \textbf{(15–7)}$$

for transient flows, where $\ll\ ;\ \gg$ denotes the scalar product along the free surface, and $\psi'^{h}_{i} = \psi^{h}_{i}$ in the absence of upwinding. For consistent SUPG (streamline-upwind Petrov-Galerkin) schemes,

$$\psi'^{h}_{i} = \psi^{h}_{i} + k \frac{\mathbf{v}}{||\mathbf{v}||} \cdot \nabla \psi^{h}_{i} \qquad \textbf{(15–8)}$$

where k is the order of the element size.

15.2.2.4. Slipping Between Two Layers in Coextrusion

The constitutive relationship in *Equation 15–4* (p. 314) does not always hold in the presence of a fluid-fluid interface. It is sometimes necessary to assume that a different fluid velocity exists on each side of the interface, and that a Navier slip model dictates the interaction of one fluid with another.

Consider $\mathbf{v}_1$ and $\mathbf{v}_2$ to be the fluid velocity vectors on either side of the interface. For steady-state flow, immiscibility requirements impose a continuous normal velocity condition:

$$\mathbf{v}_1 \cdot \mathbf{n} = \mathbf{v}_2 \cdot \mathbf{n} \qquad \textbf{(15–9)}$$

while the following kinematic condition is also imposed:

$$\mathbf{v}_1 \cdot \mathbf{n} = 0 \quad \text{or} \quad \mathbf{v}_2 \cdot \mathbf{n} = 0 \qquad \textbf{(15–10)}$$

These conditions (*Equation 15–9* (p. 315) and *Equation 15–10* (p. 315)) are sufficient for the normal velocity Dirichlet condition and displacement of the interface.

The constitutive relationship for the tangential velocity can be stated as

$$\mathbf{f}_{12} \cdot \boldsymbol{\tau} = -\mathbf{f}_{21} \cdot \boldsymbol{\tau} \qquad \textbf{(15–11)}$$

where $\mathbf{f}_{12}$ is the friction force that fluid 1 exerts on fluid 2, $\boldsymbol{\tau}$ spawns the subspace of vectors, and

$$\mathbf{f}_{12} = k_N (\mathbf{v}_2 - \mathbf{v}_1) \qquad \textbf{(15–12)}$$

where k_N is a Navier coefficient.

Equation 15–9 (p. 315), *Equation 15–10* (p. 315), and *Equation 15–11* (p. 315) form a set of valid boundary displacement conditions for a fluid-fluid moving interface problem.

Because two velocity vectors must be calculated along the interface, convergence may be more difficult than in the continuous velocity case, and an evolution scheme (on the interface or flow rate) may be required. For robustness reasons, the fluid-fluid slipping method is implemented through a penalty technique.

15.2.3. Directors

For the sake of simplicity, consider a steady-state free-surface problem, with the force $\mathbf{f}$ on the free surface equal to zero. (This formulation can be generalized to a nonzero $\mathbf{f}$, and can also be extended to moving interfaces and time-dependent problems.)

In all directions, boundary conditions are expressed as

$$\ll \mathbf{f};\ \psi_i^h \gg\ = 0 \tag{15–13}$$

and in the normal direction, they are expressed as

$$\ll \mathbf{v}\cdot\mathbf{n};\ \psi_i^h \gg\ = 0 \tag{15–14}$$

or (for time-dependent problems)

$$\ll \mathbf{v}\cdot\mathbf{n} - \frac{\delta\mathbf{x}}{\delta t};\ \psi_i^h \gg\ = 0 \tag{15–15}$$

$\delta\mathbf{x}$ is the displacement of the node:

$$\delta\mathbf{x} = h_i\mathbf{D}_i \tag{15–16}$$

where $\mathbf{D}_i$ is the director at node i.

The fact that $\mathbf{v}$ and $\mathbf{n}$ are both unknowns in *Equation 15–14* (p. 316) and *Equation 15–15* (p. 316) makes free-surface problems nonlinear. ANSYS POLYFLOW will therefore need several iterations to converge, starting from an initial guess. In most cases, these iterations are of the Newton- Raphson type (i.e., velocity and position variables are updated at the same time), but it is also possible to use a decoupled scheme between velocity and position variables, as described at the end of *Convergence Strategies* (p. 327). In the decoupled case, *all* kinematic conditions, as well as all remeshing methods, are decoupled from the flow problem.

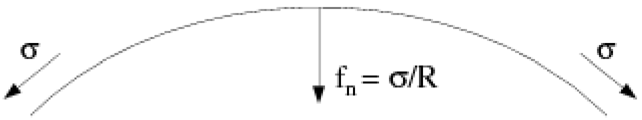
15.2.4. Surface Tension

Surface (or interfacial) tension plays an important role in free-surface and moving-interface problems. This section describes the phenomena associated with surface tension, and the equations used to model it in ANSYS POLYFLOW.

15.2.4.1. Surface Tension Force

Consider a curved surface located on the boundary, as shown in *Figure 15.2* (p. 317), and let $\mathbf{n}$ be the outward normal from the surface.

Figure 15.2 Surface Tension on a Free Surface



The surface tension force is a force of amplitude σ, acting *tangent* to the surface, whose net influence (through the vector sum on an elementary surface) is in the *normal* direction. This normal force (per unit area or length) f_n tends to reduce the surface curvature. The parameter σ is called the surface tension coefficient, and satisfies the following equation:

$$f_n \mathbf{n} = \frac{\sigma}{R}\mathbf{n} \tag{15–17}$$

R is the Gaussian curvature of the surface:

$$\frac{1}{R} = \frac{1}{R_1} + \frac{1}{R_2} \tag{15–18}$$

where R_1 and R_2 are the two principal radii of curvature (measured in orthogonal directions). For 2D axisymmetric domains, the surface curvature introduced by the symmetry about the z axis must also be taken into account at the outlet, so that f_n never becomes zero in an axisymmetric geometry when σ is nonzero.

In particular, even in the absence of applied external forces, a free jet is subject to a normal surface force given by *Equation 15–17* (p. 317) and *Equation 15–18* (p. 317).

In ANSYS POLYFLOW, surface tension forces are introduced on the right-hand side of the momentum equations when a nonzero surface tension coefficient is specified. Introduction of *Equation 15–17* (p. 317) in a finite-element code requires integration by parts (in 2D) or use of Green's theorem for Riemann's surfaces (in 3D) in order to use only first-order derivatives of the shape functions that characterize the geometry of the surface.

In 2D, integrating by parts the product of *Equation 15–17* (p. 317) and the velocity shape functions introduces tangential forces at both ends of the free surface. These forces describe the reaction that the “external world” exerts on the free surface in order to equilibrate the tensile force σ.

It can be shown that, in order to maintain the equilibrium with the tensile force on the free surface, this tangential force f_τ must be defined as

$$f_\tau \tau = \sigma\tau \tag{15–19}$$

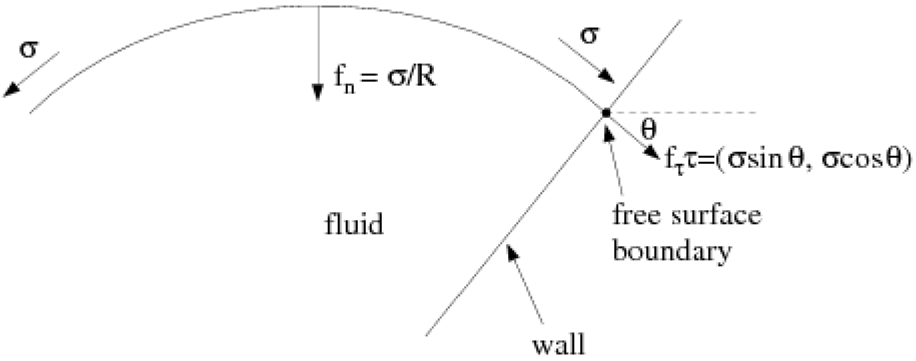
where τ is the unit vector tangent to the free surface and directed away from the surface (because the surface is in tension, not in compression).

An angle (θ) is used to describe the direction of τ:

$$\tau = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix} \quad (15–20)$$

Positive angles are measured counterclockwise with respect to a horizontal reference axis. In *Figure 15.3* (p. 318), θ is a negative angle (e.g., -30°). This angle allows ANSYS POLYFLOW to account for the reaction of the external world on the boundary of the free surface.

Figure 15.3 Surface Tension and Traction at the Extremities of a Free Surface



Surface tension can be modeled in 3D, but only if normal velocity condition has been imposed on the boundaries adjacent to the free surface, so that there is no need to define boundary conditions specific to the 3D surface tension model. Also, the angle θ cannot be specified for 3D problems.

15.2.4.2. Velocity Imposed on the Boundary of the Free Surface

An additional difficulty occurs when the flow velocity is prescribed at one edge of a free surface that includes surface tension, because the force term that describes the angle condition is overwritten by the velocity condition. If the position of the free surface at this point is known, it is necessary to suppress all possible free-surface displacements. If the position is unknown, a moving-contact-point problem is posed (see *Static and Dynamic Contact Points or Lines* (p. 329)). Also free-surface positions should be prescribed only where the velocity is also prescribed.

15.2.5. Discontinuity of the Normal Direction

3D extrusion problems require special attention along corner lines, which originate from sharp angles in the die lip section. Along these corners, the normal is actually discontinuous at the level of mesh discretization.

ANSYS POLYFLOW provides two techniques to solve this difficulty: separating the sheets of the free surface into several boundary sets, located between corner lines, or using the line kinematic condition, which does not require division into multiple boundary sets. The multiple-boundary-set method is referred to as the surface kinematic condition. Note that the line kinematic condition is not the default method, but it is recommended for complex profile extrusion.

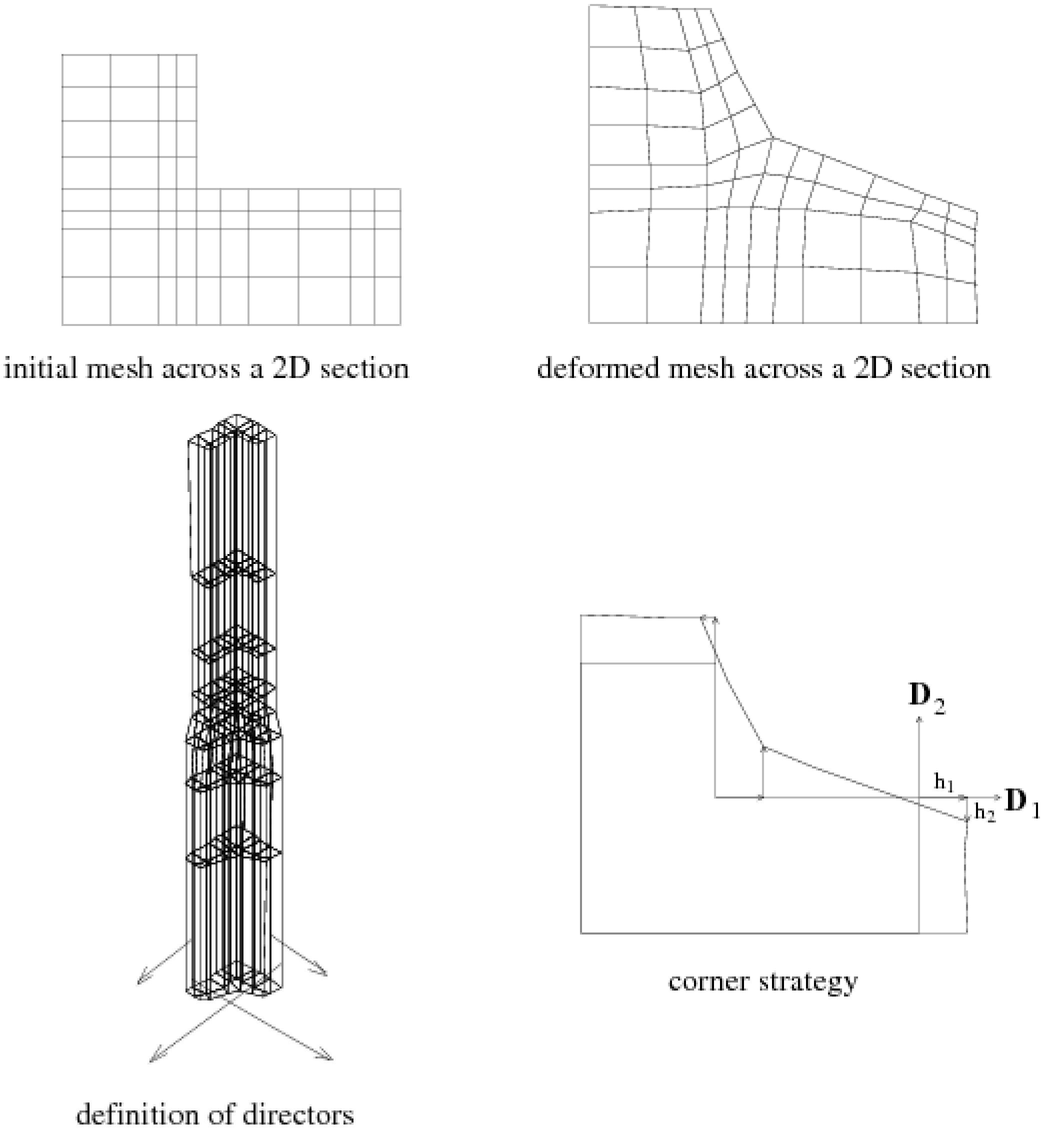
15.2.5.1. Corner Lines

Given that discontinuous slopes between finite elements will always occur for a nonplanar, free surface, the concept of a corner line may not be immediately clear. In fact, a corner exists when slope discontinuities do not disappear when the element size approaches zero (i.e., when the mesh is refined).

15.2.5.2. Surface Kinematic Condition

For the surface kinematic condition, it is convenient to consider two boundary sets. In this case, there will be two directors $\mathbf{D}$, two geometrical degrees of freedom h, and two kinematic conditions associated with each node of a corner line, because the node belongs to two boundary sets on which separate free-surface conditions are defined. Therefore, these special nodes have the freedom to move in a plane orthogonal to the main direction of extrusion. *Figure 15.4* (p. 319) shows an example for the surface kinematic condition.

Figure 15.4 Definition of a Separate Boundary Set for Each Sheet of the Free Surface (Surface Kinematic Condition)



initial mesh across a 2D section

deformed mesh across a 2D section

definition of directors

corner strategy

When this method is used, it is important for the mesh to have a sufficient number of boundary sets to allow for proper piecewise definition of the free surface around edges.

15.2.5.3. Line Kinematic Condition

For the line kinematic condition, the kinematic condition is written differently than in *Equation 15–14* (p. 316) or *Equation 15–15* (p. 316). Rather than requiring a zero velocity normal to the free surface, the velocity vector is required to be tangent to a series of lines included in the surface and starting from the die lip section. These lines generate the free surface.

This condition is stronger than the standard kinematic condition, because it allows each nodal position to be relocated in a plane *normal* to the direction of extrusion (i.e., in two directions rather than one). Because each point (rather than just the corner points, as in the multiple-boundary-set method) is relocated in two directions, there is no need to define multiple boundary sets (although it is possible to do so without affecting the solution). The number of position variables is slightly higher with the line kinematic condition than with the surface kinematic condition.

The line kinematic condition is the recommended numerical scheme for complex profile extrusion (direct or inverse), and it can only be used with the Optimesh, streamwise, or improved elastic remeshing techniques described in *Remeshing* (p. 343).

15.2.6. Drag

Fiber spinning process may involve high take-up velocity. It is not unusual that drag force originating from the air surrounding the free boundary of the fiber becomes significant with respect to the take-up force. When simulating fiber spinning process with a 2D axisymmetric model, it is possible to include the drag force into the calculation. From the point of view of modeling, the vector of drag force density $\mathbf{f}_{drag}$ is given as:

$$\mathbf{f}_{drag} = -0.5\frac{p_1}{\mathrm{Re}_{gas}^{p_2}}\rho_{gas}V^2\hat{\mathbf{v}} = -0.5\frac{p_1}{\left(\frac{RV}{\nu_{gas}}\right)^{p_2}}\frac{M_{gas}P}{R_u\left(T + T_{ref} + T_0\right)}V^2\hat{\mathbf{v}} \qquad \textbf{(15–21)}$$

In the equation above, p_1 and p_2 are independent parameters, while Re_{gas} and ρ_{gas} are the local Reynolds number and the density of the surrounding gas, and V is the magnitude of the local velocity on the surface of the fiber. The quantity $\hat{\mathbf{v}}$ denotes a unit vector parallel to the local velocity. The Reynolds number Re_{gas} is evaluated on the basis of the local fiber radius R, the magnitude V of the fiber velocity, and the kinematic viscosity ν_{gas} of the surrounding gas. The density ρ_{gas} of the gas is evaluated from the law of perfect gases, that involves the mole mass M_{gas} of the gas, the local pressure P, the universal thermodynamic molar gas constant R_u and the absolute temperature. For the temperature, reference temperature T_{ref} and offset temperature T_{0f} can be defined as well.

From the point of view of the physics, the relationship indicates that the intensity of the drag force increases with the density of the surrounding gas and with its velocity as well. The quadratic dependence of the drag force is affected by the local Reynolds number.

15.3. User Inputs for Free Surfaces and Moving Interfaces

15.3.1. General Procedure

The inputs directly related to the free surface or moving interface are provided in this section. You will need to set up the rest of the problem (material properties, other boundary conditions, etc.) as usual. See also *Defining the Direction of Motion* (p. 325) – *Constraint on Global Displacement* (p. 335) for guidelines about free-surface and moving-interface problems.

Note that you have access to POLYFLOW project templates, which are Workbench project files that you can modify in order to quickly and easily set up your own problem. These templates include extrusion problems. See POLYFLOW Project Templates in the POLYFLOW in Workbench User's Guide for further details.

1. Specify which boundary is the free surface or moving interface.

   - For a free surface, follow the steps below:

     a. In the **Flow boundary conditions** menu, select the boundary set that represents the free surface and click **Modify**.

        ≣ **Flow boundary conditions**

     b. Select **Free surface** as the boundary type.

        ≣ **Free surface**

   - For a moving interface, follow the steps below:

     a. In the **Flow boundary conditions** menu, select the boundary set that represents the moving interface and click **Modify**.

        ≣ **Flow boundary conditions**

     b. Select **Interface** as the boundary type.

        ≣ **Interface**

     c. Enable the motion of the interface.

        ≣ **Switch to moving interface**

     d. If you want to include the slip effect between the two fluids along the interface (as described in *Slipping Between Two Layers in Coextrusion* (p. 315)), turn on the fluid-fluid slip effect.

        ≣ **Specify fluid/fluid slip effect**

        (You will need to click **Yes** to confirm that you want to proceed.)

        i. Select the Navier form of the slip law (*Equation 15–12* (p. 315)).

           ≣ **F(v)= Generalized Navier's law**

        ii. Specify the friction coefficient for the Navier slip law (k_N in *Equation 15–12* (p. 315)).

           ≣ **Modify k**

        iii. Click **Upper level menu** twice to return to the **Interface** menu.

> **Important**
>
> Note that there is also an **Advanced options** menu, where you can modify the value of ε, a parameter that influences the penalty formulation for the slip effect. Changing the value of ε from the default value is not recommended, except on the advice of your support engineer or ANSYS POLYDIAG.

e. Move to the menu where you can specify additional parameters for the moving interface (described in the next few steps).

Specify moving interface parameters

2. If you want to model surface tension on the free surface or moving interface, specify a nonzero value for the surface tension coefficient (σ in *Equation 15–17* (p. 317)).

Surface tension

3. Specify the conditions to be imposed at the boundary of the free surface or moving interface.

Boundary conditions on the moving surface

The inputs will be slightly different, depending on whether surface tension is being modeled.

- If you are not modeling surface tension (i.e., if you did not specify a nonzero surface tension coefficient in step 2, above), the inputs will be as follows:
  a. Select the adjacent boundary or subdomain on which the position of the free surface is to be imposed.
  b. Click **Modify**.
  c. Select **Position imposed**.
- If you are modeling surface tension (i.e., if you specified a nonzero surface tension coefficient in step 2, above), the inputs will be as follows:
  a. Select the adjacent boundary or subdomain (or the first one if there are multiple choices).
  b. Click **Modify**.
  c. Indicate whether you want to impose the position or the angle at this boundary:
    - To impose an angle at the boundary (for modeling the traction force), select **Angle imposed**, and enter the value of θ in *Equation 15–20* (p. 318), when prompted.
    - To impose the position of the free surface at the boundary, select **Position imposed**.
  d. If there are multiple adjacent boundaries or subdomains, repeat the steps above until conditions for all of them have been set.

Important

Specifying a zero-force boundary condition (e.g., an outflow) for a 2D axisymmetric problem will violate the requirement that the normal force cannot be equal to zero in an axisymmetric geometry when the surface tension coefficient σ is nonzero (see *Surface Tension* (p. 316)). For both generalized Newtonian and viscoelastic flows, you can alleviate this problem by imposing a constant exit velocity instead of using an outflow boundary.

4. (free surfaces only) Specify the normal force on the free surface ($\mathbf{f}$ in *Equation 15–1* (p. 312)).

   Normal force

5. By default, directors for the free surface ($\mathbf{D}_i$) are normal to the initial mesh, and are calculated before each step of a time-dependent or evolution problem. If you prefer, you can prescribe the components of the director yourself and/or have the directors calculated just once for a time-dependent or evolution problem. See *Defining the Direction of Motion* (p. 325) for guidelines and further explanation.

   Direction of motion

   a. Select which condition you want to impose (along the whole surface, or on the intersection with an adjacent boundary).
   b. Click **Modify**.
   c. Modify the constraint on the appropriate component(s) of the direction vector by selecting the corresponding menu item and entering the desired value when prompted.
   d. When you are satisfied, click **Accept the current condition**. If you change your mind and want to return to the default (no condition imposed), select **Deletion of the current condition**.
   e. (evolution and time-dependent problems only) By default, ANSYS POLYFLOW will compute the director $\mathbf{D}$ before each evolution or time step of the task calculation (**D calculated before each step**). If you want the director $\mathbf{D}$ to be computed just once at the beginning of the task calculation, click the button next to **D calculated before each step**. The condition will now be **D calculated once for the whole task**.
   f. When you are satisfied with all the settings for the direction of motion, click **Upper level menu** at the top of the **Imposing the direction of surface motion** panel.

6. By default, no upwinding is used for the calculation of the kinematic equation. If you want to include it, select the **Upwinding in the kinematic equation** option.

   Upwinding in the kinematic condition

   Invoking upwinding is generally recommended for steady state or evolution calculations, also for 2D flows, as it may have a positive impact on the stability of the solver. This is especially true when wiggles (oscillations) appear in the shape of the free surface, as can be typically observed when a take-up force or velocity is imposed at the exit of the calculation domain. Note that upwinding is automatically applied when you choose the line kinematic condition, described in *Discontinuity of the Normal Direction* (p. 318) and *Guidelines for 3D Extrusion Problems* (p. 328) .

   Upwinding is *not* recommended for use with surface tension.

7. By default, no drag is applied on the free surface. If you want to include it, select the **Drag** option

   Drag

   In some fiber spinning cases, it is not unusual that drag force originating from the air surrounding the free boundary of the fiber becomes significant with respect to the take-up force. When this is the case, the option should be selected, and numerical values for the various parameters should be entered. Also it is important to explicitly activate the selected feature by clicking

   Enable drag

   Failing to do so will simply not activate the calculation of the drag, regardless of the numerical values for the various parameters.

   Note that this option is available for 2D axisymmetric calculations only.

8. When running a so-called inverse extrusion case, it is important to specify the shape of the extrudate. For this, select the option **Outlet (Inv. Prediction)**:

   Outlet (Inv. Prediction)

   See *Inverse Extrusion and Die Design* (p. 331) for further details on inverse prediction and die design.

9. Define the remeshing procedure for the sub-task (after you finish setting all boundary conditions and return to the sub-task menu one level above the **Flow boundary conditions** menu).

   Global remeshing

   See *Remeshing* (p. 343) for details. For inverse extrusion problems, see *Inverse Extrusion and Die Design* (p. 331) for information about additional inputs in the **Global remeshing** menu.

10. Modify the interpolation scheme(s) for the sub-task, if necessary.

    Interpolation

    See *Controlling the Interpolation* (p. 326) for details.

11. If necessary, impose a constraint on the free jet displacement (in the task menu).

    Constraint on free jet displacement

    See *Constraint on Global Displacement* (p. 335) for details.

12. For 3D extrusion problems, specify the use of the line kinematic condition for handling discontinuities in the normal direction (in the task menu).

    Numerical parameters

    See *Guidelines for 3D Extrusion Problems* (p. 328) for details. Note that the line kinematic condition can be used only if you are using the Optimesh or streamwise remeshing method for every local remeshing region in the task.

15.3.2. Defining the Direction of Motion

By default, ANSYS POLYFLOW evaluates the direction $\mathbf{D}_i$ at a given node i as the normal to the initial mesh. $\mathbf{D}_i$ does not change, except for evolution and time-dependent problems, as discussed below. You can, however, specify the direction yourself, either along the boundary of the moving surface or along the entire moving surface. The direction that you specify will apply to all affected nodes; you cannot specify a different direction for each node.

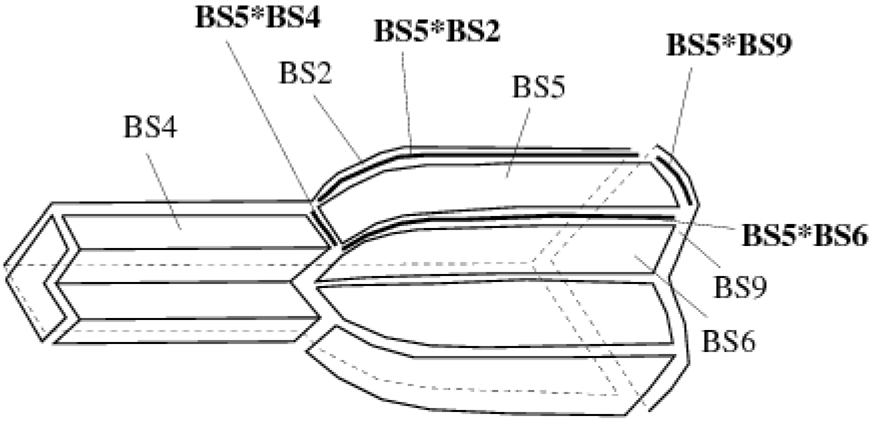
Specifying $\mathbf{D}$ along the boundary of the moving surface is useful if a plane (or line) of symmetry is located on one side of the moving surface. Specifying $\mathbf{D}$ along the entire moving surface is useful if large deformations are expected, in order to prevent directors from crossing. If you specify $\mathbf{D}$ along the entire moving surface, use the direction that is mostly orthogonal to the surface.

Note that the directors have no effect on the displacement of the moving surface when the line kinematic condition is used.

15.3.2.1. Boundary of the Free Surface or Moving Interface

The boundary of the free surface or moving interface is identified as a sequence of intersections with adjacent boundary sets and (for moving interfaces) adjacent subdomains. In *Figure 15.5* (p. 325), for example, the boundary of the free surface represented by boundary set 5 is composed of the intersections between BS5 and BS2, BS5 and BS4, BS5 and BS6, and BS5 and BS9, where the intersection is represented by a *.

Figure 15.5 Boundary of a Free Surface



15.3.2.2. Directors and Symmetry Planes

Along the intersection between a plane (or line) of symmetry and a free surface or moving interface, it is recommended that the director $\mathbf{D}$ be contained in the plane of symmetry. For steady-state problems where the normal to the initial mesh is contained in the plane of symmetry, this is not required (although it is correct), since the director will never be computed on the deformed mesh.

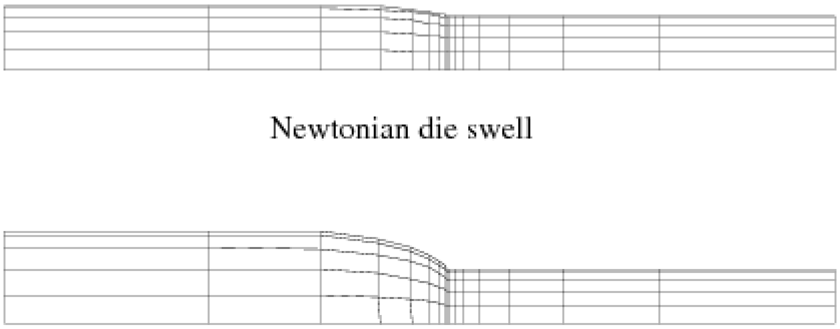
It is absolutely required for evolution and time-dependent problems, and also if you restart a problem from a deformed initial mesh. If you do not define the director in the plane of symmetry, nodes can move out of the plane of symmetry when remeshing occurs.

15.3.2.3. Frequency of the Director Calculation

Although directors are not recalculated during iterations, they are recalculated before each evolution or time step by default. You can override this by specifying that directors should be evaluated only once at the beginning of the task (See *General Procedure* (p. 320)).

Calculating directors at each step is recommended for problems involving large deformations, such as viscoelastic flow problems where large swelling effects are expected. The normal to the free surface near the die lip is almost vertical for a Newtonian fluid, and almost horizontal for a viscoelastic fluid (*Figure 15.6* (p. 326)). The director calculation at each step is highly recommended for high-Weissenberg-number flows.

Figure 15.6 Newtonian and Viscoelastic Die Swells



Newtonian die swell

viscoelastic die swell

15.3.3. Controlling the Interpolation

Free-surface and moving-interface problems involve velocity and pressure variables and coordinate unknowns. The shape functions for the velocity $\mathbf{v}$ and the position $\mathbf{x}$ should not be selected independently.

You should follow the guidelines listed below:

- In the absence of surface tension, only first-order derivatives of the free-surface or moving-interface position appear in the basic set of equations through $\mathbf{n}$ in *Equation 15–14* (p. 316) or *Equation 15–15* (p. 316). In this case, it has been shown that the shape function for $\mathbf{x}$ should be of order equal to that of the velocity shape function minus one. The default settings in 2D are quadratic velocities and linear coordinates (**Quadratic velocities, linear pressure** and **Linear coordinates** in the **Interpolation** menu for the sub-task in ANSYS POLYDATA), in order to satisfy this requirement.You should not need to change from these default settings.

  The default setting in 3D is the mini-element, which also satisfies this requirement, but the 8-node velocity brick does not. You should therefore use either the mini-element or quadratic velocities for 3D free-surface or moving-interface problems, always with linear coordinates (**Linear coordinates** and **Mini-element for velocities, constant pressure** or **Quadratic velocities, linear pressure**).
- In the presence of surface tension, second-order derivatives of the free-surface or moving-interface position appear through R in *Equation 15–17* (p. 317). In this case, the velocity and position (coordinates) shape functions should be of the same order. This is *not* the default in ANSYS POLYDATA, so you will need to change the interpolation type for coordinates to be quadratic (**Quadratic coordinates**) if you use the (default) quadratic velocity shape function (**Quadratic velocities, linear pressure**).

15.3.4. Convergence Strategies

By default, ANSYS POLYFLOW uses the Newton-Raphson technique to deal with nonlinearities introduced by free surfaces and moving interfaces. Position variables for all free surfaces and moving interfaces are embedded in the Newton-Raphson scheme, together with the expressions relating the internal node locations to the position variables. As with all nonlinear problems, you may encounter some convergence difficulties. Try the following suggestions to improve convergence:

- Apply evolution to the moving boundaries. To use this method (which will be available only for evolution problems), select **Enable evolution on moving boundaries** in the **Numerical parameters** menu or in the **Global remeshing** menu.

  This technique progressively introduces the kinematic condition into the system of equations being solved. When the evolution parameter S is equal to 0, the kinematic condition is ignored. When S reaches a maximum value of 1, the full kinematic condition is taken into account.

  This method has been shown to work well for most flow problems (generalized Newtonian, viscoelastic, variations of shape, etc.), except those involving a recirculation on the free surface, as in coating applications. For such cases, you should switch to a time-dependent calculation.

- Replace all free-surface conditions with a free-slip condition (i.e., $v_n = 0$, $f_s = 0$). The normal force will then not correspond to the desired value, but all problems linked to position nonlinearity will be eliminated and the solution will be a good starting point for the desired free-surface-problem solution. Use this solution as an initial guess for your free-surface problem. Alternatively, you can use evolution on moving boundaries, as described in the previous section.

- Variations of shape can occur due to important nonuniformities of the velocity at the die outlet. If convergence difficulties arise in solving a steady-state problem without surface tension, it is recommended that you replace the zero-velocity boundary condition along walls adjacent to free surfaces by a slip condition. Free slip occurs when the slip coefficient F_{slip} is equal to zero, and the velocity is equal to zero when F_{slip} is very large. If the problem converges for a low value of F_{slip}, use an evolution scheme to increment F_{slip}. In order to estimate the value of F_{slip}, you can perform a 2D channel flow simulation with the appropriate material properties, and then define F_{slip} so that the value of the wall velocity is 10 to 30% of the average fluid velocity. In many practical situations, nonzero slip along the die wall predicts more realistic extrudates.

  See *Slip Condition* (p. 176) for information about defining slip conditions, and *Evolution* (p. 517) for information about evolution.

- Replace all moving-interface conditions with $v_n = 0$, $f_s = 0$ conditions on *both sides* of the interface. If ANSYS POLYFLOW converges, then start the moving-interface calculation from the converged fixed-interface solution. If ANSYS POLYFLOW does not converge, the problem is not caused by the moving boundaries; you should look elsewhere to determine the cause of the convergence trouble. If the fixed-interface problem requires an evolution scheme to converge, start the moving-interface calculation from the converged fixed-interface solution obtained upon completion of the evolution.

- For small capillary numbers (i.e., when capillary forces are large compared to viscous forces), use quadratic coordinates and quadratic velocities for 2D, or linear coordinates and the mini-element for 3D. The capillary number is defined by *Equation 15–22* (p. 330).

- An infinite value of σ corresponds to a straight free surface (or moving interface). If this is acceptable from the point of view of mesh deformation, start the calculation with a straight mesh and a large value of σ, and decrease it using an evolution scheme.

- For integral viscoelastic problems, decouple the free-surface updates from the velocity and pressure calculations. See *Calculations Involving Moving Boundaries* (p. 264) for details.

15.3.5. Guidelines for 3D Extrusion Problems

15.3.5.1. Convergence

If you encounter convergence difficulties in a 3D extrusion problem, applying evolution to the moving boundaries (as described in *Convergence Strategies* (p. 327)) is the method of choice.

15.3.5.2. Discontinuity of the Normal Direction

As discussed in *Discontinuity of the Normal Direction* (p. 318), the line kinematic condition is recommended over the default surface kinematic condition. To enable the line kinematic condition, select **Line kinematic condition** in the **Numerical parameters** menu or in the **Global remeshing** menu.

Line kinematic condition

As mentioned in *Surface Kinematic Condition* (p. 319), if you use the surface kinematic condition for handling discontinuities in the normal direction (rather than the recommended line kinematic condition), it is important for the mesh to have a sufficient number of boundary sets to allow for proper piecewise definition of the free surface around edges.

15.3.6. Guidelines for Coextrusion Problems

Coextrusion is a process that is widely used to take advantage of the different mechanical, chemical, electrical, thermal-insulation, and possibly optical properties of different materials. Combining different grades of the same material into an extruded product also allows the use of cheaper materials or recycled resin where it is possible.

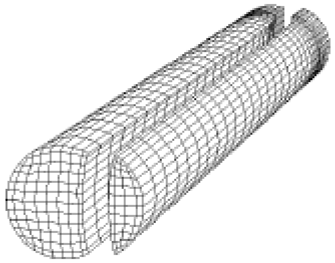
Coextrusion of two or more materials can be simulated with ANSYS POLYFLOW in 2D and in 3D. For 2D cases, there are very few limitations on the complexity of the deformation of the moving interface. With advanced 2D remeshing techniques such as the Thompson transformation and Optimesh 2D methods, there is no constraint on the displacement of the nodes, so they can move anywhere in the plane of the flow.

For full 3D coextrusion problems, however, difficulties may arise in some circumstances. The limitation of the Optimesh and streamwise remeshing techniques that the nodes can be relocated only in the plane orthogonal to the extrusion direction may be too constraining. In some cases, the nodes should be allowed to move in all three dimensions. The Thompson 3D remeshing technique allows for full 3D motion of the nodes, but it also requires longer CPU time (due to the 3D relocation of the points) and may be less robust in some cases. Other techniques are currently under consideration.

Depending upon the complexity of the geometry and/or the shape of the die itself, the intersection line between the die wall and the moving boundary may move along a curved line (as shown in *Figure 15.7* (p. 329)). Along this line, the normal component of the velocity is 0, whereas the tangential component is nonzero.

You will need to specify a nonzero slip coefficient along the wall in order to allow the intersection line to slide along the die border. Then the intersection line can move in a way that keeps it inside the geometrical entity defined by the wall. The slip coefficient should be specified such that the slip velocity remains small with respect to the mean flow velocity.

Figure 15.7 3D Coextrusion Example

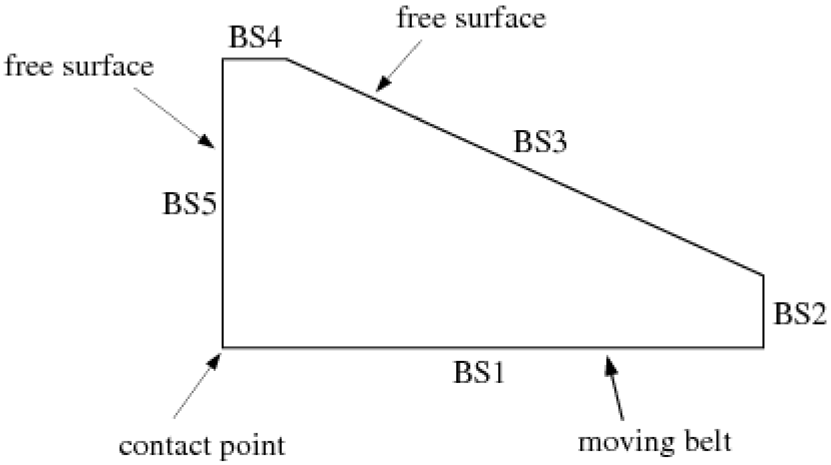


15.3.7. Static and Dynamic Contact Points or Lines

A contact problem occurs at points (in 2D) or lines (in 3D) where the velocity is fixed and the position of the free surface or moving interface is not prescribed. In this case, you have a moving contact point (or line). Consider, for example, the curtain coater shown in *Figure 15.8* (p. 329). The flow enters at a constant speed at the top of the domain (BS4) and falls onto a belt that moves horizontally (BS1). The problem involves two free surfaces: one on the right (BS3) and one on the left (BS5). The fluid is carried by the belt, and the point located at the intersection between the belt and the left free surface is a contact point. The position of free surface BS5 is not known, and the velocity is imposed based on the motion of the belt.

Since the position of the moving surface at the contact point is not prescribed, a contact angle is prescribed between the free surface and the adjacent boundary.

Figure 15.8 Example of a Contact Point



If your problem includes a point or line where the velocity is fixed and the position of the free surface or moving interface is not prescribed, ANSYS POLYDATA will automatically request input of the contact angle, and implement the moving-contact-point strategy.

ANSYS POLYDATA will warn you if the model is incoherent (i.e., if a velocity has not been imposed at a location where the free-surface or moving-interface position has been imposed, or in a moving-contact-point problem if the director has not been imposed).

15.3.7.1. Contact Points and Lines

Contact points are, in some sense, a paradox, since the Lagrangian kinematic requirement $\partial \mathbf{x} / \partial t = \mathbf{v}$ readily describes the displacement on such points. If $\mathbf{v} = 0$, these points or lines do not move. In the numerical simulation, however, you may want to include additional effects such as capillary forces, which bring the fluid back to the wall when sharp free-surface or moving-interface gradients exist.

15.3.7.2. The Moving-Contact-Point Model

The moving-contact-point model allows you to take these additional effects into account. This model can be used whenever the capillary number Ca is not too large. The capillary number Ca measures the importance of viscous forces compared to capillary forces:

$$Ca = \frac{\mu V}{\sigma} \tag{15–22}$$

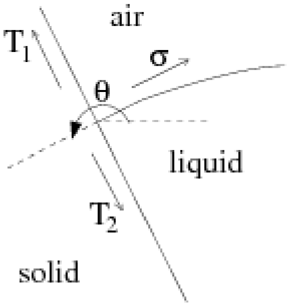
where μ is the viscosity and V is a characteristic velocity of the flow.

For contact problems that do *not* include surface tension (or when the capillary number is large), the time-dependent approach of *Blow Molding and Thermoforming* (p. 379) must be used instead of the moving-contact-point model.

When surface tension is present, you will need to specify a static contact angle wherever the moving-boundary position is not prescribed. This angle is called static because it will be respected only in the limit of small viscous forces (i.e., when capillary forces dominate all other forces). This angle corresponds to the wetting angle (θ in *Equation 15–20* (p. 318)) between the fluid and the wall, as shown in *Figure 15.9* (p. 330).

The wetting angle is completely determined by the interfacial force balance at a triple point in situations where the fluid does not move and therefore does not generate viscous or inertial forces. If the fluid moves, viscous (and possibly inertial) forces will be combined with the surface tension force (which is tangent to the free surface, as described in *Surface Tension* (p. 316)), and the point will move up to a position where the projection of the resulting force on the direction tangent to the wall vanishes.

Figure 15.9 Static Contact Angle



This assumes that the moving contact point has no mass in itself, and that momentum equilibrium is satisfied in the direction tangent to the wall. In a direction normal to the wall, no displacement is allowed.

15.3.7.3. Inputs for Dynamic Contact Points

To use the moving-contact-point model in ANSYS POLYFLOW, follow the guidelines below when setting up your model in ANSYS POLYDATA:

- Be sure that your model actually involves a moving contact point (i.e., a velocity boundary condition is prescribed on the boundary adjacent to the free surface, surface tension is included, and a wetting angle is prescribed). As discussed in *Surface Tension* (p. 316), positive angles are measured counterclockwise with respect to a horizontal reference axis (see *Figure 15.3* (p. 318)).
- Prescribe the director of the contact point as the tangent direction to the wall. This means that moving-contact-point problems can be modeled only for straight lines, since only a constant director can be specified.
- In practice, moving-contact-point problems can be steady-state as well as time-dependent. For the steady-state case, the kinematic condition degenerates into a dynamic condition that states the equilibrium of the tangential forces. However, in most moving-contact-point problems, it is likely that a time-dependent approach will be more computationally stable than a steady-state approach. A time-dependent approach guarantees that the displacements will be small, provided that the time steps are small. Using a transient scheme is therefore highly recommended.

  If the surface tension is very small, the *only* way to handle a moving contact point is by using a time-dependent scheme with contact detection (penetration check). See *Blow Molding and Thermoforming* (p. 379) for details.

ANSYS POLYDATA will replace the kinematic condition (*Equation 15–2* (p. 312) or *Equation 15–3* (p. 312)) by the momentum balance in the direction tangent to the wall. Therefore, the static angle will be taken into account in the momentum balance, although the Dirichlet boundary condition prescribed on the velocity field supersedes this momentum equation.

If the free surface is normal to the wall, perfect wetting occurs. In this case, the direction tangent to the wall is also normal to the free surface, so the director never becomes tangent to the free surface. Conversely, if the free surface becomes tangent to the wall, no wetting occurs and the direction tangent to the wall is also tangent to the free surface. This results in an ill-posed problem, as discussed in *Free Surfaces* (p. 311). In this case, the only option is to use a contact detection algorithm such as that described in *Blow Molding and Thermoforming* (p. 379).

15.3.8. Inverse Extrusion and Die Design

In many cases, you will want to predict the die shape required in order to obtain a desired extrudate shape, rather than predicting the extrudate shape for a particular die. This is referred to as an inverse extrusion problem.

The procedure for setting up an inverse extrusion problem is similar to the procedure for a direct extrusion problem, but the remeshing will be applied to the die in addition to the extrudate, because the die shape is unknown *a priori* and will be computed based on the given extrudate shape. The extrudate must be remeshed as well, because only the exit shape is known. To define an inverse extrusion problem in ANSYS POLYDATA, follow the steps below (in conjunction with the general steps listed in *General Procedure* (p. 320)):

1. In the menu for the free-surface or moving-interface boundary condition, identify the outlet of the die, which is the beginning of the moving surface. This imposes a boundary condition on the free surface or moving interface, as described in step 3 in *General Procedure* (p. 320).

Boundary conditions on the moving surface

2. In the menu for free surface boundary condition, identify the exit of the calculation domain, whose shape will be considered as fixed and will be interpreted as the required extrudate shape:

   Outlet (Inv. Prediction)

   Usually, the outlet coincides with the exit of the calculation domain. The outlet is defined by specifying the boundary or subdomain adjacent to the free surface. It is important to note that this information is not related to the assignment of a boundary condition on the moving surface. Instead, it will be handled as a constraint on the development of the extrudate, which will be used for calculating the die shape that allows matching this constraint at the exit of the calculation domain.

   Alternatively, the outlet for inverse prediction can also be defined via the **Global remeshing** menu as indicated below under item 3(b).

3. Define the inverse prediction in the **Global remeshing** menu.

   Inverse prediction management

   a. Turn on the inverse prediction.

      Enable the inverse prediction

      Important

      You must define an outlet for each free surface and moving interface. Otherwise, ANSYS POLYDATA will issue a warning when you try to save the data file.

   b. Define the outlet of the free surface or moving interface.

      Free surface along boundary/subdomain x

      or

      Interface along boundary/subdomain x

      You will define the outlet in the same way as the inlet, by specifying the adjacent boundary or subdomain. The outlet will be the intersection of the free surface or moving interface with the specified boundary or subdomain.

      Note that you can also specify the outlet using the **Outlet (Inv. prediction)** item in the menu for the free-surface or moving-interface boundary condition.

4. Create a new local remeshing area (or select the first one), and select the subdomain(s) that represent the part of the die for which ANSYS POLYFLOW has to compute the shape (referred to as the adaptive section of the die), based on the specified extrudate shape.

   1-st local remeshing

This will allow the mesh to adapt between the constant entry section of the die and the die lip section, as ANSYS POLYFLOW calculates the die lip profile that produces an extrudate of the prescribed shape.

5. Select **Adaptive section for prediction** or **Adaptive section for prediction (no slice)** in the **Remeshing technique** menu.

   Adaptive section for prediction

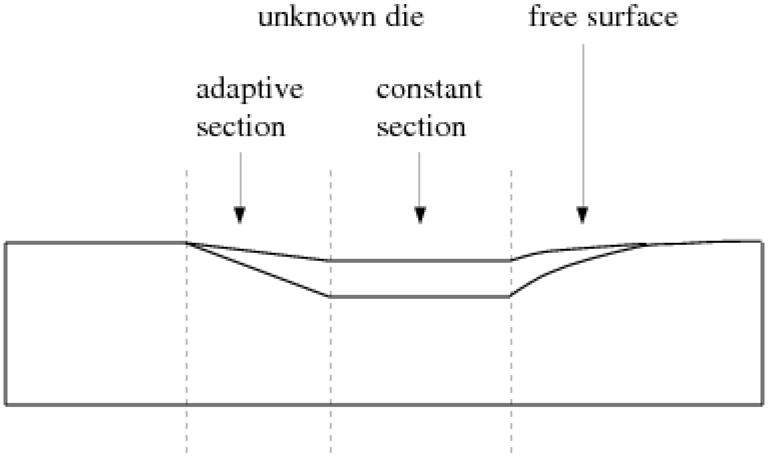
   Adaptive section for prediction can be chosen when the mesh of this section can be “sliced". This can be done if the mesh in this section is semistructured, that is, along the direction of extrusion, it is made of a series of successive “slices", all slices being topologically identical (i.e., same number of nodes, segments and faces). In such a case, the adaptive section will be deformed by applying homothetic transformations of various amplitudes on the slices.

   Adaptive section for prediction (no slice)

   Adaptive section for prediction (no slice) should be chosen when the mesh in this section is unstructured (for example, a mesh made of tetrahedra). In such a case, this section will be deformed using an elastic technique.

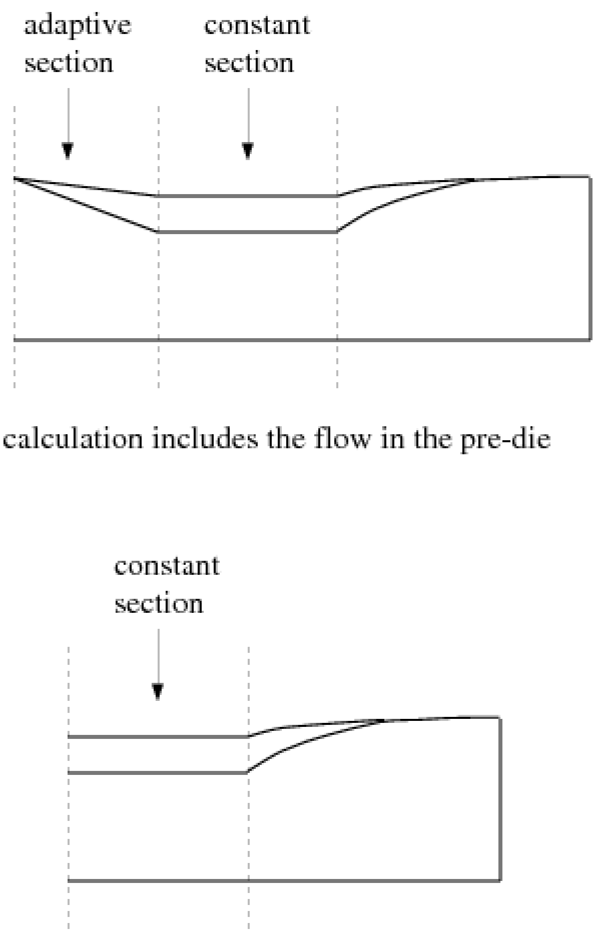
   When prompted, indicate the beginning (inlet) and end (outlet) of the adaptive section by specifying the intersection with the appropriate adjacent boundary or subdomain. See *Figure 15.10* (p. 333) and *Figure 15.11* (p. 334)

Figure 15.10 Typical Inverse Extrusion Problem



ANSYS POLYFLOW will adapt the mesh linearly between the fixed inlet section and the die exit (which is remeshed using Optimesh or streamwise remeshing).

Figure 15.11 Modeling Approaches for the Problem in *Figure 15.10* (p. 333)



(a) calculation includes the flow in the pre-die

(b) calculation involves mainly the die lips

15.3.8.1. Maintaining a Constant Shape for a Portion of the Die

To maintain the die lip shape over a given length in the direction of extrusion, define two local remeshing areas: one for the adaptive section of the die, and one for the constant section. To maintain the section at a constant shape, select **Constant section for prediction** or **Constant section for prediction (no slice)** instead of **Adaptive section for prediction** or **Adaptive section for prediction (no slice)**.

Constant section for prediction

As with Adaptive sections, **Constant section for prediction** can be chosen when the mesh of this section can be “sliced". In such a case, the constant section will be deformed by applying homothetic transformations of the same amplitude on all of the slices.

Constant section for prediction (no slice)

Constant section for prediction (no slice) should be chosen when the mesh in this section is unstructured (for example, a mesh made of tetrahedra). In such a case, this section will be deformed using an elastic technique.

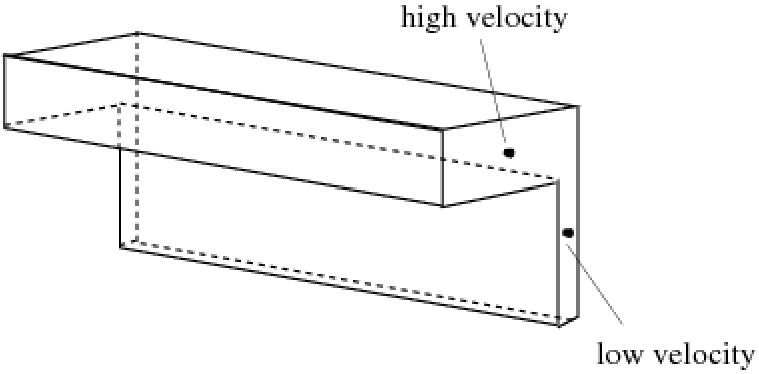
15.3.9. Constraint on Global Displacement

For steady-state extrusion simulations that do not involve any explicit constraints on the extrudate (e.g., plane of symmetry or slip conditions), ANSYS POLYFLOW will apply a global displacement constraint if the problem satisfies certain conditions, which are described later in this section.

To avoid excessive global displacement, the model allows the addition of a force and a torque on the free jet (or a portion of it). The computed force and torque can be interpreted as those produced by a guide used to prevent deviations in the extrudate shape. In a well-balanced die, the deviation of the extrudate is small and the force and torque are also small. An unbalanced die, however, will have very large deviations. Consider the unbalanced die illustrated in *Figure 15.12* (p. 335).

The extrusion of such a profile leads, in general, to a large deviation in the extrudate due to the differences in velocity. The deviation of the jet is produced by a nonzero average transverse velocity (ATV) in the cross-section (e.g., at the die exit).

Figure 15.12 Flow Through an Unbalanced Die



To avoid large displacements (translation and rotation) of the free jet, ANSYS POLYFLOW applies a force and a torque on all or part of the extrudate, if the problem satisfies certain conditions described later in this section.

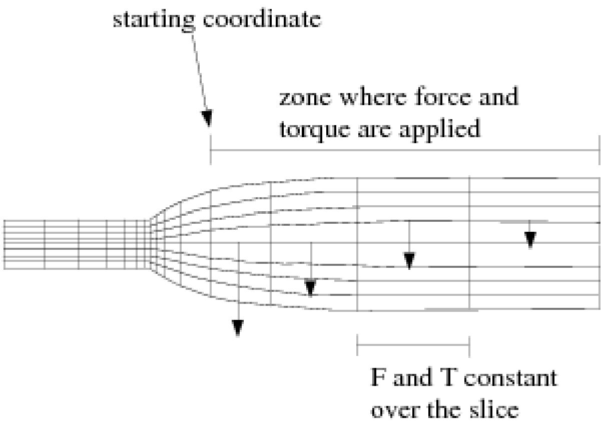
The force F acts like a body force in the equation of motion, while the torque T acts like a distributed torque by use of an asymmetric stress tensor in the equation of motion. The values of the force and torque components are computed by ANSYS POLYFLOW precisely so as to obtain a zero averaged transverse velocity (or rotation). When the large displacements are avoided, convergence of the solution becomes much easier.

15.3.9.1. Finite-Element Formulation

The mesh is sliced according to the definition of the remeshing rule. On each node slice (i.e., group of all nodes belonging to the same slice), the force and torque are assumed to be constant, as shown in *Figure 15.13* (p. 336).

Furthermore, the component of the force in the flow direction and the component of the torque in the transverse direction have imposed values of zero. Since the force and torque are constant in each slice, their effect on the shape of the extrudate section is small.

Figure 15.13 Sliced Domain with Force and Torque


starting coordinate
zone where force and torque are applied
F and T constant over the slice

15.3.9.2. Limitations

The following limitations apply to the constraint on global displacement:

- It is available only for non-axisymmetric extrusion problems. (Axisymmetry is a global displacement constraint in itself.)
- The extrusion direction must be aligned with the x, y, or z direction. When Optimesh 3D or streamwise remeshing is used, ANSYS POLYDATA will check that the inlet and outlet of the system of planes are real planes and are perpendicular to the x, y, or z direction.
- Optimesh, the streamwise method, the method of spines (2D), or the Euclidean method (3D) must be used as the remeshing method, so that slicing is available; the Thompson transformation cannot be used.
- Although improved elastic remeshing does not require a sliceable mesh, the constraint on the free jet requires such a mesh. Hence, you will be able to apply the constraint on the free jet coupling when using this remeshing technique, if the mesh is at least sliceable. If the mesh is not sliceable on the extrudate, ANSYS POLYDATA will not warn you, but ANSYS POLYFLOW will stop on an error message.
- The plane passing by the starting point and perpendicular to the extrusion direction must not cross any slice (i.e., the force or torque must either apply on an entire slice or not apply at all on the slice). If this condition is violated, the solver will not converge. *Figure 15.14* (p. 337) illustrates correct and incorrect positions for the starting coordinate.
- For multiple-jet simulations, the extrusion direction must be the same for all jets. The force and torque on each jet will, however, be computed separately. *Figure 15.15* (p. 338) illustrates multiple-jet problems for which the constraint can and cannot be applied.

Figure 15.14 Constraint on the Starting Coordinate

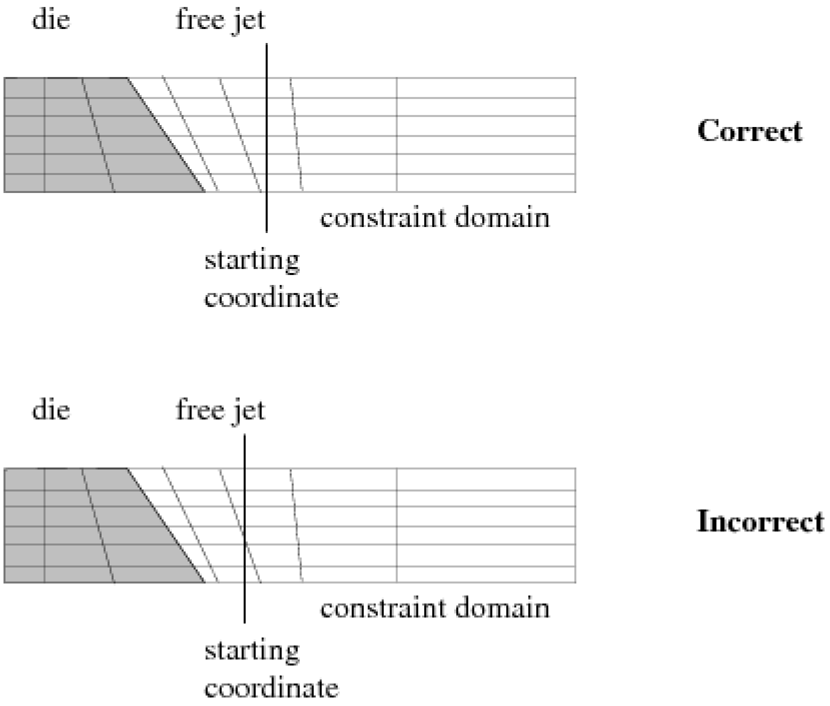
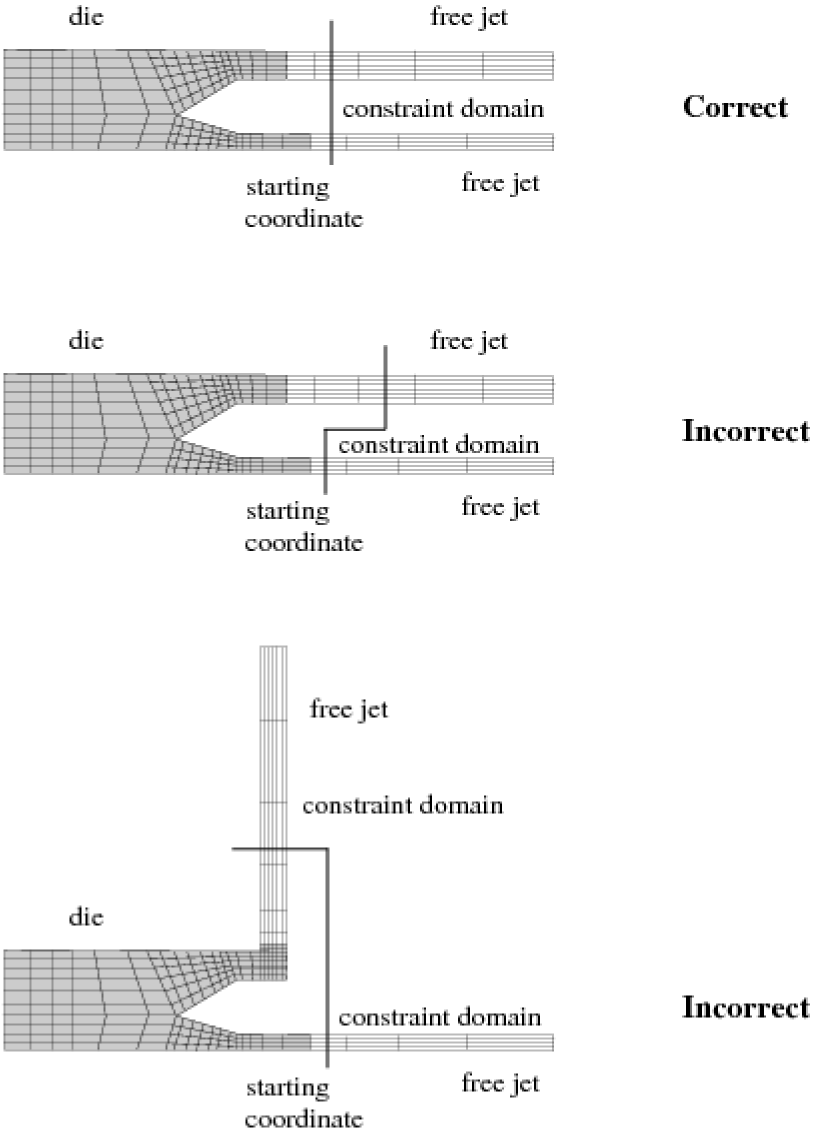


Figure 15.15 Constraint on Multiple-Jet Problems



15.3.9.3. Compatibility with Specific Geometries and Boundary Conditions

The constraint on global displacement is not compatible with velocity boundary conditions on the extrudate (e.g., plane of symmetry: $v_n = 0$; belt conveyor: $v_n = 0, v_s = V$). These velocity conditions are sufficient to avoid global displacements, so no additional constraints are required. For this reason, some force or torque components can be set to zero. *Table 15.1: Force and Torque Components for Geometries and Boundary Conditions* (p. 339) identifies the force and torque components for different cases as computed (C), equal to zero (0), or not computed (N) because they are incompatible with the boundary conditions. In this table, the flow direction is y for non-axisymmetric 2D geometries, and z for axisymmetric and 3D geometries.

The table tells you which component of the force or torque will be non-zero when the global displacement constraint is used. If you specify an inconsistent constraint, ANSYS POLYDATA will warn you. ANSYS POLYDATA cannot tell you when you might benefit from a global displacement constraint.

Table 15.1 Force and Torque Components for Geometries and Boundary Conditions

Geometry	Boundary Conditions	F_x	F_y	F_z	T_x	T_y	T_z
2D axisym (*Figure 15.16* a)	line of axisymmetry	N	–	0	–	–	–
2D planar (*Figure 15.16* b)	no plane of symmetry	C	0	–	–	–	–
2D planar (*Figure 15.16* c)	plane of symmetry, $v_x = 0$	N	0	–	–	–	–
2D planar (*Figure 15.16* d)	$v_x = 0, v_y = V$	N	–	–	–	–	–
3D (*Figure 15.17* a)	no plane of symmetry	C	C	0	0	0	C
3D (*Figure 15.17* b)	one plane of symmetry, $v_y = 0$	C	0	0	0	0	0
3D (*Figure 15.17* c)	two planes of symmetry, $v_x = 0$ or $v_y = 0$	0	0	0	0	0	0

This information must come either from your knowledge of the process being modeled or from your observation of large deviations in a previous simulation.

- For 2D geometries, there is no torque and F_x or F_y is zero. Axisymmetric geometries (*Figure 15.16* (p. 340) a) are incompatible with the constraint on global displacement.
- For a 2D geometry without velocity boundary conditions on the free jet (*Figure 15.16* (p. 340) b), there is only one component of force that is computed. 2D geometries with a symmetry plane (*Figure 15.16* (p. 340) c) or with normal (to the extrusion axis) velocity imposed (*Figure 15.16* (p. 340) d) are incompatible with the constraint on global displacement.
- For 3D geometries without velocity boundary conditions on the free jet (*Figure 15.17* (p. 340) a), the transverse components of the force are computed in order to avoid the displacement in the transverse direction, and the force component in the flow direction is equal to zero; the torque components in the flow direction is computed in order to avoid extrudate rotation, and the other torque components are equal to zero.
- For 3D geometries with one plane of symmetry (*Figure 15.17* (p. 340) b), extrudate rotation is not possible, so the torque is not taken into account. Since displacement in the direction perpendicular to the symmetry plane is not allowed, the force in this direction is imposed to be zero.

Figure 15.16 2D Geometries

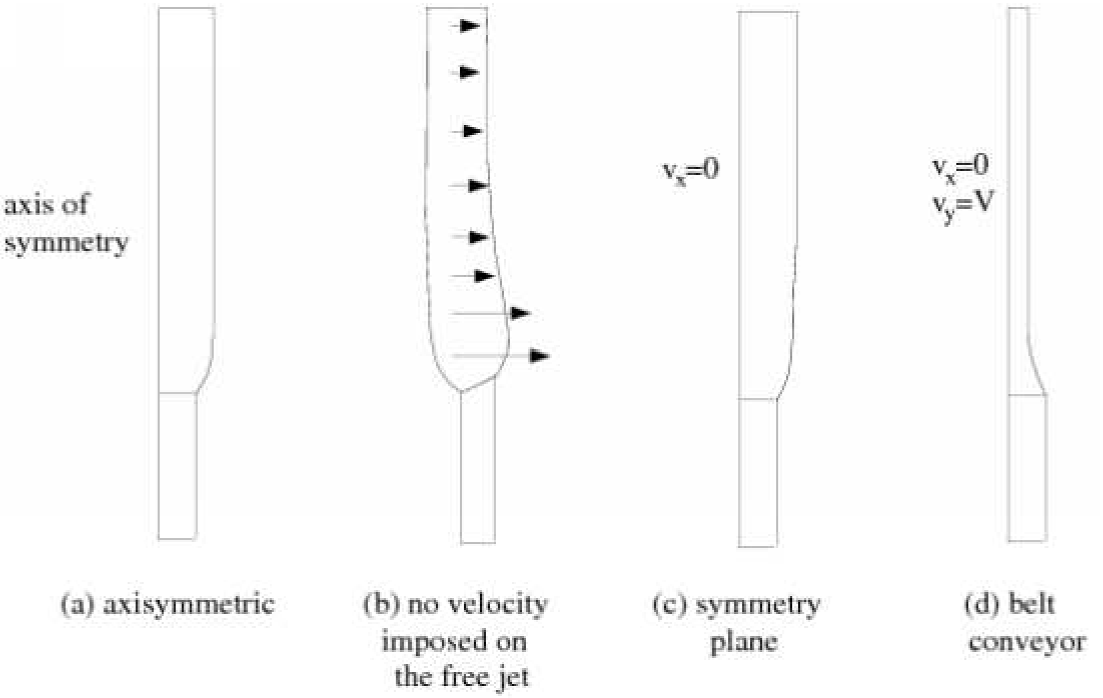
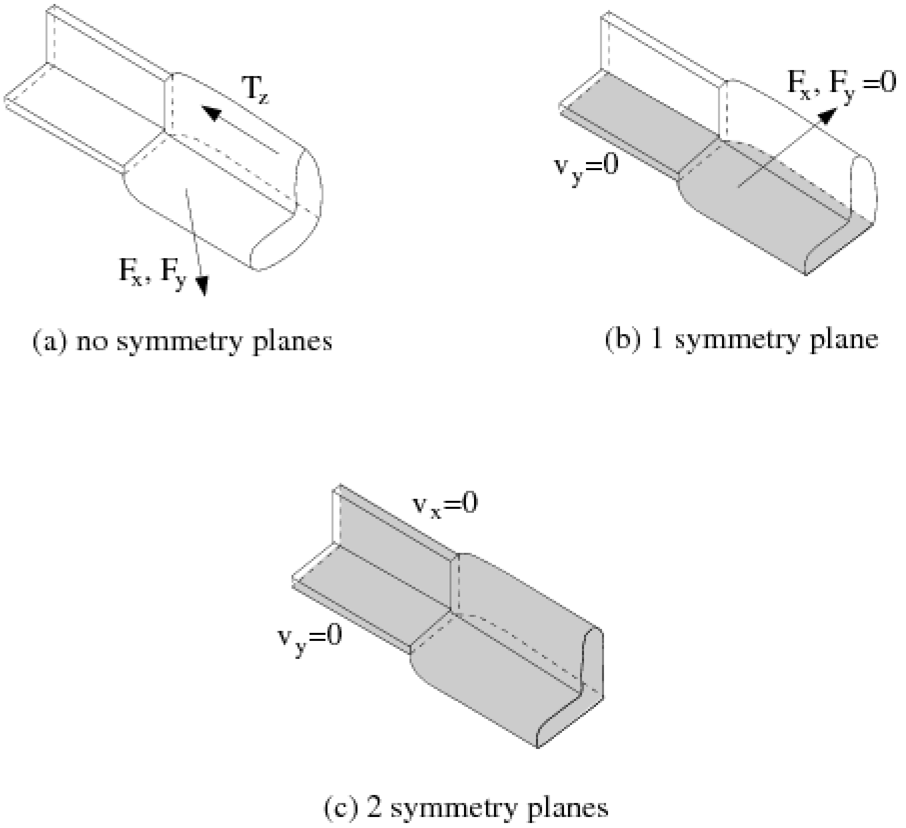


Figure 15.17 3D Geometries



- For a 3D geometry with two planes of symmetry or other velocity boundary conditions (*Figure 15.17* (p. 340) c), the transverse and rotation displacements of the free jet are not possible, so the force and torque are not taken into account.

ANSYS POLYDATA detects these situations automatically and sets the appropriate force and torque components to zero. If you try to use the constraint on global displacement when it is incompatible with the boundary condition, ANSYS POLYDATA will inform you of the incompatibility in an error message.

15.3.9.4. User Inputs for the Constraint on Global Displacement

For steady-state extrusion simulations that do not involve any explicit constraints on the extrudate (e.g., plane of symmetry or full slip conditions), the constraint on global displacement is recommended, and ANSYS POLYDATA will enable it automatically.

If you disable it, poor convergence or even divergence may occur, typically with a large-amplitude displacement of the free surface or moving interface for intermediate converged steps when evolution on moving boundaries is used.

If you want to modify the constraint on the global displacement of the free jet, the procedure is as follows:

1. Set up the extrusion problem as usual, following the steps in *General Procedure* (p. 320).
2. Enter the menu for the constraint on global displacement (in the task menu).

   Constraint on free jet displacement

   The option is already turned on. To turn it off, select **Disable constraint on displacement**.
3. Modify the application of the constraint on all or part of the remeshing domain.

   Options

   - By default, ANSYS POLYDATA will compute the extrusion direction and apply the appropriate force and torque on *all* remeshing domains. This is indicated by the default selection of the **Constraint on all the remeshing domain** option.

     Constraint on all the remeshing domain
   - To apply the constraint to only a portion of the remeshing domain, select **Constraint on a part of the remeshing domain** and follow the steps below:

     Constraint on a part of the remeshing domain

     a. Here, the algorithm stabilizes the extrudate along the actual extrusion direction. In other words, if the extrudate moves towards positive values of the coordinate along the extrusion direction, ANSYS POLYDATA informs you that

        Forces applied on [starting coord; +inf]

        This indicates that the constraint should be applied to the portion of the remeshing domain where the coordinate in the extrusion direction is greater than the starting coordinate:

        If the extrudate moves towards negative values of the coordinate along the extrusion direction, ANSYS POLYDATA informs you that

        Forces applied on [-inf; starting coord]

This indicates that the constraint should be applied to the portion of the remeshing domain where the coordinate in the extrusion direction is smaller than the starting coordinate.

b. Specify the starting coordinate for the portion of the remeshing domain where the constraint should be applied.

   Modify starting coordinate

   The coordinate will be in the direction of extrusion.

Chapter 16: Remeshing

Remeshing techniques are necessary when free surfaces or moving interfaces are present. The purpose of a remeshing technique is to relocate internal nodes according to the displacement of boundary nodes. Remeshing techniques control mesh deformations in order to avoid unacceptable element shapes.

Information about ANSYS POLYFLOW's remeshing techniques and instructions for using them are provided in this chapter.

16.1. Introduction to Remeshing

16.1.1. About Remeshing

Kinematic conditions dictate boundary displacements in the normal direction only. In an Eulerian formulation, displacements of the interior nodes are not prescribed, and you can freely select internal node positions so as to minimize mesh deformation. It is also perfectly valid to move nodes tangentially along boundaries, since such displacements do not interfere with the kinematic conditions. Most remeshing techniques will also guarantee a tangential smoothness of the mesh along boundaries.

In ANSYS POLYFLOW, remeshing techniques are based only on the positions and displacements of the boundary nodes, and not on kinematic considerations, unless a Lagrangian or streamwise method is used for remeshing. In many cases, it is impossible to maintain good mesh regularity without relocating nodes tangentially to the domain boundary. Tangential remeshing preserves the original node distribution along a line for 2D moving domains, and along a surface for 3D moving domains.

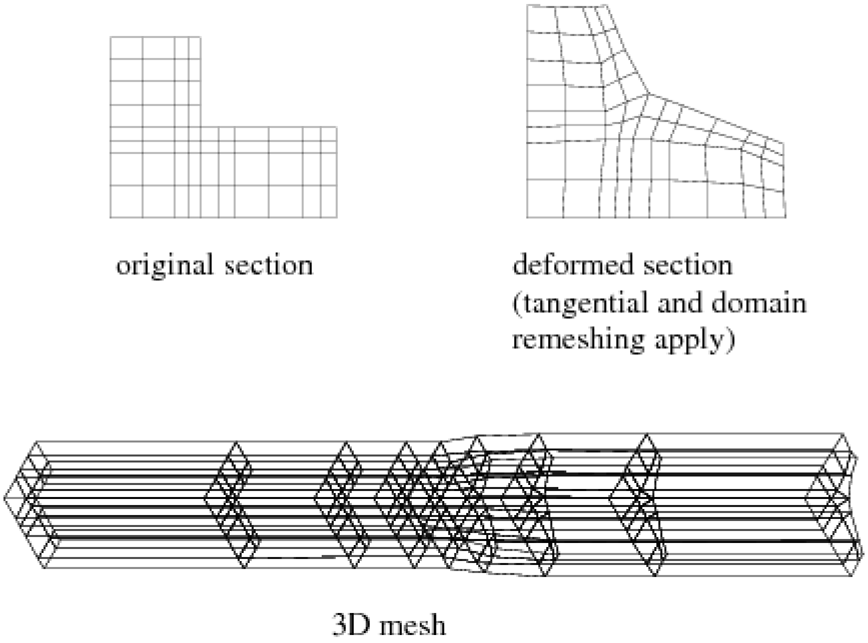
Figure 16.1 (p. 344) shows tangential and domain remeshing. Note the following:

- The kinematic condition controls the displacements along directors $\mathbf{D}_i$, except with the line kinematic condition, where any displacement can occur in a plane orthogonal to the direction of extrusion.
- Each kinematic condition introduces a tangential displacement along adjacent free surfaces or boundary sets.

- Remeshing in the interior of the domain is based on the nodal displacements on the boundary.

In *Figure 16.1* (p. 344), the normal displacement of nodes on the free surfaces is controlled by the kinematic equation; a tangential remeshing is applied on the planes of symmetry, while the position of the internal nodes is controlled by a remeshing technique.

Figure 16.1 Tangential and Domain Remeshing



original section

deformed section (tangential and domain remeshing apply)

3D mesh

In ANSYS POLYFLOW, these operations occur simultaneously. Constraints resulting from kinematic conditions generate tangential remeshing constraints. All boundary displacements (whether tangential or normal) are substituted into the internal domain remeshing rule, so that a full system of constraints between all nodal displacements is assembled. A full Newton-Raphson scheme on all nodal positions (interior and on the domain boundary) is derived.

16.1.2. Remeshing Techniques in ANSYS POLYFLOW

ANSYS POLYFLOW provides several different remeshing methods, in order to satisfy the remeshing requirements for a variety of flow problems:

- method of spines
- Euclidean method
- method of planes
- Thompson transformation
- Optimesh
- thin shell method
- Lagrangian method
- thin shell method with Lagrangian master
- Lagrangian method on the borders only
- streamwise method
- elastic method

- improved elastic method

Remeshing rules that are topologically general (i.e., Thompson transformation and elastic methods) will require more CPU time and memory than those that apply only to specific topologies, because they involve a wider coupling between the position variables and the other solution fields.

You will have to define the remeshing technique separately for each sub-task of a multiple-sub-task problem. Remeshing rules do not apply across sub-tasks. On the other hand, you can define different remeshing techniques for different subdomains associated with a given sub-task. This is referred to as local remeshing, and is discussed in *Local Remeshing* (p. 365).

See *Adaptive Meshing* (p. 366) for information about adaptive meshing, which can be used with or without remeshing.

16.1.3. Choosing a Remeshing Technique

Each of the remeshing techniques available in ANSYS POLYFLOW is designed to work well for a certain class of problem. Recommendations for the use of the remeshing techniques are listed below:

- The method of spines is recommended for 2D jet-like extrusion and coating problems, where 1D remeshing is sufficient
- The method of planes can be used for some 3D extrusion problems for which remeshing is not needed in the direction of extrusion. Note that this method cannot handle large deformations; Optimesh is recommended instead.
- The Thompson transformation is recommended for 2D and 3D problems with complex deformations in all directions.
- Optimesh 2D, like the Thompson transformation, is recommended for 2D problems with large deformations in all directions.
- Optimesh 3D is recommended for extrusion (or inverse extrusion) problems for which large deformations of the extrudate are expected.
- Thin shell remeshing is recommended for 2D and 3D blow molding (large deformations of thin parisons).
- Lagrangian remeshing is available only for blow molding problems. It is the simplest algorithm available for such cases, and is available for 2D, 3D volumetric, and 3D shell models. Its primary application is for 3D shells.
- Thin shell remeshing with a Lagrangian master is available only for 2D blow molding problems, and is appropriate for a thin fluid parison.
- Lagrangian remeshing on the borders is available only for blow molding problems, and is recommended for thick 2D fluid regions.
- Streamwise remeshing is recommended for 3D extrusion problems, as an alternative to Optimesh, especially when large isotropic deformations of the extrudate exist, as in 3D fiber spinning.
- Elastic and improved elastic remeshings are recommended for cases where the inherent topological complexity makes it difficult (or even impossible) to apply topologically regular methods.

Note that of the methods described above, the following require that the mesh of the free jet is a sliceable mesh: method of planes, Optimesh 3D, streamwise, and improved elastic remeshing. See *Generating a Sliceable Free Jet Mesh* (p. 164) for more information about sliceable meshes, as well as how to convert meshes that are not sliceable.

16.2. Method of Spines

The method of spines is the simplest remeshing rule in ANSYS POLYFLOW. Mesh nodes are organized along lines of remeshing, called spines. This method is available only for 2D flows, since its applicability in 3D would be very limited from a geometric point of view. Spines are collections of nodes logically arranged in a one-dimensional manner. They are obtained by slicing up the 2D domain. The slicing process is performed in a direction that is topologically normal to the free surface or moving interface, and therefore the definition of initial and final spines is required.

Spines are connected lines (which are not necessarily straight) and have boundary nodes at each extremity. Consider a spine characterized by extremities $\mathbf{x}_1$ and $\mathbf{x}_2$. With the spine remeshing rule, the displacement of an internal mesh node $\mathbf{x}_i$ on this spine is defined as follows:

$$\delta\mathbf{x}_i = w_{1i}\delta\mathbf{x}_1 + w_{2i}\delta\mathbf{x}_2 \tag{16–1}$$

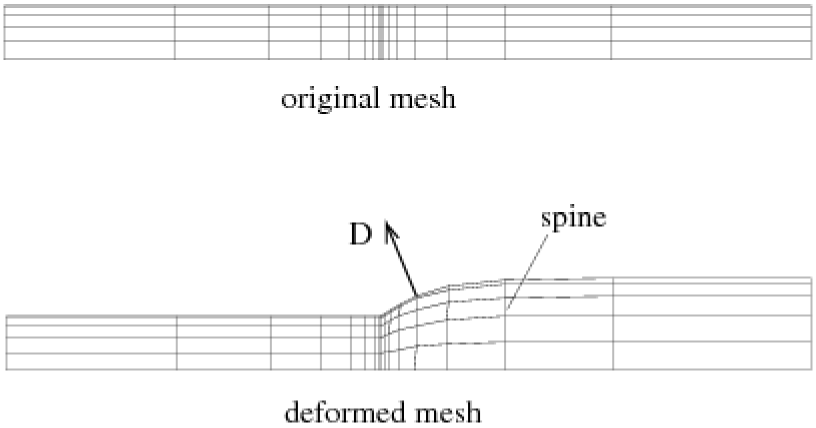
where $\delta\mathbf{x}$ is the variation of position and

$$w_{1i} = \frac{|\mathbf{x}_i - \mathbf{x}_2|}{|\mathbf{x}_2 - \mathbf{x}_1|} \tag{16–2}$$

$$w_{2i} = \frac{|\mathbf{x}_i - \mathbf{x}_1|}{|\mathbf{x}_2 - \mathbf{x}_1|} \tag{16–3}$$

The spine remeshing technique is illustrated in *Figure 16.2* (p. 346), which shows the original and deformed mesh configurations.

Figure 16.2 The Method of Spines



The spine technique does not remesh tangentially along free surfaces and moving interfaces. In view of its low requirements of computer resources, it is well adapted to a large class of 2D problems with jet-like free surfaces and moving interfaces.

See *User Inputs for Remeshing* (p. 359) for information about using the method of spines.

16.3. Euclidean Method

The Euclidean method extends the concepts of the 1D spine technique to two dimensions. The displacement of an internal node is a linear function of the displacements of all boundary nodes of the current slice. The weighting depends on the Euclidean distance between the internal node being relocated and the corresponding boundary node. The weighting factor equals 1 when the node to be relocated coincides with a boundary node, and it equals 0 when the node is very far from a given boundary node. This can be mathematically expressed as follows:

$$\mathbf{x}_i^{\text{new}} = \mathbf{x}_i^0 + \sum_{j=1}^{\text{NBD}} w_{ij} \left(\mathbf{x}_j^{\text{new}} - \mathbf{x}_j^0 \right) \tag{16–4}$$

where w_{ij} are weighting factors defined as

$$w_{ij} = \frac{\sum_{k=1}^{\text{NBD}} \left\| \mathbf{x}_i^0 - \mathbf{x}_k^0 \right\|_2}{\text{NBD} \cdot \left\| \mathbf{x}_i^0 - \mathbf{x}_j^0 \right\|_2} \tag{16–5}$$

NBD is the number of boundary nodes in a given section and i and j are indices of the internal and boundary nodes, respectively.

Before the internal nodes are relocated according to *Equation 16–4* (p. 347), the mesh nodes on the free surface or moving interface are relocated according to the kinematic condition. Nodal displacements propagate into tangential displacement on adjacent faces. This results in a tangential remeshing technique applicable for small deformations. *Figure 16.1* (p. 344) illustrates how sections will be remeshed.

An advantage of the Euclidean remeshing technique is that the number of position unknowns is small compared to the velocity and pressure unknowns. Moreover, this technique has a complete topological generality.

Its drawback is that it is robust only for moderately small displacements of the free surface or moving interface, because of its linearity. It has been found that only elliptic techniques (based on partial differential equations of the elliptic type), such as the Thompson transformation (described in *Thompson Transformation* (p. 348)) or Optimesh (described in *Optimesh* (p. 352)) are robust enough to deal with large boundary displacements. See *User Inputs for Remeshing* (p. 359) for information about using the Euclidean remeshing method.

16.4. Method of Planes

ANSYS POLYFLOW also offers a remeshing rule called the method of planes. This method is similar to the spine technique and can be considered a 3D generalization of the method of spines. The method of planes has been implemented for 3D extrusion applications (direct and inverse), but it remains limited to small displacements of the free surface. The method is based on the algorithm described below.

First, ANSYS POLYFLOW cuts the 3D domain to be remeshed into a series of 2D slices orthogonal to the direction of extrusion (or topologically normal to the free surface or moving interface). The slicing of the domain lowers by one the geometrical dimension of the remeshing rule: for a 2D spine technique, a one-dimensional rule is used; for a 3D Euclidean technique, a two-dimensional rule is used. Each slice (or plane) is then independently remeshed. This means that mesh deformations cannot occur in the flow direction. Each node belongs to only one slice, which is internally considered a 2D mesh. Information required for this slicing includes the definition of initial and final planes. In each slice, ANSYS POLYFLOW will apply the 2D Euclidean distance method described in *Euclidean Method* (p. 347). *Figure 16.1* (p. 344) illustrates how sections will be remeshed.

The method of planes is useful for jet-like free-surface problems, such as extrusions. It is not expensive in terms of CPU time, due to the few additional unknowns it introduces. It inherits the topological generality of the Euclidean distance method, but suffers from the same limitation; it is robust only for relatively small deformations of the initial mesh. For extrusion simulations, the 3D Optimesh technique and streamwise method have been proven to be able to cope with much larger displacements of the boundary. See *User Inputs for Remeshing* (p. 359) for information about using the method of planes.

16.5. Thompson Transformation

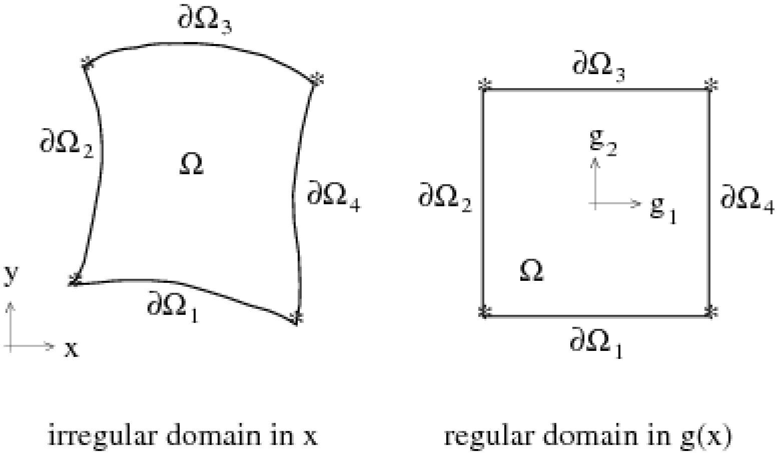
Unlike standard algebraic remeshing techniques, the Thompson transformation remeshing technique [*28*] (p. 716) is based on the resolution of a partial differential equation of the elliptic type, and remains robust for larger mesh deformations. One of the advantages of using the Thompson transformation is that the mesh can contain any type of element (quadrilaterals, triangles, bricks, wedges, tetrahedra). The other models are more limited due to their use of slicing; if tetrahedra or triangles are included in the mesh, slicing becomes impossible because the mesh is not logically organized.

The topology of the parent mesh is almost arbitrary. The drawback is the large CPU time and memory requirements. For most 2D problems, however, this cost increase remains quite acceptable.

16.5.1. Theory

The transformation maps a parent domain into a deformed domain. The transformation begins with the construction of a global coordinate system, defined on a simply shaped domain. These global coordinates represent a mapping of the irregularly-shaped computational domain into a regular parent domain (such as a square in 2D or a cube in 3D). For example, the irregular domain shown in *Figure 16.3* (p. 348) is mapped onto a square by the global-coordinates transformation function $\mathbf{g}\,(x\,)$. The coordinates $\mathbf{g}$ have the same number of components as the mesh coordinates $\mathbf{x}$.

Figure 16.3 Thompson Transformation



In the 2D case shown in *Figure 16.3* (p. 348), the global coordinates $\mathbf{g}$ are constructed on the basis of the following equation:

$$\nabla^2 \mathbf{g}(\mathbf{x}) \quad = \quad 0 \quad \text{on} \quad \Omega \tag{16–6}$$

with the following boundary conditions:

$$\begin{aligned}
\mathbf{n}\cdot\nabla g_1 &= 0 \quad \text{and} \quad & g_2 &= -1 \quad \text{on} \quad \partial\Omega_1 \\
g_1 &= -1 \quad \text{and} \quad & \mathbf{n}\cdot\nabla g_2 &= 0 \quad \text{on} \quad \partial\Omega_2 \\
\mathbf{n}\cdot\nabla g_1 &= 0 \quad \text{and} \quad & g_2 &= 1 \quad \text{on} \quad \partial\Omega_3 \\
g_1 &= 1 \quad \text{and} \quad & \mathbf{n}\cdot\nabla g_2 &= 0 \quad \text{on} \quad \partial\Omega_4
\end{aligned} \tag{16–7}$$

The Thompson transformation for this example is simply the inverse mapping $\mathbf{g}^{-1}(\mathbf{x})$. Once $\mathbf{g}(\mathbf{x})$ has been constructed, the Thompson transformation is the solution $\mathbf{x}$ that satisfies *Equation 16–6* (p. 349) with appropriate boundary conditions. The calculation of the $\mathbf{g}$ field, as well as the Thompson transformation itself, is solved by standard finite-element techniques.

Equation 16–6 (p. 349) is linear in $\mathbf{g}$, but not in $\mathbf{x}$. The Thompson transformation is inherently nonlinear in $\mathbf{x}$, even though the Laplacian is a linear operator. The construction of the $\mathbf{g}$ field is performed only once at the beginning of the simulation, before the main solver starts. The Thompson transformation is then solved in a coupled fashion, together with the equations governing the flow.

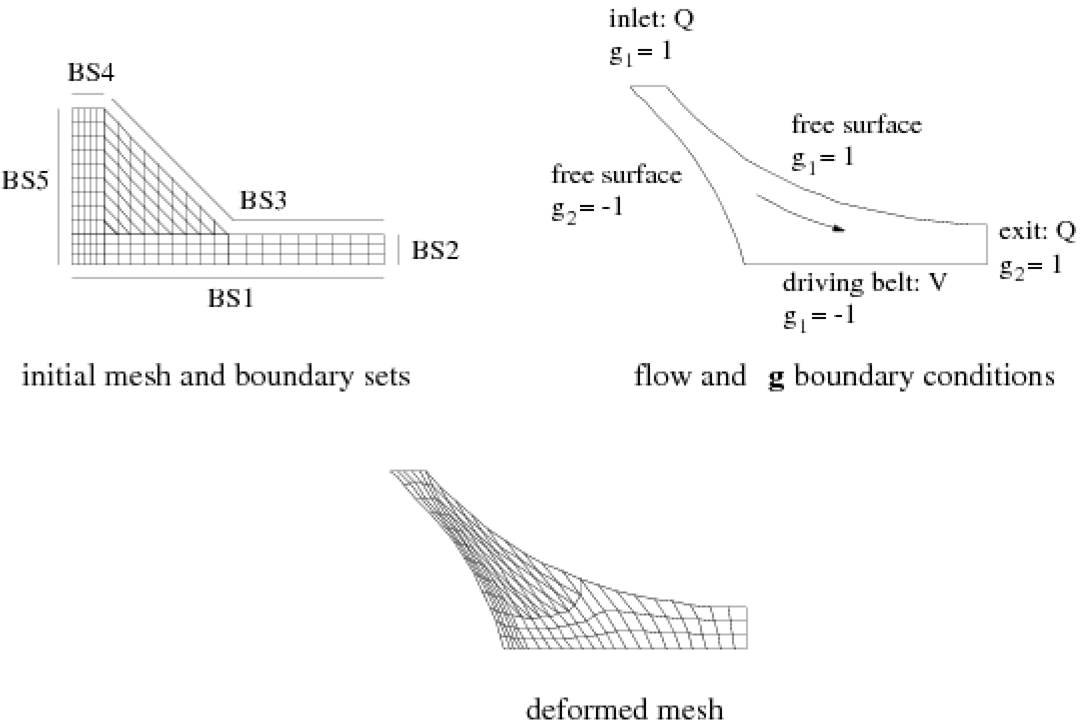
The choice of boundary conditions given to $\mathbf{g}$ and $\mathbf{x}$ in the Thompson transformation governs the remeshing itself. The Thompson transformation is applied to the remeshing subdomain. Each separate sub-boundary of the remeshing subdomain is referred to as a segment. On each segment, one component of the vector $\mathbf{g}$ (essential boundary condition) must be specified, while a zero-normal-derivative condition (natural boundary condition) is applied to the other component(s).

16.5.2. Example

Consider the example shown in *Figure 16.4* (p. 350). This flow problem models a coating process, which involves two free surfaces. A Thompson transformation is used for the whole domain, since large mesh deformations are expected. The boundary conditions are described as follows: the fluid enters the flow domain at the top, is driven at the bottom by a belt that moves from left to right, and exits the computational domain on the right-hand side; the two remaining sides are free surfaces.

Five boundary sets are defined in *Figure 16.4* (p. 350), and boundary conditions are selected so that the mesh domain maps logically onto a square (2D) or cube (3D) region. The choice of g_1 or g_2 is not always unique. Values on opposite sides of the square or cube must be different, but the value itself is not relevant. For this reason, values of 1 and -1 are generally used. On BS1, a value of -1 for g_1 is imposed. On BS3 and BS4, $g_1 = 1$; on BS2, $g_2 = 1$; and on BS5, $g_2 = -1$. In general, a zero-normal-derivative condition is imposed on boundary sets without an imposed component, in order to close the problem. A zero-normal-derivative condition is therefore imposed for g_1 on BS2 and BS5, and for g_2 on BS1, BS3, and BS4. ANSYS POLYFLOW will calculate the solution of the Laplace equation for both g_1 and g_2. This solution is shown in *Figure 16.5* (p. 350). Then, upon deformation of the mesh, ANSYS POLYFLOW will relocate nodes in such a way that g_1 and g_2 are unchanged for a given node. Contour lines of the g_1 and g_2 fields are shown on the deformed and original (parent) meshes in *Figure 16.5* (p. 350). The corresponding mesh is shown in *Figure 16.4* (p. 350).

Figure 16.4 Coating Example for Thompson Transformation

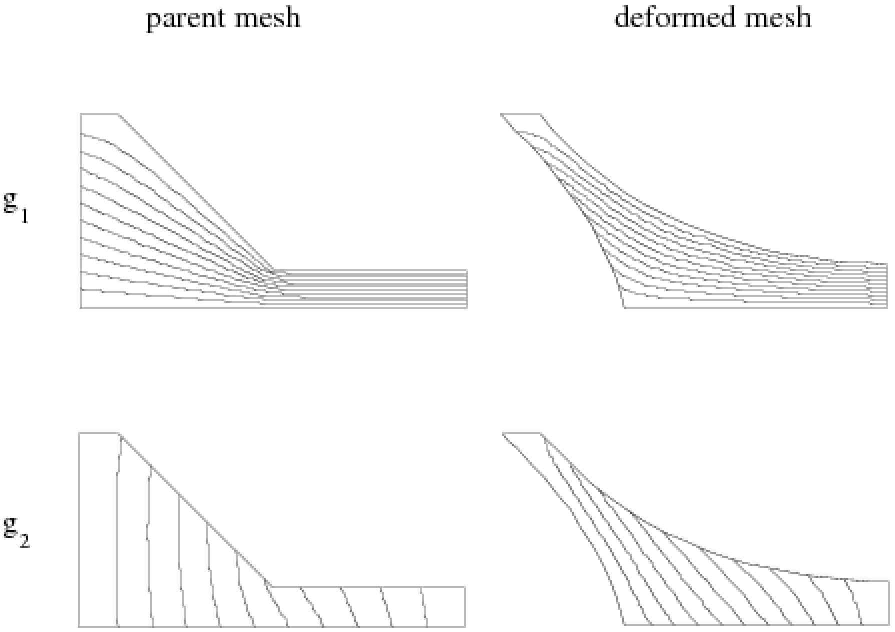


initial mesh and boundary sets

flow and **g** boundary conditions

deformed mesh

Figure 16.5 The g Field for the Thompson Transformation



16.5.3. Implementation

This section describes in detail how the Thompson transformation is implemented. You should familiarize yourself with the basics described above before reading further. When defining boundary conditions for the **g** field, ANSYS POLYDATA verifies that the following conditions are satisfied:

- All components of the **g** field have been imposed on at least two different faces.

- The same component of $\mathbf{g}$ is not imposed with different values on adjacent faces.

The first requirement ensures that the remeshing domain is mapped onto a rectangular geometry (a rectangle in 2D or a parallelepiped in 3D). The second requirement ensures that no possible singularities in the $\mathbf{g}$ field are introduced by the imposed boundary conditions. Only one component of $\mathbf{g}$ can be imposed on a segment. For the other components, natural boundary conditions are used, as in the example of *Figure 16.4* (p. 350). Similarly, on a given face $\partial\Omega_f$, a component g_i is imposed, while natural conditions are used for the remaining components:

$$g_i = \text{constant on } \partial\Omega_f \qquad \textbf{(16–8)}$$

$$\mathbf{n}\cdot\nabla g_k = 0 \text{ for all } k \neq i \text{ on } \partial\Omega_f \qquad \textbf{(16–9)}$$

For these, ANSYS POLYDATA allows two possible conditions for $\mathbf{x}$ on $\partial\Omega_f$. The first condition requires that points on this face be fixed (i.e., the corresponding displacement vanishes):

$$\Delta\mathbf{x} = 0 \text{ on } \partial\Omega_f \qquad \textbf{(16–10)}$$

For this condition, the face is referred to as a fixed face: all coordinates $\mathbf{x}$ remain fixed on $\partial\Omega_f$. This condition is automatically applied to a face on which neither tangential nor normal remeshing applies. The second possible condition requires that

$$\Delta x_i = 0 \text{ on } \partial\Omega_f \qquad \textbf{(16–11)}$$

and

$$\mathbf{n}\cdot\nabla g_k = 0 \text{ for all } k \neq i \text{ on } \partial\Omega_f \qquad \textbf{(16–12)}$$

with the further restriction that

$$|\, n_i \,| \ = 1 \text{ and } n_k = 0 \text{ for all } k \neq i \text{ on } \partial\Omega_f \qquad \textbf{(16–13)}$$

For this second condition, the face is referred to as a symmetry face. It is important to note that there is an implicit connection between the imposed component g_i and the component x_i that is fixed in a symmetry face. Imposing g_1, g_2, or g_3 requires fixing x, y, or z, respectively. The remaining non-imposed components (k) define the axis or plane of symmetry. For example, if g_1 is imposed, x becomes fixed and the face has symmetry about the y- z plane (or the y axis in 2D). Furthermore, the requirement of *Equation 16–12* (p. 351) stipulates that this face is parallel to the axis or plane of symmetry.

In addition to the fixed and symmetry faces, another type of condition is required to maintain smooth mesh point distributions. Consider, for example, a face that is a free surface. It is desired that the original distribution of points along the free surface itself be maintained. This face cannot be a fixed face, since it must be free to move in order to satisfy the kinematic condition (*Equation 15–2* (p. 312) or *Equa-*

tion 15–3 (p. 312)). Moreover, a normal boundary condition (a generalization of the symmetry-type condition) can sometimes lead to an undesirable redistribution of points on the free surface.

The solution chosen for such faces is to create a separate tangential remeshing of the Thompson transformation type on the boundary itself. ANSYS POLYDATA automatically chooses this solution and generates the corresponding problem with the associated boundary conditions. Still, for free surfaces, the displacements of nodal points along the directors remains controlled by the kinematic condition.

Faces that are not free surfaces can also be allowed a tangential remeshing. In such cases, the original tangential distribution of points on the face is maintained, while a zero displacement is imposed in the direction normal to the surface. Such a condition is useful for faces that change size (but not curvature) due to the presence of a free surface or interface. To simplify matters for you, ANSYS POLYDATA automatically makes some decisions about which faces are fixed faces, symmetry faces, and tangentially remeshed faces.

For moving boundaries, ANSYS POLYDATA uses the following rules:

- Free surfaces are given tangential remeshing.
- Faces that form borders between subdomains are fixed faces. The only exception to this rule is moving interfaces as the Thompson transformation currently is not applied to moving interfaces. The position of points on these faces is governed solely by the kinematic condition.

For the remaining faces, ANSYS POLYDATA uses the following rules:

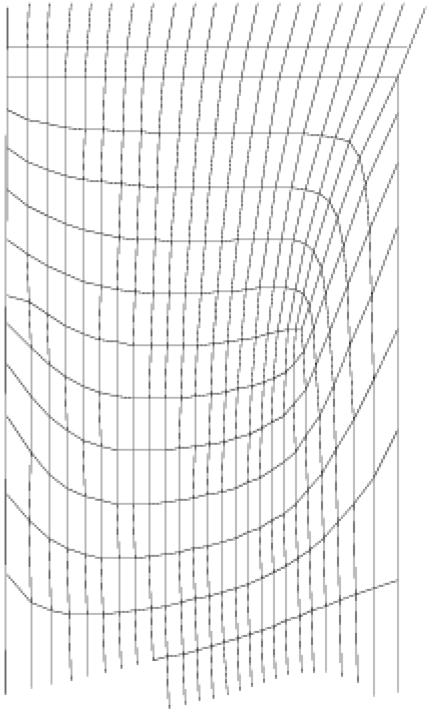
- Faces directly adjacent to free surfaces are given tangential remeshing.
- Symmetry conditions can be imposed on a face. If imposed, this overrides the previous rule.
- Faces not defined by the other four rules are fixed faces.

In 3D problems, the intersection of two tangentially remeshed faces forms a line that is given tangential remeshing by a separate one-dimensional Thompson transformation. All boundary conditions associated with this one-dimensional remeshing are inherited from the two- and three-dimensional transformations. Definition of such one-dimensional transformations is handled automatically by ANSYS POLYDATA.

16.6. Optimesh

Optimesh is an acronym for a remeshing scheme based on a “minimum energy" rule. The Optimesh algorithm belongs to the family of elliptic remeshing rules and, as such, handles internal nodal displacements as variables. The CPU requirements for Optimesh are therefore similar to those of the Thompson transformation. However, Optimesh is often more robust than the Thompson transformation because the angular deformation of each individual element is under control of the remeshing rule.

Figure 16.6 Detail of a 2D Grid Remeshed with Optimesh



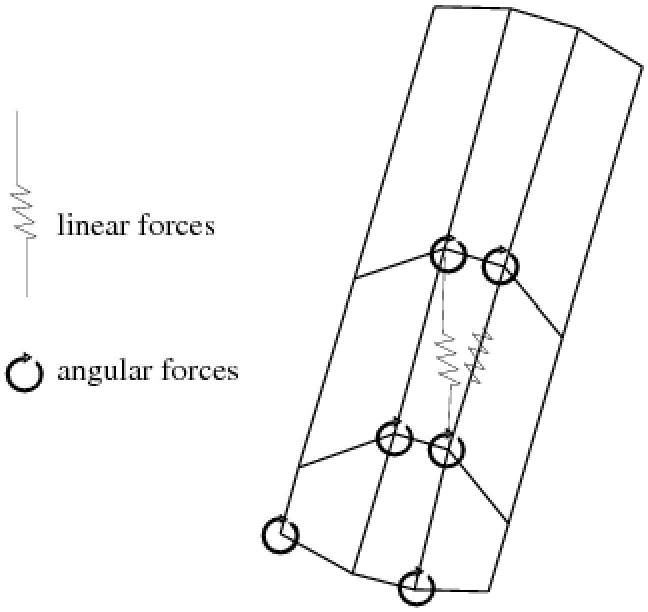
Optimesh can be used for any 2D problem, and for any 3D extrusion problem (or any 3D problem that does not require remeshing in one direction, typically the direction of extrusion).

For 3D extrusion problems, the domain is sliced in the direction of extrusion, generating a series of 2D meshes on which the Optimesh rule can be applied. *Figure 16.6* (p. 353) shows a portion of a 2D grid remeshed with Optimesh.

16.6.1. Theory

The Optimesh method locates nodes of an element in such a way as to minimize the energy of deformation of the mesh. This energy of deformation is a function of the angular distortion of the elements, and of the elongation of “springs" located along the segments and diagonals of the mesh (see *Figure 16.7* (p. 354)). One non-dimensional parameter α controls the relative weight of the angular and diagonal springs. When $\alpha = 0$, only angular springs exist, so the relative size of the elements is uncontrolled. A reasonable mesh regularity is usually obtained for $\alpha = 0.1$, which is the condition used by ANSYS POLYDATA (and cannot be modified).$\alpha = 0.1$ also leads to optimal convergence of the remeshing method.

Figure 16.7 How the Optimesh Remeshing Works



Angular springs are nonlinear, in order to produce an infinite deformation energy when elements become degenerate. This guarantees that the mesh is regular upon convergence of the remeshing method. This also makes the remeshing problem itself nonlinear.

16.6.2. Boundary Conditions

Optimesh requires prescribed nodal displacements along the boundary of the domain being remeshed. ANSYS POLYDATA does this automatically by relocating nodes along 1D boundaries using a proportionality rule, just as it does for the Thompson transformation.

In 2D, you will need to introduce boundary conditions for the $\mathbf{g}$ field, as discussed in *Thompson Transformation* (p. 348) and *User Inputs for Remeshing* (p. 359). ANSYS POLYDATA will then relocate nodes along 1D boundary segments in such a way that the curvilinear abscissa of the node is maintained. This constraint serves as a boundary condition for the Optimesh remeshing rule. In the normal direction, nodes are located in such as way that the kinematic condition is satisfied. In 3D, a different strategy is used. Along boundaries of the remeshing domain, the normal displacement is prescribed by the kinematic condition (for a free surface) or imposed to zero (for all other cases). In the tangential direction, nodes are relocated by the Optimesh technique to satisfy the equilibrium of forces.

16.7. Thin Shell Method

The thin shell remeshing technique combines some of the advantages of the Thompson and Optimesh methods (elliptic rules) with some of the advantages of the algebraic spine technique (low cost and control of the motion of internal mesh nodes). This remeshing algorithm is designed to handle the large mesh deformations and nodal displacements that are encountered in blow molding and thermoforming applications. The thin shell rule is available in both 2D and 3D.

The thin shell remeshing method exploits the topological regularity that is usually found in blow molding applications (single- or multilayer), but it is by no means limited to the simulation of blow molding.

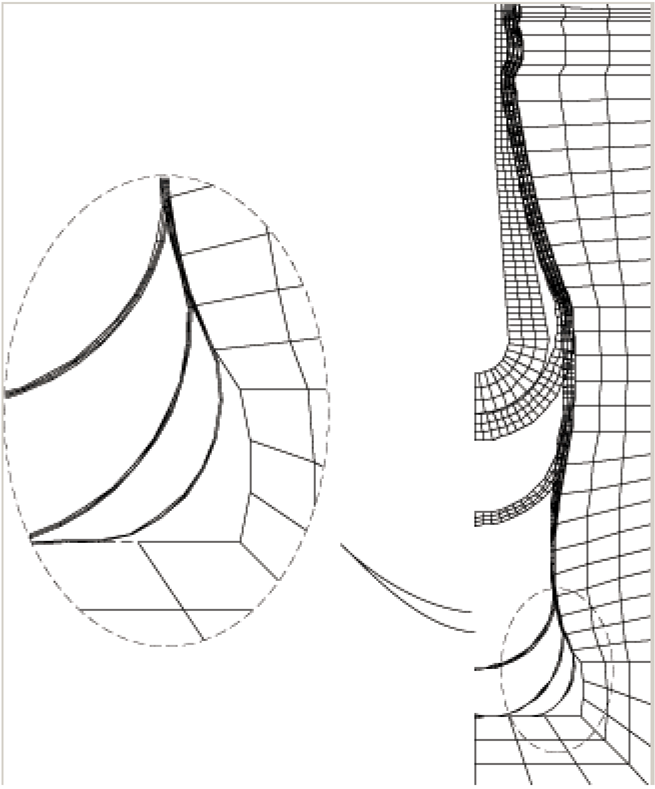
Note that the use of the thin shell remeshing technique does not introduce any particular thin shell assumptions into the governing equations. There is, in fact, no interaction between a particular remeshing rule and the equations to be solved, and even when the thin shell remeshing technique is used, the full flow equations (momentum, energy, incompressibility, and constitutive equations) are solved.

16.7.1. Theory

The thin shell remeshing process is characterized by large deformations in the direction tangent to the parison (in 3D, we refer to the 2D subspace tangent to the surface), combined with a relatively small thickness. In addition, the regularity of the mesh topology is such that it can be organized in a system of spines generated in the thickness of the parison. *Figure 16.8* (p. 355) shows a 2D axisymmetric blow molding problem on which the thin shell remeshing method is to be used.

Thin shell remeshing combines tangential remeshing along a single free surface (or interface) called the master free surface with a spine rule in the thickness of the parison. In other words, a tangential remeshing is applied to one of the free surfaces, in order to keep an acceptable element distribution within the mesh. The internal nodes are relocated along spines that are defined as geometrically normal to the master free surface.

Figure 16.8 2D Axisymmetric Blow Molding Problem



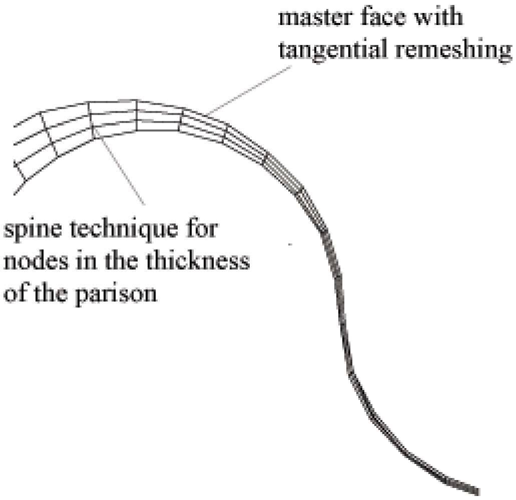
The procedure is as follows:

1. You select the master free surface among all free surfaces and moving interfaces. It is common to select the boundary set that will come into contact with the mold.
2. The system of spines is automatically built as the set of nodes connected through mesh segments to each node of the master free surface.

3. In multilayer applications, you must select a remeshing rule of the thin shell type in each sub-task (one for each layer). However, definition of the master free surface is unique throughout all remeshing rules of the thin shell type currently defined. See *User Inputs for Remeshing* (p. 359) for information about inputs for this remeshing method.

Nodes of the master free surface are tangentially remeshed using a 1D proportionality rule (for 2D problems) or a 2D Optimesh rule (for 3D problems). The internal nodes are proportionally relocated with the spine technique; for each spine, the internal nodes are relocated along a direction geometrically normal to the master surface. *Figure 16.9* (p. 356) illustrates the application of the thin shell remeshing rule.

Figure 16.9 The Effect of Tangential Remeshing in the Case of Large Deformations



In 2D, the thin shell remeshing technique preserves the node distribution originally introduced along the master free surface. Note that the standard Thompson remeshing technique is not recommended when the parison is thin, because element distortion in the normal direction might then become large.

In 3D, nodes will be relocated tangentially on the master free surface at the beginning of the calculation. Nodes located on the boundary of the master surface (i.e., along 1D curves) will follow material lines, in order to provide the Optimesh rule with appropriate boundary conditions. Due to the constraint that prevents boundary nodes from moving for very small time steps, displacements of the original mesh on the master surface are rather limited, and the convergence behavior is generally good.

16.8. Lagrangian Method

The Lagrangian remeshing method is applicable for all transient 2D, 3D volumetric, and 3D shell models, although it is recommended for use only for 3D shell models. In this method, the nodes simply follow the displacements of the material points

$$\frac{dx}{dt} = v \tag{16–14}$$

in all directions of the Cartesian space. For a 3D shell model (whether the simulation occurs on the boundary of a 3D mesh or on a shell mesh), one of the advantages is that the nodal displacements follow the material points, making the physical deformations more readily apparent.

16.9. Thin Shell Method with Lagrangian Master

In blow molding simulations, as well as for related applications, the finite-element mesh for the fluid domain may undergo large deformations. The primary type of deformation is an extension. In addition, a moderate amount of shear is also possible. Specific remeshing algorithms have been developed that allow the mesh elements to retain an acceptable shape. The objective is to keep the shape of the finite element as close as possible to a rectangle.

For 2D blow molding applications, several remeshing techniques are available in ANSYS POLYFLOW, in order to meet the various geometrical characteristics of the computational domain. If your model includes a thin fluid parison, like the example shown in *Figure 17.1* (p. 380), the most appropriate technique involves the use of the thin shell method (described in *Thin Shell Method* (p. 354)) combined with a Lagrangian representation (described in *Lagrangian Method* (p. 356)) along the “master" surface.

This method is available only in 2D. In this combined method, a Lagrangian displacement (*Equation 16–14* (p. 356)) is imposed for the nodes on the surface that undergoes the contact (the master surface), whereas nodes in the thickness get remeshed with a spine rule similar to the one used for the thin shell method. This method remains robust in the presence of shear.

Note the following requirements for use of the thin shell method with Lagrangian master:

- The computational domain for the fluid region must be surrounded by four boundary sides.
- Topologically, the mesh must be an $n \times m$ regular checkerboard, as illustrated in *Figure 16.9* (p. 356); in other words, the same amount of element must exist along two opposite boundary sides.
- For boundary conditions, two free surfaces must be defined, along two opposite sides.

16.10. Lagrangian Method on Borders

In classical blow molding applications where thin fluid regions are considered, the shear component is essentially absent from the flow kinetics. In other flow processes, such as squeezing flows where a thick fluid volume is present, shear is not always negligible.

Here, half of a sphere of fluid will be squeezed by the downward motion of the upper plate. A full Lagrangian representation (described in *Lagrangian Method* (p. 356)) for the equations is acceptable as long as the elements are not sheared too much. Other situations may exhibit a complex kinematics development, and a full Lagrangian representation may not be the most appropriate.

For flows involving contact occurring over time, a Lagrangian representation is used at least for the free surface that undergoes the contact; this improves the robustness of the contact algorithm. Consequently, it is appropriate to select a remeshing algorithm based on the combination of a Lagrangian representation on the border of the fluid domain and a minimum-pseudo-energy representation for the inner mesh nodes. This method is referred to as Lagrangian on the borders.

This method is applicable only to 2D cases, and is appropriate for rather thick parisons where remeshing with a spine technique in the thickness direction is not appropriate. This method uses *Equation 16–14* (p. 356) on all the boundaries, and uses Optimesh to remesh the interior nodes.

16.11. Streamwise Method

The streamwise method is applicable to extrusion and coextrusion models only. It assumes a regularity of the domain, which can be seen as a series of mesh lines emitted from the extrudate section. ANSYS POLYDATA will therefore ask you for inlet and outlet sections for the remeshing region.

Along these mesh lines, which can be on the boundary, on the interface between coextruded fluids, or inside the domain, the streamwise method aligns the mesh with the local velocity vector. ANSYS POLYFLOW applies the line kinematic condition on all mesh lines, both interior and on the boundary. In the direction of extrusion, the position is kept constant.

This method is an alternative to Optimesh for extrusion, and is well adapted to situations where large isotropic deformations occur. A typical example is 3D fiber spinning with large drawdown ratios. The streamwise method is applicable to both direct and inverse extrusion.

16.12. Elastic Methods

One of the options for adapting a mesh to the displacements of its boundary is based on an elastic material analogy. This remeshing technique belongs to the family of elliptic remeshing schemes and has the advantage of complete topological generality (in 2D and 3D), since no regular organization of the mesh is required. Its cost (in terms of CPU and memory) is comparable to that of the Thompson transformation described in *Thompson Transformation* (p. 348).

In the interior of the domain, a nonlinear small displacement pseudo-elastic problem is solved, stated in terms of displacement $\mathbf{d}$ as

$$\nabla \cdot \left(E\left(\nabla \mathbf{d} + \nabla^T \mathbf{d}\right)\right) = 0 \tag{16–15}$$

where E has the dimension of a Young's modulus and the Poisson's ratio is equal to zero.

For the improved elastic remeshing, *Equation 16–15* (p. 358) is modified as follows:

$$\nabla \cdot \left(\mathbf{E} \cdot \left(\overline{\nabla}\,\mathbf{d} + tf\,\tilde{\nabla}\,\mathbf{d}\right)\right) = 0 \tag{16–16}$$

where

$$\overline{\nabla}\,\mathbf{d} = \frac{1}{2}\left(\nabla \mathbf{d} + \left(\nabla \mathbf{d}\right)^T\right) \tag{16–17}$$

$$\tilde{\nabla}\,\mathbf{d} = \frac{1}{2}\left(\nabla \mathbf{d} - \left(\nabla \mathbf{d}\right)^T\right) \tag{16–18}$$

In *Equation 16–16* (p. 358), tf is a torsion factor. If tf is equal to zero, *Equation 16–15* (p. 358) and *Equation 16–16* (p. 358) are identical. A nonzero value for tf adds a rotation stiffness with respect to the initial configuration. This avoids deforming elements too much with respect to their initial shape. In ANSYS POLYFLOW, the value of tf is set to 5. Note that you cannot modify this parameter.

The improved elastic remeshing behaves in a similar way to the Optimesh technique. But whereas Optimesh cannot be used for meshes that require remeshing in all three dimensions, the improved elastic remeshing is applicable to any 3D mesh.

Equation 16–15 (p. 358) and *Equation 16–16* (p. 358) are nonlinear because they are solved in the deformed configuration and not on the original domain. It corresponds, however, to the small displacement formulation.

Free-surface or interface positions act as Dirichlet boundary conditions on *Equation 16–15* (p. 358) and *Equation 16–16* (p. 358) More precisely, a prescribed displacement is imposed in the direction perpendicular to the domain boundary, while zero-traction conditions apply in the tangential directions, leaving to the system the possibility of adapting the mesh tangentially. In regions where no particular displacement applies, a zero-normal/zero-tangential-traction condition is applied, so that tangential remeshing coming from adjacent boundary displacements is allowed.

Examples include cases where the inherent topological complexity makes it difficult (or even impossible) to apply topologically regular methods, such as 2D and 3D extrusion and 3D parison extrusion modeling.

Moreover, the coupling between the remeshing and the kinematic condition (line or surface) is treated differently. In the elastic remeshing, the physical dimension of the pseudo-Young's modulus E matters, and should be small enough so that the remeshing equation does not perturb the kinematic condition.

On the contrary, the improved elastic remeshing does not perturb the kinematic condition, and hence the pseudo-Young's modulus value in *Equation 16–16* (p. 358) has no importance.

Because the pseudo-Young's modulus must be small so as not to perturb kinematic conditions, and because of its nonlinearity (to allow the proper application of tangential remeshing), the elastic method may have a lower radius of convergence than the spine or Optimesh methods, so you may need to use an evolution strategy to obtain convergence. See *Evolution* (p. 517) for details about evolution.

The improved elastic method suffers less from this drawback. Because the improved elastic remeshing does not require a sliceable mesh, if you want to add a constraint on the free jet (which does require a sliceable mesh), you have to provide the inlet and outlet sections of the remeshing region for the slicing. In order to do that, after the selection of the **Improved Elastic remeshing** menu item, ANSYS POLYDATA displays the following question:

```
Is this remeshing defined on a free jet part and do you intend to stabilize the free jet?
```

If you click the **Yes** button, ANSYS POLYDATA asks you to select the inlet and outlet sections and the **Constraint on the free jet displacement** option will be available. If you click the **No** button, ANSYS POLYDATA does not ask any further questions and the **Constraint on the free jet displacement** option will not be available.

16.13. User Inputs for Remeshing

The procedure for defining the remeshing method is as follows:

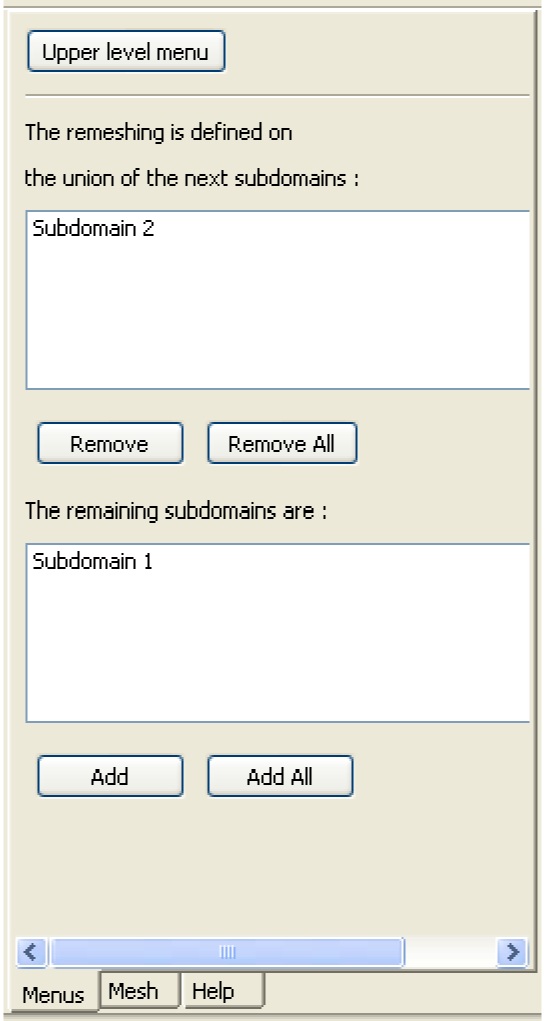
1. In the sub-task menu, select the **Global remeshing** menu item.

   Global remeshing

2. Specify the region where the first (or only) remeshing is to be performed. (See *Local Remeshing* (p. 365) for information about defining multiple remeshing regions.)

   1-st local remeshing

Figure 16.10 TheLocal remeshing domain Panel



The remeshing is defined on

the union of the next subdomains :

Subdomain 2

The remaining subdomains are :

Subdomain 1

In the **Local remeshing domain** panel (*Figure 16.10* (p. 360)), follow the appropriate steps below:

- If the subdomain or subdomains in the top list completely define the region where the remeshing is to be performed, simply click **Upper level menu** at the top of the panel.
- If the top list does not contain the correct subdomain(s), follow these steps:

  a. If there is a subdomain in the top list that is *not* a part of the region where the remeshing is to be performed, select it and click **Remove**. It will be moved from the top list to the bottom list, indicating that it will not be remeshed.

  b. If a subdomain that should be part of the remeshing region is not in the top list, select it in the bottom list and click **Add**. It will be moved from the bottom list to the top list, indicating that it will be remeshed.

  c. When the top list contains the correct subdomain(s), click **Upper level menu** at the top of the panel.

3. Select the remeshing method to be used, and set the relevant parameters. See *Choosing a Remeshing Technique* (p. 345) for guidelines on choosing the remeshing method.

   - The inputs for the Thompson transformation (described in *Thompson Transformation* (p. 348)) are as follows:

a. Select the **Thompson Transformation** menu item.

   Thompson Transformation

b. For each boundary (including each intersection with an adjacent subdomain), specify the imposed value for one of the g components.

   i. Select the boundary or adjacent subdomain in the **Condition on Global Field** panel.

   ii. Click **Modify**.

   iii. Select the appropriate component of the $\mathbf{g}$ vector (first, second, or (in 3D) third), and enter the **New value** for the selected component when prompted. See *Example* (p. 349) for more information.

   iv. Repeat for each boundary or adjacent subdomain.

   v. Click **Upper level menu** at the top of the panel when you are done.

- The inputs for the method of spines (described in *Method of Spines* (p. 346)) are as follows:

  a. Select the **Method of Spines** menu item.

     Method of Spines

  b. Specify the inlet for the system of spines (i.e., the initial spine for the remeshing domain) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

  c. Specify the outlet for the system of spines (i.e., the final spine for the remeshing domain) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

  The method of spines is not available in 3D.

- The input for the Euclidean distance method (described in *Euclidean Method* (p. 347)) is as follows:

  Select the **Euclidean Method** menu item.

  Euclidean Method

- The inputs for the method of planes (described in *Method of Planes* (p. 347)) are as follows:

  a. Select the **Method of Planes** menu item.

     Method of Planes

  b. Specify the inlet for the system of planes (i.e., the initial plane for the remeshing domain) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

  c. Specify the outlet for the system of planes (i.e., the final plane for the remeshing domain) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

- The inputs for the thin shell method (described in *Thin Shell Method* (p. 354)) are as follows:

  a. Select the **Thin Shell Method** menu item.

     Thin Shell Method

  b. Specify the master moving surface (commonly the boundary set that comes into contact with the mold) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

When there are several sub-tasks, one for each layer of the parison, the master surface must be defined only once. If the master surface has been specified previously (i.e., in another sub-task), you will just confirm it in this step.

Important

For multilayer applications, you must be sure to use the thin shell remeshing rule for all layers (i.e., in the sub-task for each layer).

- The inputs for Optimesh (described in *Optimesh* (p. 352)) in 2D are as follows:

  a. Select the **Optimesh-2D** menu item.

     Optimesh-2D

  b. For each boundary (including each intersection with an adjacent subdomain), specify the imposed value for one of the g components.

     i. Select the boundary or adjacent subdomain in the **Condition on Global Field** panel.

     ii. Click **Modify**.

     iii. Select the appropriate component of the $\mathbf{g}$ vector (first or second), and enter the **New value** for the selected component when prompted. See *Example* (p. 349) for more information.

     iv. Repeat for each boundary or adjacent subdomain.

     v. Click **Upper level menu** at the top of the panel when you are done.

- The inputs for Optimesh (described in *Optimesh* (p. 352)) in 3D are as follows:

  a. Select the **Optimesh-3D** menu item.

     Optimesh-3D

  b. Specify the inlet for the system of planes (i.e., the initial plane for the region to be sliced, described in *Euclidean Method* (p. 347) for the Euclidean method) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

     Important

     Recall that the slices must be perpendicular to the extrusion axis. ANSYS POLYDATA will check the inlet and outlet of the system of planes; they must be real planes and they must be perpendicular to the extrusion axis (x, y, or z direction).

  c. Specify the outlet for the system of planes (i.e., the final plane for the region to be sliced, described in *Euclidean Method* (p. 347) for the Euclidean method) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

Important

Note that the line kinematic condition is recommended when Optimesh is used. See *Discontinuity of the Normal Direction* (p. 318) and *Guidelines for 3D Extrusion Problems* (p. 328) for details.

- The input for the Lagrangian method (described in *Lagrangian Method* (p. 356)) is as follows:

  Select the **Lagrangian** menu item.

  Lagrangian

- The inputs for the thin shell method with Lagrangian master (described in *Thin Shell Method with Lagrangian Master* (p. 357)) are as follows:

  a. Select the **Thin Shell Method+Lagrangian master** menu item.

     Thin Shell Method + Lagrangian master

  b. Specify the master moving surface (commonly the boundary set that comes into contact with the mold) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

- The input for the Lagrangian method on the border only (described in *Lagrangian Method on Borders* (p. 357)) is as follows:

  Select the **Lagrangian on the border only** menu item.

  Lagrangian on the border only

- The inputs for the streamwise method (described in Section *Streamwise Method* (p. 357)) are as follows:

  a. Select the **Streamwise Method** menu item.

     Streamwise Method

  b. Specify the inlet for the system of planes (i.e., the initial plane for the remeshing domain) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

  c. Specify the outlet for the system of planes (i.e., the final plane for the remeshing domain) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

- The inputs for the elastic method (described in *Elastic Methods* (p. 358)) are as follows:

  a. Select the **Elastic remeshing** menu item.

     Elastic remeshing

  b. For each non-free-surface boundary (including each intersection with an adjacent subdomain), specify the type of constraints you want to impose.

     i. Select the boundary or adjacent subdomain in the **Condition on displacement along borders** panel.

     ii. Click **Modify**.

iii. To allow the points on the border to move along the tangent to the border, select **No NORMAL displacement** (the default condition). To fix the points so that they do not move in the normal or tangential direction along the border, select **No displacement**.

The default condition (**No NORMAL displacement**) allows for more degrees of freedom, and therefore produces smoother remeshing, *but* it can lead to a change in the geometry of a border that contains corners. The **No displacement** option is therefore recommended for borders that contain corners.

iv. Repeat for each boundary or adjacent subdomain.

v. Click **Upper level menu** at the top of the panel when you are done.

- The inputs for the improved elastic method (described in *Elastic Methods* (p. 358)) are as follows:

a. Select the **Improved Elastic remeshing** menu item.

Improved Elastic remeshing

b. Specify whether the remeshing is defined on a free jet and whether you intend to stabilize the free jet by clicking either **Yes** or **No**.

Important

In order to click **Yes**, you should verify that the remeshing domain is sliceable. There is no check performed in ANSYS POLYDATA related to the sliceability of the mesh, but ANSYS POLYFLOW will stop with an error message if the mesh is not sliceable. Furthermore, you should verify that the remeshing domain and constraint of the free jet is defined appropriately. If the remeshing domain includes the free jet and the die domains, then the constraint of the free jet should be defined such that the starting coordinates for the portion of the remeshing domain where the constraint is applied is after the die end. See *Constraint on Global Displacement* (p. 335) for more details.

If you clicked **Yes**, perform the following:

i. Specify the inlet for the system of planes (i.e., the initial plane for the remeshing domain) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

ii. Specify the outlet for the system of planes (i.e., the final plane for the remeshing domain) by selecting the intersection with the appropriate boundary set or subdomain and clicking **Confirm**.

c. For each non-free-surface boundary (including each intersection with an adjacent subdomain), specify the type of constraints you want to impose.

i. Select the boundary or adjacent subdomain in the **Condition on displacement along borders** panel.

ii. Click **Modify**.

iii. To allow the points on the border to move along the tangent to the border, select **No NORMAL displacement** (the default condition). To fix the points so that they do not move in the normal or tangential direction along the border, select **No displacement**.

The default condition (**No NORMAL displacement**) allows for more degrees of freedom, and therefore produces smoother remeshing, *but* it can lead to a change in the geometry of a border that contains corners. The **No displacement** option is therefore recommended for borders that contain corners.

iv. Repeat for each boundary or adjacent subdomain.

v. Click **Upper level menu** at the top of the panel when you are done.

4. Modify the element distortion check information, if necessary. After completing the inputs for your selected remeshing method, ANSYS POLYDATA will bring you to the **Element distortion check** menu, where you can make changes to the criteria ANSYS POLYFLOW will use to check for element distortion as a result of the remeshing. The default settings are appropriate for most cases, so you can generally select **Accept the current setup** without making any changes.

   If any of the distortion limits is met during an evolution or time-dependent simulation, ANSYS POLYFLOW will stop the calculation or simply warn you, depending on what you have specified in this menu.

   When using **Stop if distortion limit exceeded** mode, the calculation of the current time step is stopped and restarted with a time step divided by 2. This option is to be used combined with Adaptive Meshing Technique (see *User Inputs for Adaptive Meshing* (p. 368) for frequency of the meshing).

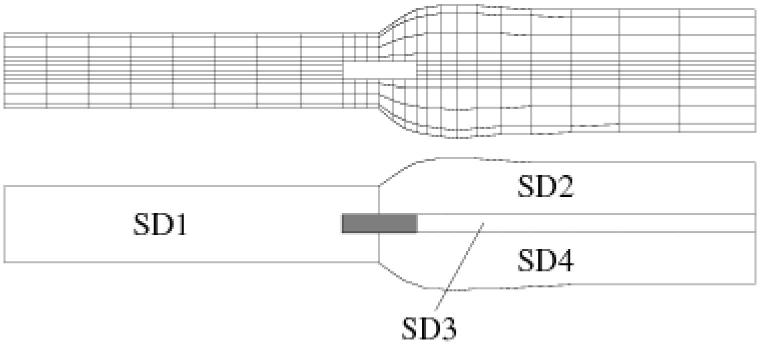
16.14. Local Remeshing

For a flow problem involving moving boundaries, each sub-task defined in ANSYS POLYDATA has its own remeshing rule. This is, in fact, the most common situation. There are some situations, however, where it is desirable to define more than one remeshing method for a single sub-task. You can accomplish this by defining several local remeshing regions within the domain of the sub-task.

For example, the flow may require the use of the method of spines in one part, while the Thompson transformation may be preferred for another part. The definition of more than one local remeshing region requires that there are multiple subdomains in the domain of the sub-task. Each local remeshing region will be identified by the subdomain(s) that comprise it.

A typical situation where several local remeshing rules are needed arises when a topological restriction of a particular remeshing rule makes it inappropriate over the entire domain of the sub-task, although it is perfectly acceptable for a portion of the domain. As an example, consider the 2D problem illustrated in *Figure 16.11* (p. 365).

Figure 16.11 Application of Local Remeshing



This problem involves two free surfaces and an obstacle in the middle of the domain. A single material is used, so there is just one sub-task. However, the mesh topology prevents the use of a single spine technique applied on the entire domain (due to the presence of the obstacle in the middle). In this case, two independent spine techniques (one for each free surface) will be defined. With separate remeshing rules in subdomains 2 and 4, the presence of the obstacle no longer presents any difficulty.

16.14.1. User Inputs for Local Remeshing

To define multiple remeshing rules for a single sub-task, follow the steps in *User Inputs for Remeshing* (p. 359) to set up the first remeshing rule. Then follow these additional steps:

1. Create another local remeshing rule.

   Creation of a local remeshing
2. Define the remeshing domain, remeshing method, and associated parameters, as described in *User Inputs for Remeshing* (p. 359) for the first remeshing rule.
3. Repeat these two steps until you have defined all the remeshing rules you need.

If you need to delete one of the remeshing rules you have defined, select the **Deletion of a local remeshing** menu item and choose the remeshing rule to be removed.

16.15. Adaptive Meshing

In a transient simulation, it is possible for the mesh-quality criteria to change over time, so that a mesh that was acceptable at the start of the simulation is no longer adequate later on in the simulation. The adaptive meshing technique is provided to address these issues.

16.15.1. Overview and Usage

When adaptive meshing is used, the quality of the mesh is evaluated at regular intervals during the transient simulation. For each element of the mesh, if the current quality is below a specified threshold, then the element is selected for refinement or replacement. Different criteria have been developed for different types of simulations:

- For problems that use the mesh superposition technique (MST) described in *Flows with Internal Moving Parts* (p. 457), it is important to have a good mesh near the border of the moving parts. A nonzero gradient of the “inside" field (described in *Adaptive Meshing Parameters for Moving Parts* (p. 370)) indicates this area. If the gradient inside a given element is above the specified threshold, it needs to be refined. If the gradient is close to zero, the element can be “unrefined" (coarsened), if necessary.

  The problems involving large variations of fields is based on fields and domains that the user prescribes. The domains and the fields available depend on type of simulation involved. Elements where the variation of the selected fields is too high, are refined.
- For problems involving transport of species, the criterion is similar to that described above for the MST, and is based on the gradient of the concentration field.
- For blow molding of thick parisons, elements may become distorted or stretched over time. Poor-quality elements are detected and replaced by elements of better quality.
- For blow molding simulations in general, it is important to modify the size of elements in the flow domain based on the shape of the mold in front of them. Larger elements are needed in areas that are in contact with the flat zone of the mold, and smaller elements are needed in areas where the mold is curved.

Moreover, it is important to anticipate the contact, in order to have smaller elements before contact (after contact, the elements no longer move so it is not necessary to refine them).

16.15.2. Adaptive Meshing Technique

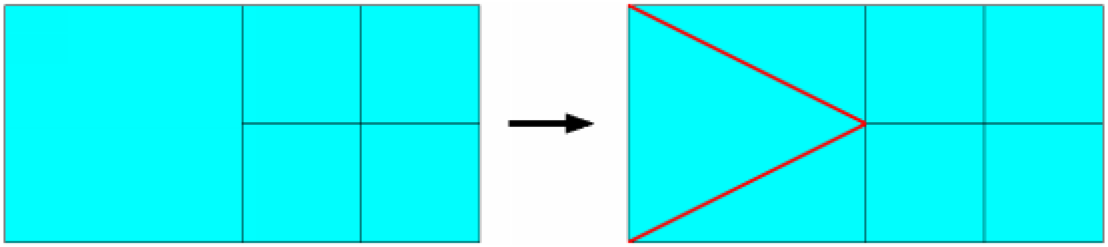
Depending on the type of problem, adaptive meshing method uses two types of techniques:

- Recursive subdivision of elements
  - A segment is divided into 2.
  - A surface is divided into 4.
  - A brick or wedge is divided into 8.
  - A tetrahedron is divided into 12.
  - A pyramid is divided into 10.

  Note that during the subdivision, an element may be subdivided while its neighbor is not, resulting in a non-conformal mesh. In such cases, the solver creates “constraints of conformity” for the midface and/or midsegment nodes, in order to guarantee the continuity of the fields (e.g., velocity, stresses, temperature, and thickness).

  For 2D and shell meshes, you can specify that a “conformalization” step is undertaken after subdivision, during which the elements adjacent to the subdivided elements are replaced with elements that yield a conformal mesh (see *Figure 16.12* (p. 367)). Conformalization thus avoids the automatic creation of constraints of conformity.

  Figure 16.12 Example of Conformalization



- “TGrid" method
  - A triangle or tetrahedron is replaced by multiple elements of the same type (smaller and/or with a better shape).
  - This method is available only for triangles in 2D and tetrahedra in 3D.

For MST simulations and simulations involving species (concentration fields) transport, the recursive subdivision technique is used. For simulations with fixed meshes, subdivision technique is used to catch large variations of fields (velocity, viscoelastic tensions, etc.).

For blow molding simulations (i.e., simulations involving contact), the recursive subdivision technique is used with shell elements and the TGrid method is used with volume elements. For simulations that use the Thompson, Lagrangian, or elastic remeshing methods, the TGrid method is used for all cases except for shell blow molding (as noted above). The two methods cannot be used simultaneously.

Note

When using TGrid method for adaptive meshing, the domains that will be remeshed have to be composed with triangles in 2D or tetrahedra in 3D.

16.15.3. User Inputs for Adaptive Meshing

The procedure for defining adaptive meshing is as follows:

1. In the task menu, select the **Numerical parameters** menu item.

   Numerical parameters

2. In the **Numerical parameters** menu, select **Adaptive meshing**.

   Adaptive meshing

3. Enable the appropriate type of meshing by selecting the corresponding item in the **Adaptive Meshing** menu:

   - For problems involving internal moving parts (i.e., that use the MST), select **Activate adaptive meshing for moving parts**.

     Activate adaptive meshing for moving parts

   - For problems involving species transport, and big variations of fields on fixed meshes, select **Activate adaptive meshing for large variations of fields**.

     Activate adaptive meshing for large variations of fields

   - For problems involving shell blow molding (contact), select **Activate adaptive meshing for contacts**.

     Activate adaptive meshing for contacts

   - For all other blow molding simulations, select both **Activate adaptive meshing for contacts** *and* **Activate adaptive meshing for remeshings**.

     Activate adaptive meshing for contacts

     Activate adaptive meshing for remeshings

     Activate adaptive meshing for remeshings is available only if you are using the Thompson, Lagrangian, elastic, or improved elastic remeshing method.

4. Set the parameters for the type of adaptive meshing you have selected.
5. Select **Upper level menu** to return to the **Adaptive Meshing** menu.
6. Set the general adaptive meshing parameters.

   a. Specify the frequency of the meshing.

      Modify Nstep

A value of 1 (the default) indicates that the adaptive meshing will be performed at each time step. This value is appropriate for MST simulations, because the fluid region is changing significantly from one time step to the next. A value of 3 is good for species transport simulations.

For simulations involving large variations on fixed mesh, a value of 1 might be necessary to assume good convergence scheme. A value of 5 is usually sufficient for contact and remeshing simulations.

For transient simulation using the TGrid method, a value smaller than 4 leads to unpredictable evolution scheme for the time step so that it will not be adapted with predictor-corrector scheme. The time step remains constant through out the simulation but can be reduced if the solution does not converge.

Note

A reliable alternative is to remesh every 4 time steps and to use the distortion check.

Stop to avoid too large deformations of the mesh between two remeshings.

b. Specify the number of times a primitive element (i.e., an element of the initial mesh, as created in the mesh generator) can be subdivided recursively.

Modify Maxdiv

A value of 3 (the default) means that a given surface of area S:

- will have 4 subfaces of area $S/4$ after one subdivision
- can have up to 16 subfaces of area $S/16$ after two subdivisions
- can have up to 64 subfaces of area $S/64$ after three subdivisions

A value of 3 (the default) also means that given brick of volume V:

- will have 8 sub-bricks of volume $V/8$ after one subdivision
- can have up to 64 sub-bricks of volume $V/64$ after two subdivisions
- can have up to 512 sub-bricks of volume $V/512$ after three subdivisions.

A value greater than 3 will lead to a large increase in the number of elements. For 3D, a maximum value of 2 is recommended.

c. Enable partial triangulation 3D.

Remeshing with TGrid depends on criteria that select elements to remesh. If **Full triangulation** is used, when one element is selected for remeshing, the whole moving domain (that is, the domain for which there are local criteria activated) will be remeshed. With big meshes, the cost of this operation could be too high.

A better solution is the **Partial Triangulation 3D**. When one element is selected for remeshing, a zone defined upon the neighbors of this element will be remeshed (and not the entire mesh). This option is preferable for large meshes.

d. For 2D and shell meshes that use the recursive subdivision technique, you can enable conformalization for the mesh.

   Enable Mesh Conformalization

   The conformalization takes place after the subdivision; it ensures that the mesh only contains conformal elements (see *Figure 16.12* (p. 367)) and eliminates all the constraints that are needed to guarantee the continuity of the fields for midsegment nodes. Conformalization is useful in the following circumstances: when your postprocessor does not support or does not correctly handle non-conformal meshes; when you want to avoid the wiggles that can appear in the solution when contact is establishing; when you want to avoid the conflict that can arise between constraints of conformity and other constraints (e.g., those created for fluid-solid contact handling and connected boundaries).

   Note that conformalization can also be applied on existing non-conformal meshes, as described in *Converting a Shell Mesh and Results* (p. 168).

If you want to reset the frequency and number of subdivisions and the Partial/Full triangulation switch to the default values, select **Reset parameters of adaptive meshing (to default values)**.

16.15.3.1. Adaptive Meshing Parameters for Moving Parts

For adaptive meshing with moving parts, you can specify the moving parts for which adaptive meshing should be performed. You can also specify different adaptive meshing parameters for each of the moving parts, or specify a single set of parameters that apply to all of them.

When you select **Activate adaptive meshing for moving parts**, as described above, you will enter the **Global criteria for moving parts** menu, where you can follow the steps below to set the relevant parameters:

1. To enable adaptive meshing for all of the moving parts, select **Enable all the local criteria**.

   Enable all the local criteria

   If you want to enable (or disable) adaptive meshing for one or more individual parts, select the **Criteria on** menu item for the relevant part, select **Enable current criteria** (or **Disable current criteria**) in the **Local criteria for moving parts** menu, and then select **Upper level menu** to return to the **Global criteria for moving parts** menu.

2. Set the parameters for meshing, S_{up} and S_{do}, as described below.

   Modify S_up

   Modify S_do

   Each time you modify the value of a parameter, ANSYS POLYDATA will ask you if you want to apply the new value to all of the moving parts. If you click **Yes**, the new value of S_{up} or S_{do} will be assigned to all of the moving parts for which adaptive meshing has been enabled.

   In the mesh superposition technique, each node in the flow domain is evaluated to determine if it lies in the real fluid region or in a moving part. This information is stored in a field called the “inside" field. At each nodal point, if the value of the inside field is 0, the point is in the fluid region;

otherwise, its value is 1. In order to better represent the exact shape of the moving part by the inside field, the mesh must be refined in the neighborhood of the moving part's border.

The border of a moving part can be identified as the region of the flow domain where the gradient of the inside field is nonzero. Thus, the criteria for adaptive meshing are as follows:

- If the variation of the inside field across an element is greater than S_{up}, the element is subdivided. The default value for S_{up} is 0.05.
- If the variation of the inside field across an element is less than S_{do}, the element is unrefined (coarsened). The default value for S_{do} is 0.01.

  If the simulation is transient, then as a moving part rotates, all of the elements in the flow region will come into contact with its border, and they will therefore be subdivided. As the part moves away, these additional subelements are no longer necessary, and their presence will lead to an increase in CPU time and memory usage. To avoid this problem, ANSYS POLYFLOW will remove the subelements in the areas distant from the borders of the moving part.

If you want to return to the default global values of S_{up} and S_{do}, select **Reset global parameters (to default values)**.

3. If you want to modify the S_{up} and S_{do} values for a particular moving part, select the **Criteria on** menu item for the relevant part, and then modify the values in the **Local criteria for moving parts** menu. If you want to copy the global values of S_{up} and S_{do} to this moving part, select **Reset local parameters (copy global param.)**.

16.15.3.2. Adaptive Meshing Parameters for Large Variations of Fields

For transport of species simulations, the local meshing criteria are defined and you can choose to activate or deactivate them. For the adaptive meshing for large variations of fields, you must define your own local criteria for the remeshing. The procedure is as follows:

1. Select the field for which you want to define a criteria. The fields available depend on the type of the simulation you do.
   - For generalized Newtonian flows:
     - Normalized velocity
     - Each component of the velocity
     - Temperature (if non isothermal flow)
     - Local shear rate (if postprocessor defined)
   - For Darcy flow problems:
     - Pressure Darcy
   - For electrical potential problems:
     - Potential
   - For viscoelastic flows:
     - First invariant of tensions
     - Second invariant of tensions

2. Choose the domains you want to remesh. You can choose from the list of domains available for each field selected.

   When the local criterion is defined ANSYS POLYDATA returns again to the **Global criteria for large variations of fields** menu. You can see the criteria you have defined in the list of Local Criteria.

3. Repeat the procedure for defining other criteria or for modifying one.

> **Note**
>
> When a local criterion is defined, you cannot remove it from the list but only deactivate it.

The parameters to be set, and the procedures for setting them, are the same as for adaptive meshing for moving parts, as described in the previous section. The local criteria must be preferred in place of the global criteria because same parameters cannot be applied for different fields.

16.15.3.3. Adaptive Meshing Parameters for Contact and Remeshing

For blow molding simulations, which involve both a contact problem and remeshing for the parison deformation over time, you will specify two groups of parameters for adaptive meshing: one for the contact, and one for the standard remeshing.

16.15.3.3.1. Adaptive Meshing for Contact

For the contact problem, the adaptive meshing will refine the mesh in regions of the parison that are close to the mold and close to a region where the curvature of the mold is high (low radius of curvature). It is not necessary to refine the mesh when the parison is far from the mold, or if the parison comes into contact with a region of the mold where the curvature is low (high radius of curvature = a flat area).

This behavior can be modeled mathematically, as follows. For a given element E of the parison, with a size S_{cur}, ANSYS POLYFLOW searches for the closest node N of the mold. The distance between element E and node N is denoted as d, and the local radius of curvature at node N is denoted as r. Element E will be refined if its proposed new size S_{new} is less than S_{cur}, where the new size is computed from

$$S_{new} = \max\left[(Ar + C) \left(1 + \frac{d^2}{B^2}\right), S_{min} \right] \tag{16–19}$$

where A is the fraction of the radius of curvature, Ar is the desired size of element E when E is in contact with the mold, B is the typical distance to the mold, C is a coefficient of proportionality, and S_{min} is the minimum size for an element in the parison. Refinement begins when the distance d between element E and the mold is less than B. Defining the parameters allows you to modulate the refinement along the free surface. C may be interpreted as the intended size of the parison elements when the contact occurs along a flat part of the mold. There are two extreme scenarios:

- $A = 0$ and $C \neq 0$

These values should be used when the contact occurs along a flat part of the mold.

- $A \neq 0$ and $C = 0$

  These values are recommended for contact that occurs along a curved mold.

Nonzero values for both parameters A and C must be avoided or used very carefully. If the contact occurs along a flat part of the mold, Ar is very large and $(Ar + C)$ is even larger, and hence, the new size will never be smaller than the current size.

When you select **Activate adaptive meshing for contacts**, the **Global criteria for contacts** menu appears, where you can specify the values of the parameters A, B, C, and S_{min} from *Equation 16–19* (p. 372). The procedure for enabling adaptive meshing for one or more contact boundaries is the same as that for moving parts.

To specify the values of the parameters, select the relevant menu items:

Modify A

Modify B

Modify C

Modify Smin

The parameter A is dimensionless, and its default value is 0.2. This value should lead to almost 30 elements to mesh a circle. The parameters B and C have units of length. The default value for B is the typical size of segments of the mesh, whereas for C it is zero. The parameter S_{min} also has units of length, and its default value is 10% of the typical size of the segments in the mesh.

16.15.3.3.2. Mapping

The TGrid remeshing algorithm builds a new mesh on the basis of the old mesh. It would be preferable to base the new mesh on the original geometry data of the fluid boundary, but unfortunately this information is no longer available. Hence, the new nodes are located on the boundary faces of the old mesh. Because of this, the mesh increasingly deviates from the surface it is intended to represent with each remeshing. Depending on the concavity, the fluid may penetrate the mold or detach from the mold.

To avoid such undesirable effects, the new boundary nodes in contact with the mold are relocated onto the mold surface. This is done by a mapping technique which projects the nodes along the local normal to the free surface.

Along planes of symmetry, the normal is corrected such that nodes in a plane remain in the same plane after the projection. This correction should be activated each time symmetry planes are defined in the flow sub-task. In order to do this, ANSYS POLYDATA detects the planes of symmetry defined in the flow sub-task and computes the coefficients (A, B, C and D) of the plane equation: $Ax + By + Cz + D = 0$. These coefficients are computed based on the coordinates of the nodes belonging to the plane. If those nodes are not located precisely in the plane, the coefficient values may not be as expected (e.g., 1,99998 instead of 2). ANSYS POLYDATA will give you the opportunity to correct them.

The mapping algorithm is based on the following four parameters: a threshold value that triggers the mapping of nodes, a tolerance value named epsilon, an element dilatation, and a maximum displacement. These parameters are defined as follows:

- Threshold

  The contact algorithm uses internal fields that are named **contact_field** and are defined on the free surface. This field stores the contact information for each node and initially has a value of either 0 or 1. If a node of the free surface is in contact, the field value is 1, otherwise the value is 0. However, after a remeshing step, the **contact_field** values must be interpolated onto the newly generated mesh. After this interpolation, the **contact_field** values are no longer limited to being either 0 or 1, but can be intermediate values (e.g., 0.3). If a node of the free surface has a **contact_field** value greater than the threshold, the node is assumed to be in contact and will be mapped. The default value of the threshold is 0.8.

- Epsilon

  This parameter acts as a tolerance on the minimum distance between points. Points that are closer than this distance are assumed to be at the same location. The default value is calculated from the typical mesh size.

- Element dilatation

  This element dilation has the same meaning as the one in the *Setting Up a Contact Problem* (p. 391) related to the setup of a contact detection problem. The elements of the mold are slightly dilated in order to facilitate the detection algorithm. The default value is calculated from the typical mesh size.

- Maximum displacement

  In order to avoid highly distorted elements in the layer of elements adjacent to the free surface, the displacement of the mapped nodes is limited to this value. A good practice is to define the maximum displacement as 10–25% of the minimum element size imposed in the adaptive meshing setup (*Equation 16–19* (p. 372)). The default value is calculated from the typical mesh size.

The procedure for defining the mapping is as follows:

1. In the **Global criteria for contacts** menu, select the **Define Mappings** menu item.

   Define Mappings

   The **Mapping of free surfaces on molds** menu will appear. You have the following options:

   - You can activate mapping contact boundary by contact boundary by selecting the **Enable mapping along contact boundaries of mold [name of mold #i]** menu item.

     Enable mapping along contact boundaries of mold [name of mold #i]

   - You can activate the mapping for all contact boundaries at once, by selecting the **Enable Mapping along all contact boundaries** menu item.

     Enable Mapping along all contact boundaries

   - You can disable the previous settings by selecting the following menu items:

     Disable mapping along contact boundaries of mold [name of mold #i]

or

Disable Mapping along all contact boundaries

2. After you have enabled the mapping, the menu is updated and the **Numerical parameters for mapping along contact boundaries of mold [name of mold #i]** menu item appears. Select this item to modify the parameters needed to perform the mapping.

   Numerical parameters for mapping along contact boundaries of mold [name of mold #i]

   - To modify the threshold value allowing the mapping of a node, select the **Modify threshold value = [current value]** menu item.

     Modify threshold value = [current value]

   - To modify the minimum distance, select the **Modify epsilon value = [current value]** menu item.

     Modify epsilon value = [current value]

   - To modify the element dilatation, select the **Modify element dilatation = [current value]** menu item.

     Modify element dilatation = [current value]

   - To modify the maximum displacement, select the **Modify maximum displacement = [current value]** menu item.

     Modify maximum displacement = [current value]

   - To reset the default values, select the **Reset to default values** menu item.

     Reset default values

   - To return to the **Mapping of free surfaces on molds** menu, select the **Upper level menu** menu item.

     Upper level menu

3. If the flow sub-task contains planes of symmetry, they must be activated for the mapping. To do so, select the **Planes of symmetry management** menu item.

   Planes of symmetry management

   The **Planes of symmetry management** menu will appear, and ANSYS POLYDATA will summarize the features and the status of each plane in the upper part of the panel. The status indicates whether the plane is activated, while the features provide information about its orientation. For the orientation, ANSYS POLYDATA detects whether the plane is perpendicular to an axis of the coordinate system. The following is an example of the information displayed for a plane that is perpendicular to an axis:

   ```
   Plane id 1 - plane perpendicular to X axis. - activated
   ```

The addition of `- activated` in the previous example indicates that the plane of symmetry has been activated for the mapping.

The following is an example of the information displayed for a plane that is not perpendicular to an axis:

```
Plane id 2 - A=33, B=0.72, C=2.000001, D= 0.01
```

You have the following options:

- You can activate all of the planes of symmetry by selecting the **Activate all planes (for mapping)** menu item.

  Activate all planes (for mapping)
- You can activate individual planes of symmetry by selecting the **Activate plane #i** menu item.

  Activate plane #i
- You can deactivate all or single planes of symmetry by selecting the **Desactivate all planes (for mapping)** item or the **Desactivate plane #i** item, respectively.

  Desactivate all planes (for mapping)

  or

  Desactivate plane #i
- To correct the coefficient of the plane equation, select the **Modify coefficients of a plane** menu item.

  Modify coefficients of a plane

  When prompted, specify the plane index and the coefficient values.

16.15.3.3.3. Adaptive Meshing for Remeshing

For adaptive meshing for remeshing, ANSYS POLYFLOW will refine elements that have become too stretched and/or too distorted due to remeshing. An element E will be refined if its quality is lower than the threshold quality ($T_{quality}$).

A quality value of 1 means that the element is not stretched or distorted at all (e.g., a square), and a value of 0 means that the element is flat. The lower the value, the shape of the element will be more distorted. All of the elements selected for refinement will be removed from the mesh, and replaced by tetrahedra (in 3D) or triangles (in 2D) with the specified size S.

For shell blow molding, since the recursive subdivision technique is used for refinement, an element selected for refinement because of its bad shape will be replaced by subelements with the same bad shape. Thus, it is not useful to use adaptive meshing for remeshing in a shell blow molding simulation.

When you select **Activate adaptive meshing for remeshings**, the **Global criteria for remeshings** menu appears, where you can specify the values of the parameters $T_{quality}$ and S (the size for new elements). The procedure for enabling adaptive meshing for remeshing is the same as that for moving parts.

To specify the values of the parameters, select the relevant menu items:

Modify Tquality

Modify Size

$T_{quality}$ is dimensionless, and its default value is 0.25. S has units of length, and its default value is the typical size of segments in the mesh. To obtain better shaped elements, it is recommended to specify a $T_{quality}$ of 0.85 in 2D and 0.5 in 3D.

S has units of length, and its default value is the typical size of segments in the mesh. A modification on the values of size has been done between version 3.9.2 and 3.10 version of POLYFLOW. If you specify an element size S with the 3.9.2 version, divide it by the square root of the dimension of the root mesh (3 in 3D, 2 in 2D) to obtain the same results with the 3.10 version.

It is important to note that the algorithm will pick up the elements that are picked for adaptive meshing, if at least one of the following conditions is met:

- The quality is below the specified value of $T_{quality}$.
- The size is not within the interval $\left[\frac{S}{2}, 2S\right]$.

Here, it is reasonable to select a size S which is compatible with the initial mesh. If you specify a smaller value for the size S, then elements satisfying the quality requirement will be remeshed and refined in order to match the size criterion.

16.15.3.4. Manage Zones for Remeshing

In many applications involving large deformations of the mesh (glass pressing), the distortion mainly appears in some localized zones of the mesh. The features and details of the geometry that need small size for elements are also local in the mesh. The **Manage zones for remeshing** menu allows you to define very local regions where the size of the elements must be smaller to catch details of a geometry. The two types of zones available are: fixed zones and moving zones. The procedures for defining these zones are explained in the following sections.

16.15.3.4.1. Defining Fixed Zones

Boxes
Define the lower left corner and the upper right corner of the box.

Circles/Spheres
Define the center and the diameter of the zone.

1. Click the **Upper level menu** when the dimensions of the zone are correctly defined.
2. Choose an appropriate element size for the zone, constant, or linear function of coordinates.

Note

The values used for the element size are very sensitive for the remeshing. It is recommended not to define an element size smaller than half the element size specified in the local criteria menu. If greater ratios are needed, a good solution is to define embedded zones with decreasing sizes to avoid badly shaped elements.

3. Click the **Upper level menu** when the element size of the zone is defined. The **Manage zones for remeshing** menu summarizes information about existing zones.

16.15.3.4.2. Defining Moving Zones

For glass pressing simulations, the peripheral material progressing along the molds is a critical region where the size cannot be too large. It is impossible to define a fixed zone because the periphery of the material is always moving. You could define a box that surrounds the whole flow region but this method demands the use of small elements over a large area and can lead to excessive CPU costs.

The **Along boundaries zone** option solves this problem. When you select this type of zone, you must specify the free surface on which you want to attach the moving zone. Choose the free surface and click the **Upper level** menu.

The parameters for the size are similar to those for adaptive meshing parameters for contact. The size is calculated along the curvature of the free surface using the (*Equation 16–19* (p. 372)). The parameters a and S_{min} have identical meaning. The parameter b defines the depth of the zone.

16.15.3.5. Criteria for Remeshing

You can define remeshing using the following procedure:

1. Define global or local criteria for adaptive meshing for remeshing.
   a. Define $T_{quality}$ for each subdomain (Local Criteria) or for all subdomains (Global Criteria).
   b. Define **Size** for each subdomain (Local Criteria) or for all subdomains (Global Criteria).
2. If there are contact problems, define global or local criteria for adaptive meshing for contact.

   Define S_{min}, a, and b for each contact problem (Local Criteria) or for all contact problems (Global Criteria).
3. If some details need to be specified more accurately, define the zones.

If several sizes are defined simultaneously on an element through multiple criteria, the minimum of the size of all criteria encountered on this element is applied for the new size. It is important to understand that the criteria for adaptive meshing for contact and the zones for remeshing criteria only serve to define areas where size has to be smaller than that inside the subdomains. The element sizes specified in point 2 and 3 of the above procedure are smaller than that used in point 1.

Chapter 17: Blow Molding and Thermoforming

This chapter describes ANSYS POLYFLOW's contact detection capability, which is commonly used to model blow molding and thermoforming applications. See *Free Surfaces and Extrusion* (p. 311) for information about free surface problems and *Remeshing* (p. 343) for information about remeshing.

Information in this chapter is presented in the following sections:

17.1. Working of Contact Detection

ANSYS POLYFLOW has the ability to detect contact between a free surface and a wall that occurs over time. The wall is typically a mold for blow molding and thermoforming simulations, although the contact detection feature can be applied to other processes. The contact detection algorithm is applicable only for time-dependent problems.

After a free surface and wall have come into contact, it is possible to simulate their detachment (i.e., contact release) under specific circumstances. Contact release typically takes place when the motion of a mold is reversed, or when the forming pressure results in the material being pulled away from a mold.

Blow molding is an important processing method for molding hollow articles such as bottles. The preform (parison) is usually made by extrusion or injection molding, and it is then forced between the mold halves by pressurization (blowing air). The polymer solidifies upon contact with the cold mold, and the finished product is ejected.

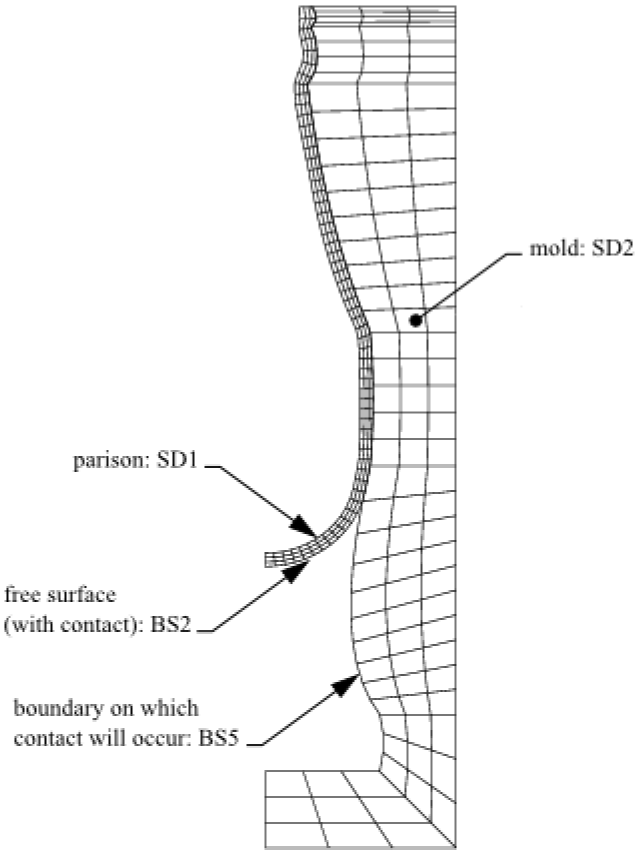
Contact detection is a *local* procedure that will be performed at each location *along a free surface*. As for all free surface problems, displacements of the free surface require the use of a remeshing scheme for the interior nodes. Some remeshing methods described in *Remeshing* (p. 343), are particularly adapted to thin shell configurations. The use of a Lagrangian representation significantly facilitates calculations involving contact detection. This is reflected in the remeshing techniques that are available.

The context in which contact detection takes place is a free surface with limited space for expansion, such as in blow molding. The motion of the "solid" regions used to detect contact with the fluid can be prescribed with a velocity or a force in order to introduce a mold displacement in the simulation. This displacement will be calculated as the time integration of assigned or resulting velocity. This velocity or force can be a function of time, and cannot be imposed simultaneously.

Contact detection is available for both 2D and 3D problems. Contact detection is an attribute of *each* free surface. When contact detection is turned on for a particular free surface, you will need to specify the solid domain at the origin of the contact and the boundary of this solid domain along which contact will be detected.

Consider, for example, the simple 2D blow molding application illustrated in *Figure 17.1* (p. 380), where the polymer has been defined to be SD1. Contact conditions for free surface BS2 are defined, so penetration will be checked with SD2. The contact surface is BS5.

Figure 17.1 Example of Contact Detection in Blow Molding



From this example, it is clear that the solid domain in which contact will be detected cannot be included in the domain of the sub-task. ANSYS POLYDATA ensures that this criterion is satisfied.

17.1.1. Shell Elements for 3D Models

For 2D blow molding or thermoforming problems, standard volume elements (i.e., quadrilaterals and triangles) are always used, because CPU time is not prohibitive. For 3D cases, however, the use of shell elements is strongly recommended. Shell elements are either defined on a boundary of a 3D volume region, or simply generated as 2D elements in 3D space.

The basic assumption of the shell element is that the thickness is small compared to other dimensions of the parison. In this case, the thickness is taken as an independent variable that is defined over the calculation domain (i.e., the shell). The mass conservation equation, when derived for the shell elements, leads to a transport equation for the thickness of the parison. The thickness appears as a separate variable that replaces the pressure and is computed together with velocities and node positions.

The shell element is not only much faster than the volume element, it is also much more robust in 3D calculations because there is no need to maintain good mesh orientation in the direction of the thickness. The shell element's major assumption is that shear effects are neglected.

See *Generating a Mesh with Shell Elements* (p. 401) for information about creating a mesh with shell elements in it.

17.2. Theory and Equations

In order to define boundary conditions for contact detection, a distinction must be established between regions of the free surface where contact has not yet been detected and regions where contact has occurred. Before contact, dynamic and kinematic conditions (*Equation 15–1* (p. 312) and *Equation 15–2* (p. 312)) dictate the displacement of the free surface. A normal force can be applied on an internal free surface to take the inflation into account. Once contact occurs, free surface displacement becomes impossible in the direction normal to the mold surface, and the velocity of the fluid becomes identical to the velocity of the mold. In the tangential direction, a “slipping" law can be defined. At the same time, the force vector f (normal and tangential components) is no longer prescribed: an essential condition on the velocity replaces the force condition that was previously prescribed.

17.2.1. Penalty Technique for Detecting Contact

In ANSYS POLYFLOW, contact is implemented through a penalty technique. Once a node *slightly* penetrates a solid domain, the discrete momentum equation of the node is modified by a penalty term, which can be different in the normal and tangential directions in order to model the physics accurately.

In the normal direction, the condition that the fluid velocity must be equal to the wall velocity is enforced with the penalty coefficient k:

$$f_n = -k\,(\mathbf{v} - \mathbf{v}_{wall})\cdot\mathbf{n} \qquad \textbf{(17–1)}$$

where n is the normal to the free surface.

The coefficient k has physical dimensions, which should be consistent with the system of units used for the simulation.

The product of k and a typical value for the relative parison velocity counterbalances the pressure applied to inflate the parison. The larger the value of k, the closer the velocity of the fluid will be to the velocity of the wall (in the normal direction for *Equation 17–1* (p. 381)). On the other hand, if k is not large enough, the fluid will progressively penetrate further into the mold and contact detection at the surface will be less accurate.

A large value of k will not, in general, result in excessive stiffness of the equation system, provided that numerical round-off errors are not encountered (i.e., provided that all numbers can be accurately represented in the computer). By default, ANSYS POLYDATA uses a value of $k = 10^9$, which is appropriate for most cases. When contact detection is modeled, the time-marching parameters must be modified as described in *Time Dependence and Contact Handling* (p. 389) and *Inputs for 2D Contact Detection* (p. 392).

The technique is similar in the tangential direction. The only difference is that a prescribed slip may be required for physical reasons. The tangential velocity is prescribed through a linear law of the “slip" type, given by

$$f_s = -F_{slip}\,(\mathbf{v}_s - \mathbf{v}_{s,wall}) \qquad \textbf{(17–2)}$$

Note that, like all slip coefficients, F_{slip} has physical dimensions that must be checked. The default value of F_{slip} in ANSYS POLYDATA is 10^9, which usually corresponds to a zero relative velocity condition in the tangential direction. Possible slipping is considered if F_{slip} is lower than the penalty coefficient k in the normal direction. As mentioned above for the value of k, a large value of F_{slip} does not cause excessive stiffness of the equation system, provided that numerical round-off errors are not encountered.

17.2.2. Contact Release

As mentioned previously, the simulation of contact release can occur under specific circumstances, e.g., when the motion of a mold is reversed, or when the forming pressure results in the material being pulled away from a mold. By default, the possibility of contact release is disabled for contact problems. When it is enabled, the detachment of the free surface from the wall occurs if

$$f_n > f_a \tag{17–3}$$

where f_n is the local force density as defined in *Equation 17–1* (p. 381), while f_a is the adhesion force density. The quantity f_a is a physical parameter that depends on the interaction between the deformed melt and the material of the plug. By default, the adhesion force density is set to zero.

Contact release can be enabled for 3D and shell models, and requires that the melt sticks at contact (i.e., .the slip coefficient F_{slip} in *Equation 17–2* (p. 381) equals the penalty coefficient k in *Equation 17–1* (p. 381)).

17.2.3. Velocity- or Force-Driven Mold

In *Penalty Technique for Detecting Contact* (p. 381), we have seen how the contact force is applied on the fluid and how the formulation employed invokes the mold velocity. If the mold velocity is imposed, this data is directly used for calculating the force at the contact. When the mold motion is controlled via a force, you must solve the classical momentum equation for the solid mold given below:

$$F_m + F_f = M \underline{a} \tag{17–4}$$

where,

F_m	=	force applied on the mold of mass M
F_f	=	resistance from the fluid that undergoes the deformation
$\underline{a}$	=	acceleration undergone by the mold
M	=	total mass of the moving part (i.e., the mold and moving components attached to it)

F_m and M are user data. The mold velocity results from the time integration of the acceleration, and is subjected to an initial velocity condition. The direction of displacement is along the direction of the force, so the force F_m must be nonzero.

F_f is obtained from the integration of the forces given by *Equation 17–1* (p. 381) and *Equation 17–2* (p. 381).

For the sake of stability, it is recommended that the convergence criterion be set to 10^{-4} at a minimum (see *Convergence and Divergence* (p. 575)).

When the mold motion is controlled via a force, it is recommended that you specify the maximum displacement of the mold. This will stop the motion of the mold when appropriate. This displacement is evaluated along the direction of the force. When the maximum displacement is reached, the simulation continues with a fixed mold at that final position. For non-shell models, it is expected that as the domain deformation increases, so too the shear forces increase inside the domain; this in turn increases the fluid forces that act on the mold and oppose its motion, which significantly slows down the motion of the mold. For such cases, you should specify whether the mold displacement is bounded and to what value it is bounded. For shell models, on the other hand, there are no shear forces building up in the domain to slow down the motion, and so the likelihood of unrealistic mold motion is higher. Therefore, you must specify the maximum mold displacement for shell models.

17.2.4. Heat Transfer and Contact (Imposed Temperature)

Whenever contact occurs during a nonisothermal simulation, thermal boundary conditions are usually not the same before and after contact. It is possible to impose thermal boundary conditions accordingly on a boundary segment (or face) for which contact detection has been enabled. Whenever contact is taken into account in the energy equation, usually an imposed flux condition before contact changes to an imposed temperature condition in regions where contact has been detected.

In order to alter thermal boundary conditions after contact, a condition of heat transfer by convection is used. This condition is reduced to a penalty formulation when a large value is selected for the transfer coefficient. When contact has been detected at a given location, the flux is expressed as

$$Q = \alpha \left(T - T_{mold} \right) \qquad \textbf{(17–5)}$$

The parameter α has physical dimensions of a convection coefficient. When α is large, *Equation 17–5* (p. 383) is equivalent to an essential boundary condition in temperature (the temperature of the polymer must be equal to T_{mold}); it is, however, now imposed by means of a penalty formulation. Whereas, when $\alpha = 0$, the mold acts as an insulated boundary. Very large values of α will not make the system excessively stiff, but thin thermal boundary layers can be expected when heat advection dominates the energy equation. A practical way to determine α is to use conductivity of the parison material. If *Equation 17–5* (p. 383) is equivalent to establishing the mold temperature in a layer thickness H, then

$$\alpha = \frac{k}{H} \qquad \textbf{(17–6)}$$

where k is the heat conductivity of the investigated fluid (polymer or glass), and H can be considered as an equivalent thermal thickness that, in general, is lower than the fluid thickness. Very large values of α mean that the mold temperature must be established in the polymer over a very small distance.

17.2.5. Heat Transfer and Contact (Conjugate Heat Transfer)

For nonisothermal contact-detection simulations that do not make use of shell elements, it is possible to couple the heat transfer calculation in the parison (or sheet) to the heat transfer in the mold, which undergoes a nonuniform temperature field. This capability—referred to as conjugate heat transfer—models the heat exchange between the mold and the parison, the warmer part heating the cooler one and the usual advection/diffusion mechanism controlling heat exchanges in each zone. Note that the calculation of conjugate heat transfer increases the number of variables in the simulation, since ANSYS POLYFLOW needs to compute a temperature distribution in the mold, rather than assuming it is constant as in *Heat Transfer and Contact (Imposed Temperature)* (p. 383). Since the elements and nodes in the parison do not correspond to the elements and nodes in the mold, the calculation of conjugate heat transfer will be based on a non-conforming finite-element technique.

The basic idea is that a temperature will be calculated in the mold at each step, using thermal boundary conditions along the wall of the mold. Along the face of the mold that will contact the parison, boundary conditions will vary depending on whether or not there is contact.

In the absence of contact, any usual type of temperature condition can be imposed (most of the time, a zero flux condition applies). At locations where contact has been detected, the heat flux per unit surface that exits the parison is equal (with an opposite sign) to the heat flux per unit surface that enters the mold. Also, the temperature of the parison must match the temperature of the mold at the point of contact. Neither the flux value nor the temperature has been imposed; only the continuity of both quantities is required.

More generally, it is also possible to model a thermal resistance between the parison and the mold to follow *Equation 17–5* (p. 383). In this case, both T (the parison temperature) and T_{mold} (the mold temperature) are variables of the simulation that vary from point to point, whereas in *Heat Transfer and Contact (Imposed Temperature)* (p. 383), T_{mold} remains constant.

Imposing continuity of the temperature between the parison and the mold is equivalent to choosing a very high value of the thermal conductivity per unit surface, α in *Equation 17–5* (p. 383). A lower value will increase the temperature difference between T and T_{mold}, and when α vanishes the condition is equivalent to insulating both the parison and the mold independently. In this latter case, the parison and the mold are not thermally coupled.

Physical considerations about the equivalent thermal resistance dictate the value of α, and that the comments about the value of α in *Heat Transfer and Contact (Imposed Temperature)* (p. 383) also apply to conjugate heat transfer.

17.2.6. 3D Viscoelastic Blow Molding Simulations

For 2D and 3D viscoelastic flow, the standard differential viscoelastic models can be used for both isothermal and nonisothermal flows. For 3D viscoelastic flow, a multi-mode integral viscoelastic model for blow molding or thermoforming applications can also be used. This method makes use of a membrane element. The fluid constitutive equations are written as follows:

$$\mathbf{T} = \mathbf{T}_1 + \mathbf{T}_2 \tag{17–7}$$

$$\mathbf{T}_1 = \int_0^{\infty} \sum_{i=1}^{N} \left(\frac{\eta_i}{\lambda_i^2} \right) \exp\left(\frac{-s}{\lambda_i} \right) \left[C_t^{-1}(t-s) - \mathbf{I} \right] ds \tag{17–8}$$

$$\mathbf{T} = 2\eta_2 \mathbf{D} \tag{17–9}$$

where η_2 is an additional viscous component.

The numerics for viscoelastic blow molding are based on an extension of the shell element for fluids (described in *Shell Elements for 3D Models* (p. 380)). The algorithm is of the “explicit-implicit" type. At each time step, viscoelastic extra stresses are evaluated explicitly by integrating *Equation 17–8* (p. 384) in time over the current time step.

These stresses are then added to the right-hand side of the momentum equation, where they act as body forces (denoted by F_n in *Equation 17–10* (p. 385), below). Then velocity, thickness, and positions (collectively denoted by the unknown vector X) are evaluated using the following equation:

$$\left[\frac{M}{\Delta t_n} + K \right] X_{n+1} = M \frac{X_n}{\Delta t_n} + K' X_n + F_n \tag{17–10}$$

Equation 17–10 (p. 385) is specific to the Euler implicit time integration scheme (the default and most recommended scheme for blow molding), and M is a mass matrix that is affected mostly by inertia terms. In order to calculate the velocity, thickness, and position at time $n+1$ from the values at time n, the algorithm requires a stiffness matrix K. The optional viscosity η_2 is not sufficient to guarantee good stability of the algorithm, so a "numerical viscosity" called η_3 is introduced. The value of η_3 affects both K and K'.

In *Equation 17–10* (p. 385), the K matrix contains contributions of both η_2 and η_3, whereas K' is affected only by η_3. ANSYS POLYDATA automatically evaluates η_3 as the sum of all individual mode viscosities divided by 30. This empirical value generally produces good and stable results. When time steps are sufficiently small, this numerical viscosity η_3 (which appears with the $n+1$ subscript on the left-hand side of *Equation 17–10* (p. 385), and with the n subscript on the right-hand side) has no influence on the time accuracy of the results. This algorithm is similar to the method described in *Integral Viscoelastic Models* (p. 251) for integral viscoelastic fluids.

Note that the viscosity η_2 that appears in the constitutive model affects the rheological response of the material, whereas η_3 does not in the limit of small time steps.

The procedure for modeling viscoelastic blow molding or thermoforming in ANSYS POLYFLOW is the same as for other 3D problems using the shell element. See *Inputs for 3D Contact Detection* (p. 397) for details.

17.2.7. Calculation of the Parison Thickness

For 2D axisymmetric blow molding models, ANSYS POLYFLOW can compute the parison thickness. The parison being extruded has at least two surfaces. At each point in the domain, ANSYS POLYFLOW computes the distance between each surface and the point. A symmetric calculation is performed, and at a given location or height along the surface, a unique quantity is given. See *Parison Thickness* (p. 565) for details about setting up the parison thickness calculation.

The calculation method is valid for relatively thin objects, with little variation in thickness; it is not appropriate for thick objects.

17.2.8. Calculation of the Extensions

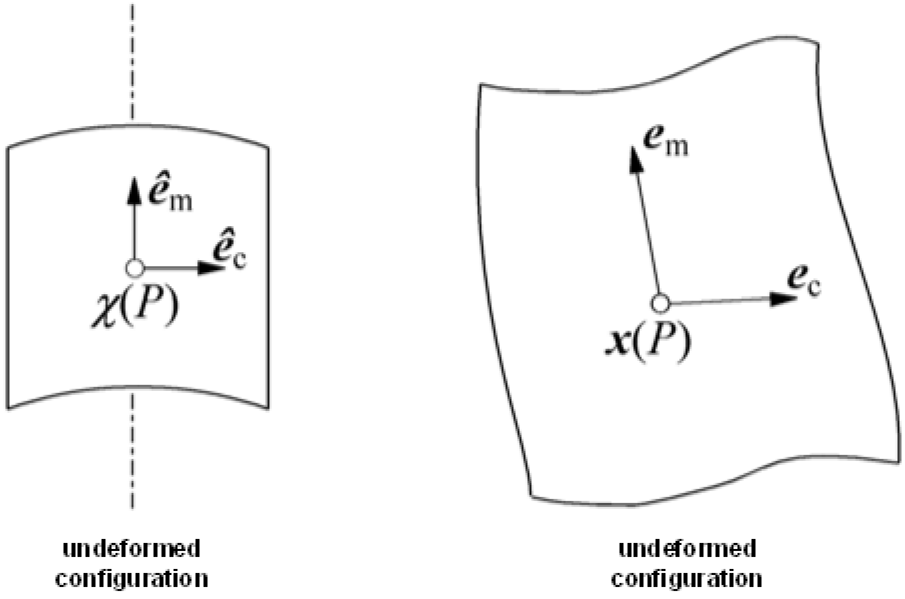
In 3D blow molding processes, and especially during inflation, the parison undergoes large deformations that are basically biaxial or, more generally, multiaxial. The amplitude of the extension components can be very large, and their isotropy or anisotropy can have an influence on the final mechanical properties of the blown product.

For 3D blow molding models solved using shell elements, ANSYS POLYFLOW can compute these extension components. See *Inputs for Computing the Extension Components* (p. 403) for details about setting up the calculation.

17.2.8.1. Evaluation of the Extension Components

In extrusion and blow molding applications, it is easy to evaluate the extension components on the basis of two natural directions considered with respect to the parison axis: the axial and circumferential or hoop directions. The axial component corresponds to the extrusion direction, while the circumferential component is perpendicular to the axial component and tangent to the parison surface, as shown in *Figure 17.2* (p. 386). In thermoforming applications, both sets of unit vectors can lie initially along the main axes.

Figure 17.2 Definition of Axial and Circumferential Extensions



Consider a grid on the undeformed parison. On this grid, a set of unit vectors is defined: $\hat{\mathbf{e}}_a$ and $\hat{\mathbf{e}}_c$ along the axial and circumferential directions, respectively. During inflation, the length and orientation of these vectors are affected, and these changes will record the resulting deformations. In particular, their magnitudes will provide quantitative information along both reference directions.

Consider an arbitrary fluid particle P initially located at χ. At the end of the inflation step, P is located at $\mathbf{x}$. In the vicinity of P, the tensor of deformation gradients $\mathbf{F}$ is given by

$$\mathbf{F} = \frac{\partial \mathbf{x}}{\partial \chi} \tag{17–11}$$

From there, the extension components ($\mathbf{e}_a$ and $\mathbf{e}_c$) are evaluated from

$$\mathbf{e}_a = \mathbf{F} \cdot \hat{\mathbf{e}}_a, \quad \mathbf{e}_c = \mathbf{F} \cdot \hat{\mathbf{e}}_c \tag{17–12}$$

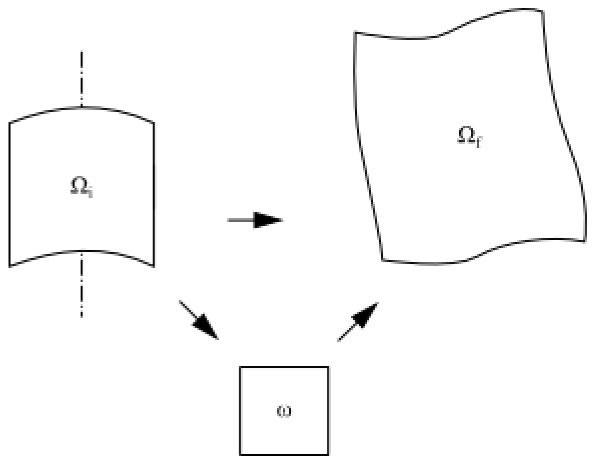
and the magnitude of the extension components is given by the amplitudes $\mathbf{e}_a$ and $\mathbf{e}_c$, respectively.

17.2.8.2. Evaluation of the Area Stretch Ratio

In addition to the extension components, which indicate the possible anisotropy of the deformations, it is also relevant to evaluate the area stretch ratio. A first approximation can be obtained by considering a material surface and calculating the ratio of its area in the deformed configuration to that in the undeformed state. Such a representation is easy, but not continuous.

ANSYS POLYFLOW uses a continuous representation that takes advantage of the finite-element technique and Lagrangian formulation. Internally, the numerical integration of the equations on a finite element requires a geometric transformation between the actual element Ω and its parent element ω, as illustrated in Figure 17.3 (p. 387).

Figure 17.3 Geometric Transformations for Evaluating the Area Stretch Ratio



The transformation is based on a Jacobian J. This Jacobian can be understood as the ratio of infinitesimal areas respectively considered in the actual and parent elements:

$$J = \frac{d\Omega}{d\omega} \tag{17–13}$$

Consider the material element Ω in its initial (undeformed) and final (deformed) configurations, Ω_i and Ω_f, respectively. For these configurations, the respective Jacobians are obtained from

$$J_i = \frac{d\Omega_i}{d\omega}, \quad J_f = \frac{d\Omega_f}{d\omega} \tag{17–14}$$

From Equation 17–14 (p. 387), the area stretch ratio A is given by

$$A = \frac{J_f}{J_i} = \frac{d\Omega_f}{d\Omega_i} \tag{17–15}$$

With a bilinear interpolation, nodal values for A can be evaluated in a post-calculation based on the least mean squares technique. Hence, a continuous representation is obtained for A.

17.2.9. Calculation of the Mass of the Blown Product

For 3D blow molding models solved using shell elements, ANSYS POLYFLOW can also compute the mass of the blown product. The fluid parison or sheet is characterized by a density ρ (specified during the problem definition), a layer thickness h, and a surface area A. The mass of the blown product can be evaluated as

$$m_{blown} = \int_A \rho h dA \quad \textbf{(17–16)}$$

See *Inputs for Computing the Mass of the Blown Product* (p. 404) for details about setting up the calculation.

17.2.10. Calculation of the Volume of the Blown Product

For 3D blow molding models solved using shell elements, ANSYS POLYFLOW can also compute the volume of the blown product. This is actually the volume defined between the parison (or sheet) and an oriented reference plane. The oriented reference plane is defined by

$$Ax + By + Cz + D = 0 \quad \textbf{(17–17)}$$

A point on the volume with coordinates (x_1, y_1, z_1) is considered if

$$Ax_1 + By_1 + Cz_1 + D > 0 \quad \textbf{(17–18)}$$

For example, consider an oriented horizontal plane whose “positive side" is above. Typically, if the blown product is a sphere and the oriented plane is located below the sphere, the algorithm will evaluate the total volume of the sphere. The volume lying between the plane and the lower part of the sphere is actually evaluated twice, once with a positive sign and once with a negative sign, so that the two contributions cancel each other. If the plane cuts through the sphere, the volume of the part of the sphere above the plane will be evaluated. See *Inputs for Computing the Volume of the Blown Product* (p. 405) for details about setting up the calculation.

17.2.11. Calculation of the Permeability of the Blown Product

For 3D blow molding models solved using shell elements, ANSYS POLYFLOW can also compute the permeability of the blown product. This calculation is useful in the packaging of pharmaceuticals, where the moisture barrier is an essential property. The permeability is known to be inversely proportional to the local thickness h. Given the permeability coefficient π, the permeability per unit area can be evaluated as

$$p = \frac{\pi}{h} \quad \textbf{(17–19)}$$

If the material involves several overlapping layers, the permeability per unit area is computed from

$$\frac{1}{p} = \sum_i \frac{1}{p_i} = \sum_i \frac{h_i}{\pi_i} \tag{17–20}$$

where h_i is the thickness of the ith layer, and π_i is the corresponding permeability coefficient.

The total permeability for the blown product is computed from

$$P = \int_A p dA \tag{17–21}$$

The permeability coefficient denotes a given amount of water passing through a sheet of material with a given thickness and area, during a given time. Typical values for π are of the order of 0.1 to 1 $\mathrm{g\text{-}mm/day/m^2}$ or 10^{-7} to 10^{-6} $\mathrm{g\text{-}mm/day/mm^2}$. See *Inputs for Computing the Permeability of the Blown Product* (p. 407) for details about setting up the calculation.

17.2.12. Time Dependence and Contact Handling

ANSYS POLYFLOW is an implicit code that uses a predictor-corrector scheme to solve time-dependent problems. The implicit nature of the time-marching scheme has tremendous advantages for process simulations, because most of the process is in a nearly steady-state regime. Blow molding is not an exception to this rule: pure dynamic effects (like inertia) are usually small and the dominant event that occurs over time is the contact.

In the predictor-corrector scheme, ANSYS POLYFLOW first evaluates all variables (position, velocity, temperature, etc.) at the next time step with an explicit extrapolation scheme. In view of the explicit nature of this prediction step, contact is not detected; this would require a large computational effort. ANSYS POLYFLOW then evaluates corrections to the predicted values, using an implicit iterative scheme (i.e., solving a nonlinear system). The implicit step builds all finite-element matrices and hence takes contact into account.

The deviation between predicted values and corrected values is used as an error estimate, and determines the next time step. Occurrence of contact cannot be predicted during the prediction step, since ANSYS POLYFLOW does not anticipate the contact. This means that it is not possible to use the difference between the prediction and correction steps to control the time step for a contact detection problem. More precisely, the error evaluation cannot be performed on all variables, and in particular cannot be performed on the velocity field.

Whenever contact occurs, the position of the parison remains unchanged regardless of the inflation pressure. There is a discontinuity in the time derivative of the position, but not in the position of the parison itself. The discontinuity for position variables is rather mild. For the velocity field, however, the discontinuity is much stronger because the value of the velocity suddenly jumps from a nonzero value before contact to a zero value after contact (for a zero wall velocity).

The discontinuity in the velocity field at the occurrence of contact makes use of a predictor-corrector scheme inappropriate. Since the Reynolds number for blow molding simulations is usually small, inertia terms can almost always be neglected with respect to diffusion (viscosity) terms, and the velocity field after contact is influenced very little by the history of the momentum.

Therefore, adjustment of the next value of the time step based on a predictor-corrector scheme should be avoided for all simulations that include contact. When you set up a contact detection problem, ANSYS

POLYDATA will automatically disable the prediction for the velocity field, as described in *Inputs for 2D Contact Detection* (p. 392). The option for disabling the prediction is in the **Numerical parameters** menu.

17.2.13. Residual Deformations and Stresses

ANSYS POLYFLOW can also compute the residual deformations and stresses due to a nonuniform temperature distribution for the domain. Consider a fluid parison or a sheet, which has a nonuniform temperature distribution after a cooling phase. During the cooling phase, it is assumed that the stresses are relaxed, and thus, the material is stress free at this stage. However, the temperature distribution will lead to a deformation/stress at ambient temperature.

The model is based on the linearity of the elasticity equations (*Equation 25–1* (p. 503) – *Equation 25–3* (p. 504)), for which the superposition principle can be applied. Starting from the assumption that the domain is stress free for a given nonuniform temperature field, stresses and deformations that would exist at ambient temperature are calculated. With such an assumption, the results of the computation are qualitative, since the real residual stresses and deformations are time dependent, and also depend upon the temperature history. But, for a first approximation, this approach allows you to ascertain which parts of the product are in traction, compression, or shear, as well as how the product is deformed.

The computation for the residual deformations and stresses is not only applicable for blow molding or thermoforming simulations, but can also be used for applications like extrusion.

The thermoelastic problem requires boundary conditions for the displacements. The boundary conditions used for the fluid-structure interaction remains valid, except for the interface between the fluid and solid. All other boundary conditions (such as normal/tangential displacement/force imposed, or plane of symmetry) can be applied. See *Elasticity Boundary Conditions* (p. 505) for more details.

In order to compute the residual stresses and deformations, there are several steps:

1. The first step is to simulate the blow molding, thermoforming, or extrusion. This first step may or may not be isothermal.
2. The second step is only required if the first step is performed with shell elements. With shell elements, there is only one temperature unknown across the thickness. Since there is no temperature gradient across this thickness, it is not possible to simulate the cooling phase. Thus, you have to convert the shell element geometry and the related results into a real 3D geometry. See *Converting a Shell Mesh and Results* (p. 168) for details on how to convert the mesh and results.
3. The third step is a cooling phase, for which you need to define a transient heat transfer task. If the previous simulation already took into account the temperature, you have to restart from this result, in order to initialize the temperature field. Otherwise, you have to initialize it with a constant value (e.g., the **average temperature** item). The transient task may be interrupted when a criterion like the minimum of temperature reaches a limit. See *Interrupting Evolution* (p. 527) for details on interrupting the transient task.
4. Finally, the fourth step is the computation of the residual stresses and deformations.

17.2.14. Temperature Programming

At the end of a thermoforming process, we can observe nonuniformities in the thickness distribution: in some areas, the sheet can be too thick, while in other areas, it is too thin. Instead of increasing the initial thickness of the sheet in order to obtain a minimum thickness everywhere, it is possible to play with initial temperature distribution. We can increase the heat on a part of the initial sheet that eventually leads to an area with an excess of matter. This makes it possible to improve the thickness distribution in the final sheet product without increasing the mass of the sheet.

Of course, the improvement is limited. If the viscosity law or viscoelastic model is not temperature dependent, no improvement can be expected. Moreover, if the final thickness is too far from the desired thickness, the temperature prescribed by the postprocessor could become unrealistic.

To perform optimization, we will adjust temperature at each node of the sheet in the following way: nodes that are too thick (compared to desired final thickness) will be heated (initial temperature will be increased), while nodes that are too thin will be cooled.

For each node of the sheet, its new initial temperature is obtained as follows:

$$\text{new } T_0 = \text{old } T_0 + \alpha \left(T^* - \text{old } T_0 \right) \quad \textbf{(17–22)}$$

Where α is the under-relaxation factor and T^* is the temperature solution of the following equation:

$$H\left(T^*\right) = H\,(\text{old } T_0)\,\frac{hd}{hf} \quad \textbf{(17–23)}$$

where hf is the thickness at the current node at the end of the simulation, hd is the thickness required, and $H\,(T)$ is the dependency law of the sheet as a function of temperature. If the sheet is constituted of several layers with different temperature dependency laws, the $H\,(t)$ law is a weighted average function of $H_i\,(t)$ law of each layer i:

$$H\,(t) = \sum_i H_i\,(t)\,.\frac{h_i}{hf} \quad \textbf{(17–24)}$$

Where h_i is the thickness of layer i and $H_i\,(t)$, the temperature dependency law of layer i.

The implemented technique requires several runs of ANSYS POLYFLOW. At the end of each simulation, the postprocessor evaluates a new initial temperature distribution based on initial temperature, final thickness distribution, and desired thickness distribution. This new initial temperature distribution will be imposed for the next simulation. A few optimization steps are necessary to get a better solution. In order to avoid excessive variations in temperature, we have to prescribe minimum and maximum allowed temperatures. Moreover, as in a real process, we cannot impose a specific temperature for each point of the sheet, therefore, we will divide the initial sheet into nonoverlapping rectangles. In each rectangle, the new initial temperature will be the minimum of the temperatures of the nodes laying in the rectangle and evaluated by *Equation 17–22* (p. 391). See *Inputs for Temperature Programming* (p. 412) for details about setting up temperature programming calculations.

17.3. Setting Up a Contact Problem

This section contains instructions for setting up a contact detection problem. See also *Computing Derived Quantities in Contact Problems* (p. 403) for information about computing derived quantities related to contact detection.

Note that you have access to POLYFLOW project templates, which are Workbench project files that you can modify in order to quickly and easily set up your own problem. These templates include blow molding and thermoforming problems. See POLYFLOW Project Templates in the POLYFLOW in Workbench User's Guide for further details.

17.3.1. Inputs for 2D Contact Detection

The inputs for a 2D model that are directly related to the contact detection are provided in this section. See *Inputs for 3D Contact Detection* (p. 397) for information about setting up a 3D contact detection problem. You can set up the rest of the problem (material properties, other boundary conditions, etc.) as usual. See *Remeshing* (p. 343) for information about remeshing.

1. Create a task of the time-dependent type.

   Create a new task

2. Define the mold(s).

   Define molds

   a. Create a mold.

      Create a new mold

   b. Specify the type of mold. There are three options:

      - To model an adiabatic mold, select **Adiabatic mold**:

        Adiabatic mold

        If you select this option, there will be no heat exchange between the mold and the fluid in contact.

      - To model a mold that remains at a constant specified temperature throughout, select **Mold with constant and uniform temperature**:

        Mold with constant and uniform temperature

        If you select this option, the heat transfer between the mold and the fluid in contact will be taken into account in the calculation. The heat transfer is governed by a heat transfer coefficient and a mold temperature.

      - To model conjugate heat transfer between the mold and the fluid, select **Mold with temperature calculation**:

        Mold with temperature calculation

        If you select this option, ANSYS POLYDATA will disable the option to use shell elements for the parison. This is because shell elements cannot be used in a simulation that involves conjugate heat transfer between the parison and the mold.

   c. When prompted, specify a name for the mold.

   d. Specify the solid region (i.e., subdomain or subdomains) that represents the mold.

      Domain of the mold

e. If you selected the **Mold with temperature calculation** option, define the density, thermal conductivity, heat capacity per unit mass, average temperature, and heat source per unit volume.

Material data

See *Problem Setup* (p. 286) for details about specifying these thermal properties. The average temperature that you specify applies only to this particular mold, and serves as an initial temperature condition.

f. If you selected the **Mold with temperature calculation** option, define the thermal boundary conditions for it.

Thermal boundary conditions

See *Problem Setup* (p. 286) for details about specifying thermal boundary conditions. Note that the wall along which contact has been defined will be considered insulated until contact occurs.

g. Specify which boundary represents the part of the mold that comes into contact with the fluid. When a point of a free surface enters a solid mold, ANSYS POLYFLOW computes the penetration length. This length is the distance between the point and the boundary of the solid mold or contact domain. Usually, it is efficient to restrict the contact domain boundary to the part that the free surface actually touches. Hence, it is recommended that the boundary of the mold be defined by different boundary sets. In this step, you choose which part must be taken into account to compute the penetration length.

Contact conditions

i. Select the boundary that is in contact with the free surface and click **Modify**.

ii. Select **Contact** in the resulting menu and then select **Upper level menu**.

iii. Repeat if there are other boundaries that contact the free surface.

iv. Select **Upper level menu** again to continue the mold definition.

h. Specify the motion of the mold, if required.

Mold motion

When you select **Mold motion**, the selected attributes are displayed.

i. Specify the mold motion, if required:

Modify motion type: fixed mold

A. Enter a **New value** in the **Specify the type of mold motion** dialog box that opens to indicate the motion type of the mold:

- Enter `0` to fix the mold (i.e., no velocity or force imposed).
- Enter `1` to impose a translation velocity.
- Enter `2` to impose a translation force.

B. Click **OK** to close the **Specify the type of mold motion** dialog box.

ii. If you opted to impose a translation velocity on the mold, define the velocity.

A. Select **Modify translation velocity**:

Modify translation velocity

When prompted, enter the value of the translation velocity in each coordinate direction.

B. Select **Modify angular velocity** to specify the angular velocity for the solid mold:

Modify angular velocity

This option is available only for 2.5D axisymmetric problems. The rotation axis is always the Z-axis.

iii. If you opted to impose a translation force on the mold, define the settings.

A. Select **Modify translation force**:

Modify translation force

When prompted, enter the value of the translation force in each coordinate direction.

B. Select **Modify initial velocity** to specify the initial velocity, if required:

Modify initial velocity

When prompted, enter the value of the initial velocity in each coordinate direction.

C. Specify the mass of the mold:

Modify mass of mold

This value corresponds to the total mass of the mold and of the moving part connected to the mold.

D. Specify the maximum mold displacement:

Modify max displacement

This value corresponds to the maximum admissible displacement along the direction specified by the force. The mold motion will continue until either the maximum displacement is reached, the specified time duration is reached, or the solution diverges.

For shell models, the maximum displacement must be bounded, and hence this quantity is uninitialized when you open the menu for the first time. Subsequently, the maximum displacement can be modified.

For models other than shell models, you may decide whether the mold displacement must be bounded and to what value.

   i. Select **Upper level menu** repeatedly to return to the **Define molds** menu, where you can repeat the steps above to define another mold, if necessary.

   j. Select **Upper level menu** to return to the task menu where you will continue the problem setup.

3. Create a sub-task of the appropriate type for the fluid:

   Generalized Newtonian isothermal flow problem

   Generalized Newtonian non-isothermal flow problem

   Differential viscoelastic isothermal flow problem

4. Specify the region where the sub-task applies.

   Domain of the sub-task

5. Specify which boundary is the free surface.

   a. In the **Flow boundary conditions** menu, select the boundary set that represents the free surface and click **Modify**.

      Flow boundary conditions

   b. Select **Free surface** as the boundary type.

      Free surface

6. Define the free surface boundary conditions, following the procedure outlined in *General Procedure* (p. 320).

7. Specify the contact detection problem.

   Contact (Blow mold.)

   a. Create a new contact problem.

      Create a new contact problem

   b. Specify where the free surface will contact the mold.

      Select a contact wall

      i. Select the appropriate contact wall (i.e., the one you previously defined in the **Define molds** menu).

      ii. Click **Select**.

   c. If you defined the mold as a **Mold with constant and uniform temperature** or a **Mold with temperature calculation**, specify the parameters for the heat flux across the contact boundary:

      - If the mold is at a constant temperature, specify the values for α and T_{mold} in *Equation 17–5* (p. 383).

Modify alpha

Modify T_mold

T_{mold} will be the constant temperature of the mold in this case.

- If you select the **Mold with temperature calculation** option (i.e., if you are modeling conjugate heat transfer, described in *Heat Transfer and Contact (Conjugate Heat Transfer)* (p. 383)), specify the value for α. You will not specify T_{mold} because the temperature of the mold is not constant; it is computed by ANSYS POLYFLOW during the simulation.

d. Modify the slip coefficient (F_{slip} in *Equation 17–2* (p. 381)), if necessary.

Modify slipping coefficient

The default value is 10^9. If the slip coefficient and the penalty coefficient have the same value, then it is assumed that the fluid sticks to the mold when it comes into contact. Full slippage at the contact boundary is assumed if the slip coefficient is zero. Actual cases involving partial slippage will require a slip coefficient that typically ranges from 10^2 to 10^6, depending on the physics involved as well as on the system of units being used.

e. Modify the penalty coefficient (k in *Equation 17–1* (p. 381)), if necessary.

Modify penalty coefficient

The default value of 10^9 is acceptable for most cases.

f. Modify the penetration accuracy, if necessary.

Modify penetration accuracy

If the penetration of a point into the mold is greater than the penetration accuracy, the time step will be rejected. The calculation will then be restarted from the previous time step with a smaller time-step increment. The default value is set according to the dimensions of the mesh, and you will generally not need to modify it.

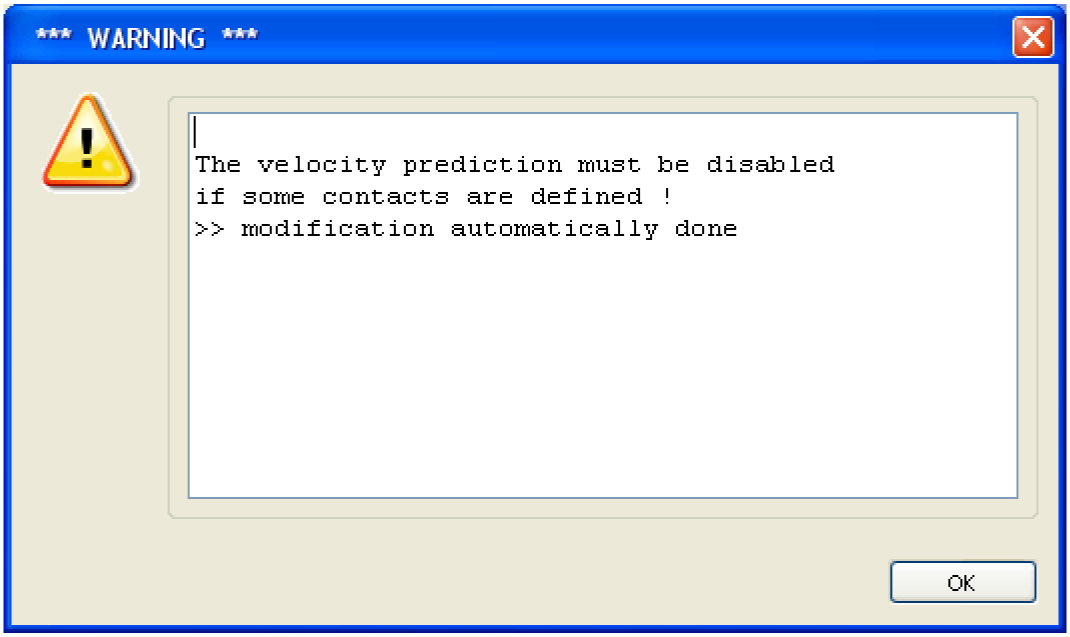
g. Specify the element dilatation, if necessary.

Modify element dilatation

This option allows for expansion of each finite element of the mold in order to help the contact detection algorithm. Some numerical errors can cause a point to “miss" the mold. A typical case can occur when a point of the fluid domain is supposed to come into contact with a mold at the location of the symmetry line of the mold. A small expansion of the finite elements fixes the problem. The default value is set according to the dimensions of the mesh, and you will generally not need to modify it.

h. Select **Upper level menu** to return to the **Define contacts** menu, where you can repeat the steps above to define another contact wall for the free surface.

i. Select **Upper level menu** two times to complete the free surface and contact problem definition and return to the **Flow boundary conditions** panel. ANSYS POLYDATA will warn you that the velocity prediction has been disabled. In contact detection problems, abrupt changes in the velocity field occur at the contact points between the fluid parison and the solid mold. These discontinuities prevent the prediction scheme from working properly, so it is automatically disabled for you. Click **OK** to continue.



```
The velocity prediction must be disabled
if some contacts are defined !
>> modification automatically done
```

8. Define the remeshing procedure for the sub-task (after you finish setting all boundary conditions and return to the sub-task menu one level above the **Flow boundary conditions** menu).

   Global remeshing

   See *Remeshing* (p. 343) for details.

9. Define the numerical parameters for the time-dependent task (one level above the sub-task menu).

   Numerical parameters

   See *User Inputs for Time-Dependent Problems* (p. 544) for details about the inputs for time-dependent calculations. If you select **Modify the transient iterative parameters**, you will see that the velocity field prediction has been disabled for you, as mentioned above. You should *not* reenable the velocity prediction for a contact detection problem.

17.3.2. Inputs for 3D Contact Detection

The inputs for a 3D model that are directly related to the contact detection are provided in this section. (See *Inputs for 2D Contact Detection* (p. 392) for information about setting up 2D contact detection problems.) You will need to set up the rest of the problem (material properties, other boundary conditions, etc.) as usual. See also *Generating a Mesh with Shell Elements* (p. 401) for information about generating a mesh with shell elements.

1. Create a time-dependent task for a shell geometry.

   Create a new task

2. Define the mold(s).

Define molds

a. Create a mold.

   Create a new mold

b. Specify the type of mold. There are two options (note that the **Mold with temperature calculation** option is not available because it cannot be used with shell elements):

   - To model an adiabatic mold, select **Adiabatic mold**:

     Adiabatic mold

     If you select this option, there will be no heat exchange between the mold and the fluid in contact.

   - To model a mold that remains at a constant specified temperature throughout, select **Mold with constant and uniform temperature**:

     Mold with constant and uniform temperature

     If you select this option, the heat transfer between the mold and the fluid in contact will be taken into account in the calculation. The heat transfer is governed by a heat transfer coefficient and a mold temperature (or temperatures, if you assign different temperatures on different parts of the mold).

c. When prompted, specify a name for the mold.

d. Specify the solid region that represents the mold.

   Domain of the mold

e. Specify the boundary that comes into contact with the fluid (i.e., the contact wall), the translation velocity or force, and other mold parameters following the instructions provided for 2D problems in *Inputs for 2D Contact Detection* (p. 392) (step 2).

f. If necessary, return to the **Define molds** menu and repeat the steps above to define another mold.

3. Create a sub-task of the appropriate type for the fluid:

   Shell model: Gen Newtonian isothermal

   Shell model: Gen Newtonian non-isothermal

   Shell model: Viscoelastic isothermal

   Shell model: Viscoelastic non-isothermal

   For viscoelastic contact detection problems, you can use only the integral viscoelastic models. It is not possible to use a differential viscoelastic model with 3D contact detection. See *Integral Viscoelastic Models* (p. 251) for details about modeling integral viscoelastic flow.

4. Specify the region where the shell sub-task applies.

   Domain of the sub-task

   First you will be prompted to select the subdomains on whose boundaries the shell domain lies. After you choose the appropriate subdomains, you will be prompted next to specify which of the boundaries define the shell domain.

5. Define the flow boundary conditions on the borders of the shell domain, as for any 3D flow problem. Each border is identified as the intersection between the shell domain and an adjacent boundary.

   Flow boundary conditions

   See *Boundary Conditions* (p. 173) for information about setting boundary conditions.

   If you want to impose an inflation pressure, turn on the **Inflation pressure imposed** option in the **Flow boundary conditions** panel. ANSYS POLYDATA will then prompt you to enter the value of the inflation pressure.

6. Define the fluid layer(s) representing the polymer in the mold.

   Define layers

   For the shell model, define one or more layers, each having its own material properties and initial thickness (which can be defined as a constant, a linear function of X, Y, Z coordinates, or a multi-ramp function of X, Y, or Z, or mapped using a CSV file). Layers can overlap (partly or entirely), but they are not required to do so.

   The inputs for a fluid layer are the same as for a film layer in a film casting problem, with the addition of the permeability coefficient. See *General Procedure* (p. 421) for details.

7. Specify the contact detection problem.

   Define contacts

   a. Specify where the fluid will contact the mold.

      Select a contact wall

      i. Select the appropriate contact wall (i.e., the one you previously defined in the **Define molds** menu).
      ii. Click **Select**.

   b. If you defined the mold as a **Mold with constant and uniform temperature**, specify the values for α and T_{mold} in *Equation 17–5* (p. 383).

      Modify alpha

      Modify T_mold

      T_{mold} will be the constant temperature of the mold in this case.

c. If necessary, modify the slip coefficient, penalty coefficient, penetration accuracy, and element dilatation as described in step 7. of *Inputs for 2D Contact Detection* (p. 392).

d. Enable contact release and define the adhesion force density (f_a in *Equation 17–3* (p. 382)), if necessary.

Modify adhesion force density

Contact release is disabled by default, as indicated by the description of the adhesion force density as being **undefined** in the **Modify adhesion force density** menu item. When you click this menu item, ANSYS POLYDATA will ask you to confirm the enabling of contact release and then allow you to specify the **New value** for the adhesion force density. If you do not have test results or specifications to guide your definition of the adhesion force density, it is reasonable to start by replacing the default value of 0 with a small value in your unit system (e.g., 10–100 Pa).

When you enable contact release, you must make sure that the penalty and slip coefficients you defined in the previous step are set to the same value, so that slip at contact is not permitted. Also, it is recommended that you set the convergence test value to 10^{-4} when you define the **Numerical parameters** in step 9. (see *Convergence and Divergence* (p. 575) for details).

e. Specify the amplitude of the volume generation.

Modify amplitude of volume generation

2D molds (defined by a surface only) need to be extended to a volume to allow for contact detection. 2D molds are extended to 3D in the local normal direction, by a constant amount that is precisely equal to the specified amplitude. This value must be entered in the unit of length used for the mesh.

f. Specify the direction for the volume generation.

Specify mold side / cavity side

When you select this option, the graphic display window in ANSYS POLYDATA will be updated to include an arrow. When extending a 2D mold to 3D, you need to specify whether the extension of the geometry occurs in the direction of the arrow (click **Yes**) or in the opposite direction (click **No**).

8. Define the remeshing procedure for the sub-task (after you return to the sub-task menu one level above the **Define contacts** menu).

Global remeshing

See *Remeshing* (p. 343) for details. The Lagrangian method is the only one available for 3D blow molding simulations with shell elements.

9. If you want to compute the extension components, create a postprocessor sub-task for it, as described in *Inputs for Computing the Extension Components* (p. 403).

10. Define the numerical parameters for the time-dependent task (one level above the sub-task menu).

Numerical parameters

See *User Inputs for Time-Dependent Problems* (p. 544) for details about the inputs for time-dependent calculations. If you select **Modify the transient iterative parameters**, you will see that the velocity field prediction has been disabled for you, as mentioned in *Inputs for 2D Contact Detection* (p. 392). You should *not* reenable the velocity prediction for a contact detection problem.

17.3.3. Defining the Thickness Interpolation for Shell Elements

For blow molding and thermoforming simulations that use the shell element, only the linear interpolation scheme is available for the temperature, velocity, and position variables. (This limitation is related to the contact detection algorithm.) The linear interpolation scheme is enabled by default for the thickness variable, but, since a Lagrangian representation is used for the thickness variable, each individual mesh element is handled as a material element.

Hence, from the mathematical point of view, with a Lagrangian representation the mass equation for a shell no longer involves any space derivative of the thickness variable. It is therefore possible to use a discontinuous, constant interpolation on an element basis for the thickness variable.

On a reasonably dense finite-element mesh, the selection of a discontinuous interpolation for the thickness does not reduce the number of unknowns. However, it allows ANSYS POLYFLOW to solve the mass equation at the level of the individual element, since the discontinuous interpolation does not require any inter-element communication. This can lead to an appreciable reduction of the computation time, typically by about 10 to 25%. Furthermore, reinforcing mass conservation at the level of the individual element may also improve the quality of the results.

Note

This interpolation is available only when both the mold and the fluid domains are described as surfaces in the geometric space.

17.3.3.1. Controlling the Thickness Interpolation

For both Newtonian and viscoelastic shell elements, you can change from the default linear (continuous) interpolation scheme for the thickness to a discontinuous scheme by selecting **Constant Thickness** in the **Interpolation** menu for the shell sub-task.

Interpolation

When a multilayer system is defined, a thickness field is defined for each individual layer, but all thickness variables use the same interpolation (either linear continuous or constant discontinuous).

17.3.4. Generating a Mesh with Shell Elements

There are three ways to prepare a mesh with shell elements:

- Generate a standard 3D volumetric mesh, and identify the membrane that represent the shell domain as a distinct boundary set. In ANSYS POLYDATA, you will select this boundary set as the shell domain (as described in *Inputs for 3D Contact Detection* (p. 397)). The rest of the volumetric mesh will be ignored.

  If the parison (or sheet) is volumetric (although only one boundary set is used, as described above), the mold must also be volumetric. If the parison (or sheet) is a shell, then the mold must also be a shell (which will be extended into 3D, as described in *Inputs for 2D Contact Detection* (p. 392)). You cannot mix 2D (shell) and 3D (volumetric) parisons/sheets and molds.

- Generate a mesh consisting of 2D elements in 3D space, representing the parison (or sheet) and the mold. In ANSYS POLYDATA, you will select the subdomain representing the parison (or sheet) as the shell domain, and the subdomain representing the mold as the mold domain. In most cases, this method will be faster and easier than the previous method.
- Generate a mesh consisting of 2D elements in 3D space for the mold. Then generate a mesh consisting of 2D elements in the 2D plane, solve a 2D axisymmetric extrusion simulation, create the circular shell mesh for the membrane from the results, and merge this mesh with the mesh for the mold. The procedure is as follows:

1. Set up the 2D axisymmetric extrusion simulation.
2. Create a postprocessor sub-task to calculate the parison thickness (as described in *Parison Thickness* (p. 565)).
3. In the **Parison thickness** menu, select **Create transfer files for shell blow molding** and define the parameters for the shell mesh generation.

   Create transfer files for shell blow molding

   The parameters you can modify are as follows:

   – Mesh filename: To specify the name of the file to which the shell mesh will be written at the end of the ANSYS POLYFLOW simulation, select **Modify MSH filename**.

   – CSV filename: Select **Modify CSV filename** to specify the name of the comma-separated variable (CSV) file to which the thickness distribution and other results will be written at the end of the ANSYS POLYFLOW simulation.

   – Number of mesh elements in the axial direction: By default, the number of elements in the axial direction is the same as in the current 2D mesh. Select **Modify no. of elements in axial direction** to modify this number.

   – Number of mesh elements in the tangential direction: Select **Modify no. of elements in azimuthal direction** to specify the number of elements in the tangential (azimuthal) direction.

   – Sector angle: Select **Modify angle of generation** to specify which sector of a circle is to be generated for the membrane (e.g., 90°, 180°, or a full 360° circle).

   – Element type: Select **Switch to triangular elements for shell mesh**to specify triangular elements. Select **Switch to quadrilateral elements for shell mesh** to specify quadrilateral elements.

   – Border for membrane: The creation of the membrane is based on a border, either the internal or the external border of the 2D axisymmetric mesh. Select **Specify generation line** to specify this border. Mesh generation on the basis of the external border is recommended, but if the 2D axisymmetric parison is very thin, with a relatively large radius, the specific choice is not significant.
4. Save the data file, exit from ANSYS POLYDATA, and compute a solution with ANSYS POLYFLOW. The shell mesh file and CSV file will be saved with the names that you specified.
5. In the ANSYS POLYDATA session for your 3D simulation, select **Combine mesh files** to use ANSYS POLYFUSE to combine the shell mesh and the 3D mesh for the mold. See *Combining Meshes with ANSYS POLYFUSE* (p. 151) for information about combining meshes.

6. When you define the fluid layers representing the polymer in the mold (see *Inputs for 3D Contact Detection* (p. 397)), you can specify the initial thickness distribution by reading the data from the CSV file.

Note that ANSYS POLYFLOW currently does not allow the parison/sheet and mold shells to be closed on themselves. That is, they both require a 1D boundary; a closed sphere is not acceptable. So-called waterproof parison/sheet or mold shells are not allowed, and cannot be set up in ANSYS POLYDATA.

17.4. Computing Derived Quantities in Contact Problems

17.4.1. Inputs for Computing the Extension Components

If you are using one of the 3D shell models, you can compute the extension components described in *Calculation of the Extensions* (p. 385). The procedure is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Extension Evaluation** (at the bottom of the list).

   Extension Evaluation

   ANSYS POLYDATA will tell you where the calculation will be performed (the intersection between two subdomains). You can click **OK** in the message panel to continue.

3. Specify which extension-related calculations you want to perform. By default, only the area stretch ratio is computed. To enable scalar or vector evaluation of the extensions, select the appropriate **Enable** menu item. To disable any of the currently-enabled components, select the corresponding **Disable** menu item.

4. Specify the reference direction vector and point for the unit vectors described in *Calculation of the Extensions* (p. 385) ($\hat{\mathbf{e}}_a$, and $\hat{\mathbf{e}}_c$).

   Modify the reference direction

   Modify the reference point

   ANSYS POLYFLOW will use a geometrical construction to determine the two sets of vectors based on the specified reference point and direction.

   The reference point is a point that does *not* belong to the parison or the polymeric sheet in the initial configuration. The vector that links each point of the parison and this point should never become aligned with the parison. The reference direction is a vector along which you want to orient one of the extension unit vectors. The other unit vector will be tangent to the parison and perpendicular to the specified reference direction.

If the parison has an axis of symmetry, specify the reference point on the axis and the reference direction as the axis.

17.4.2. Inputs for Computing the Mass of the Blown Product

If you are using one of the 3D shell models, you can compute the mass of the blown product as described in *Calculation of the Mass of the Blown Product* (p. 388). By default, the calculation will be performed for the entire parison or sheet, but you can restrict the calculation in two ways.

- You can limit it to the portion of the parison or sheet that lies on the positive side of one or more cutting planes.
- You can limit it to the portion that is in contact with a part of the mold or form.

The second option is useful, for example, if you want to evaluate the actual mass of a container, once the scraps have been trimmed. Here, use several different subdomains to define the mold, one of which represents the cavity. Then you can restrict the calculation domain to the region that is in contact with the mold cavity, by selecting the subdomain representing the cavity in the procedure below.

The procedure for setting up the calculation of the mass of the blown product is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Mass of blown product**.

   Mass of blown product

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** in the message panel to continue.

3. Check the density of the material, and modify it if necessary.

   Density of layer [Layer-name]

   (**Layer-name** represents the name you assigned when you defined the layer.)

4. Select **Upper level menu**.

5. Define one or more cutting planes, if desired. Cutting planes can be used to restrict the region of the blown product for which the mass is to be computed. If you want to compute the mass of the entire blown product, then do not define any planes; simply select **Upper level menu** again and continue the definition. If you want to restrict the computation region, follow the instructions provided for the volume calculation in step 3 of *Inputs for Computing the Volume of the Blown Product* (p. 405). Note, however, that cutting planes are optional for the evaluation of the mass of a blown product. In addition, cutting planes can be arbitrarily selected, but must be properly oriented.

6. Next you will have the opportunity to restrict the calculation to the part of the parison or sheet that is actually in contact with a part of the mold or form (as described at the beginning of this section).

If you do not want to restrict the calculation in this way, simply click on **Upper level menu** in the panel. If you want to restrict the calculation, follow these steps:

a. Select the contact domain to which you want to restrict the calculation. By default, it will be preceded by **Ignore contact with**, which indicates that the calculation will not be limited to the portion that is in contact with this domain.

b. Click **Modify**.

c. Select **Restricting contact domain**.

d. Repeat to restrict the calculation based on contact with other subdomains.

e. Click **Upper level menu** to complete the definition.

The calculation will now be limited so that only the mass of the part of the blown product that is in contact with the specified subdomain(s) will be computed.

17.4.3. Inputs for Computing the Volume of the Blown Product

If you are using one of the 3D shell models, you can compute the volume of the blown product as described in *Calculation of the Volume of the Blown Product* (p. 388). The definition for this calculation requires you to specify a reference plane that determines which portion of the volume is to be computed (e.g., above/below the plane or to the left/right of the plane). You can specify several additional planes with different orientations to define a box. Then the calculation will be performed only for the portion of the domain that lies within the box. Note that if you define multiple planes to restrict the calculation of the volume to a box, the additional planes *must* be perpendicular to the reference plane (i.e., to the first plane you defined).

A plane is defined by the equation

$$Ax + By + Cz + D = 0 \tag{17–25}$$

and a point on the volume with coordinates (x_1, y_1, z_1) is considered if

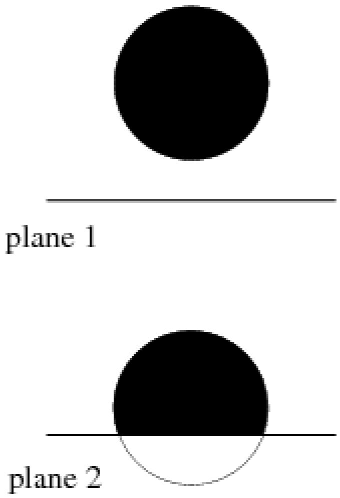
$$Ax_1 + By_1 + Cz_1 + D > 0 \tag{17–26}$$

Important

Note that the signs of the coefficients are very important, because they determine the orientation of the plane. For example, $z - 1 = 0$ and $-z + 1 = 0$ define the same geometric plane, but with different orientations.

Figure 17.4 (p. 406) shows an example in which plane 1 lies below the sphere and plane 2 cuts through it. For plane 1, the entire volume of the sphere will be calculated. For plane 2, only the volume of the upper part of the sphere will be calculated.

Figure 17.4 Reference Planes for the Volume Calculation



If you consider that plane 2 is defined by $z-1=0$, then defining it instead by $-z+1=0$ will result in a calculation for only the bottom part of the sphere.

The procedure for setting up the calculation of the volume of the blown product is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Capacity of blown product**.

   Capacity of blown product

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** in the message panel to continue.

3. Specify one or more planes to define the portion of the parison or sheet for which the volume should be calculated. See the beginning of this section for details about how this works.

   a. Define the reference plane.

      Add a new plane

      When prompted, specify the coefficients for A, B, C, and D.

Important

Remember to pay careful attention to the signs of the coefficients, as noted above.

b. Repeat to define additional planes, if desired.

> **Important**
>
> Note that these additional planes *must* be perpendicular to the reference plane. Also, remember to pay careful attention to the signs of the coefficients, as noted above.

If you want to modify or delete a plane that you have defined, select **Modify a plane** or **Remove a plane**, and then specify the index of the plane to be modified or deleted. (This is the number listed under **plane id** next to the equation displayed at the top of the ANSYS POLYDATA menu.)

c. When you are done defining planes, select **Upper level menu** to continue.

4. Next you will have the opportunity to restrict the calculation to the part of the parison or sheet that is actually in contact with a part of the mold or form (as described at the beginning of *Inputs for Computing the Mass of the Blown Product* (p. 404)).

   If you do not want to restrict the calculation in this way, simply click on **Upper level menu** in the panel. If you want to restrict the calculation, follow the procedure in the last step of *Inputs for Computing the Mass of the Blown Product* (p. 404).

17.4.4. Inputs for Computing the Permeability of the Blown Product

If you are using one of the 3D shell models, you can compute the permeability of the blown product as described in *Calculation of the Permeability of the Blown Product* (p. 388). The procedure is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Permeability of blown product**.

   Permeability of blown product

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** in the message panel to continue.

3. Specify the permeability coefficient of the material.

   Coef. of permeability of layer [Layer-name]

   (**Layer-name** represents the name you assigned when you defined the layer.)

4. Select **Upper level menu**.

5. Define one or more cutting planes, if desired. Cutting planes can be used to restrict the region of the blown product for which the permeability is to be computed. If you want to compute the permeability for the entire blown product, then do not define any planes; simply select **Upper level menu** again and continue the definition. If you want to restrict the computation region, follow the instructions provided for the volume calculation in step 3 of *Inputs for Computing the Volume of the Blown Product* (p. 405). Note, however, that cutting planes are optional for the evaluation of the permeability of a blown product. In addition, cutting planes can be arbitrarily selected, but must be properly oriented.

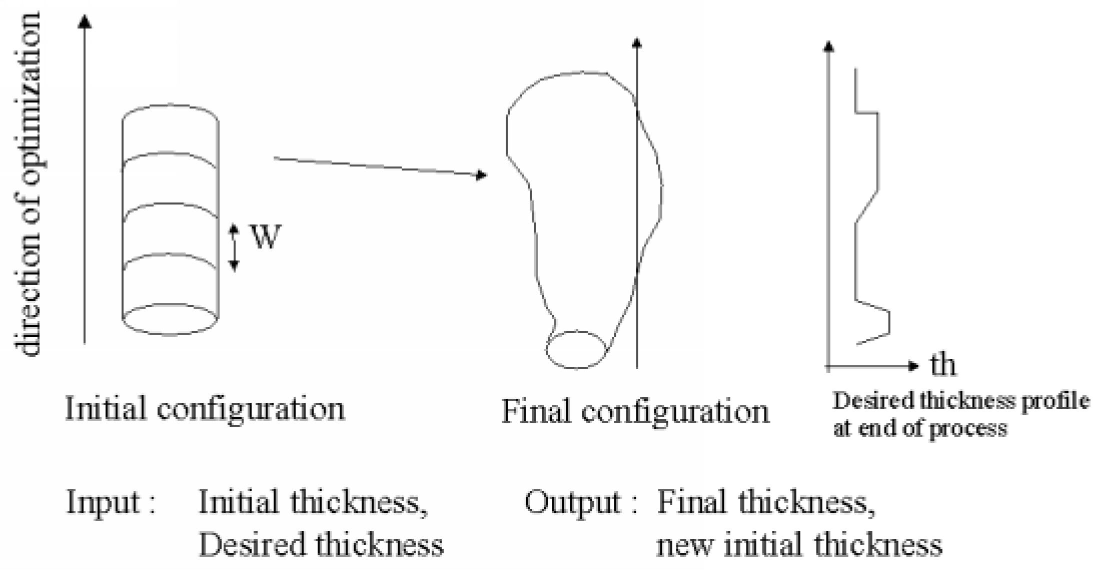
6. Next you will have the opportunity to restrict the calculation to the part of the parison or sheet that is actually in contact with a part of the mold or form (as described at the beginning of *Inputs for Computing the Mass of the Blown Product* (p. 404)).

   If you do not want to restrict the calculation in this way, simply click on **Upper level menu** in the panel. If you want to restrict the calculation, follow the procedure in the last step of *Inputs for Computing the Mass of the Blown Product* (p. 404).

17.4.5. Inputs for Parison Programming

This new postprocessor is available for blow molding and thermoforming processes in 3D shell models. Due to some constraints on thickness distribution of the final product, the thickness distribution of the initial parison, or of the initial sheet of polymer is estimated, which will eventually provide the required thickness distribution. The procedure is iterative. Start with your specified initial thickness distribution and run the simulation. At the end of it, the postprocessor is activated and a file is generated containing a new initial distribution. Then, run a new simulation with this new initial thickness distribution. Repeat this procedure for 5 to 10 optimization steps until the solution is converged. The final thickness obtained will be very close to the requested profile and with a minimum of mass.

Figure 17.5 Optimization Process for Calculating the Thickness


direction of optimization
W
th
Initial configuration
Final configuration
Desired thickness profile
at end of process
Input : Initial thickness,
Desired thickness
Output : Final thickness,
new initial thickness

Define your blow molding simulation with a specified initial thickness distribution. Then create the “parison programming" postprocessor.

1. Create a new sub-task.

   Create a sub-task

a. Select **Postprocessor** as the sub-task type.

   Postprocessor

b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Parison programming**.

   Parison programming

3. Select the layer on which the thickness distribution will be optimized. Select **thickness optimisation DISABLED for layer my_layer**. By default, the optimization for the layer **my_layer** will be disabled. To activate the menu for optimization, select:

   Enable optimisation

4. Specify the direction of optimization. Select **Modify the direction Z of optimisation**.

   The direction mentioned here is the direction of extrusion of parison in a blow molding simulation. In a thermoforming simulation, it is the direction transverse to the extrusion direction of the polymer sheet.

5. Specify the under-relaxation factor α. Select **Modify the under-relaxation factor**.

   As the technique of optimization is iterative, it is mandatory to specify an under-relaxation factor (α) to avoid oscillations in the solution for each step. α ranges between 0 and 1 and has a default value of `0.9`.

6. Specify the width of the stripes. Select **Modify the width of stripes**.

   You have two alternatives:

   - If the width of the stripes is set to zero, then for each nodal value, a new thickness is evaluated, in order to obtain for that node, at the end of blowing simulation, a specific desired thickness.

     $$\text{new}\, ho\,(i) = ho\,(i) + \alpha \frac{hd\,(i) - hf\,(i)}{hf\,(i)} ho\,(i) \qquad \textbf{(17–27)}$$

     where i is the index of the current node of the parison/sheet, $ho\,(i)$ is the initial thickness at that node, $hd\,(i)$ is the requested thickness for that node, and $hf\,(i)$ is its final thickness.

   - If the width of the stripes is set to W (W>0), then you will evaluate one constant initial thickness for each stripe of the parison/sheet, i.e., for each node, you will estimate its new initial thickness (see *Equation 17–27* (p. 409)) and determine in which stripe it is included. The new initial thickness of the stripe S will then be:

     $$\text{new}\, ho\,(S) = \max\left(\{\text{new}\, ho\,(i\,), \ldots \mid \forall\, i \subset S\} \right) \qquad \textbf{(17–28)}$$

7. Specify the time of activation of the postprocessor. Select **Modify the time of activation**.

   The new initial thickness distribution is calculated at the end of the simulation, not in the beginning. It is recommended that you specify the time of activation as $3/4^{th}$ the time of simulation.

8. Specify the required thickness distribution (piecewise linear) in the direction of optimization.

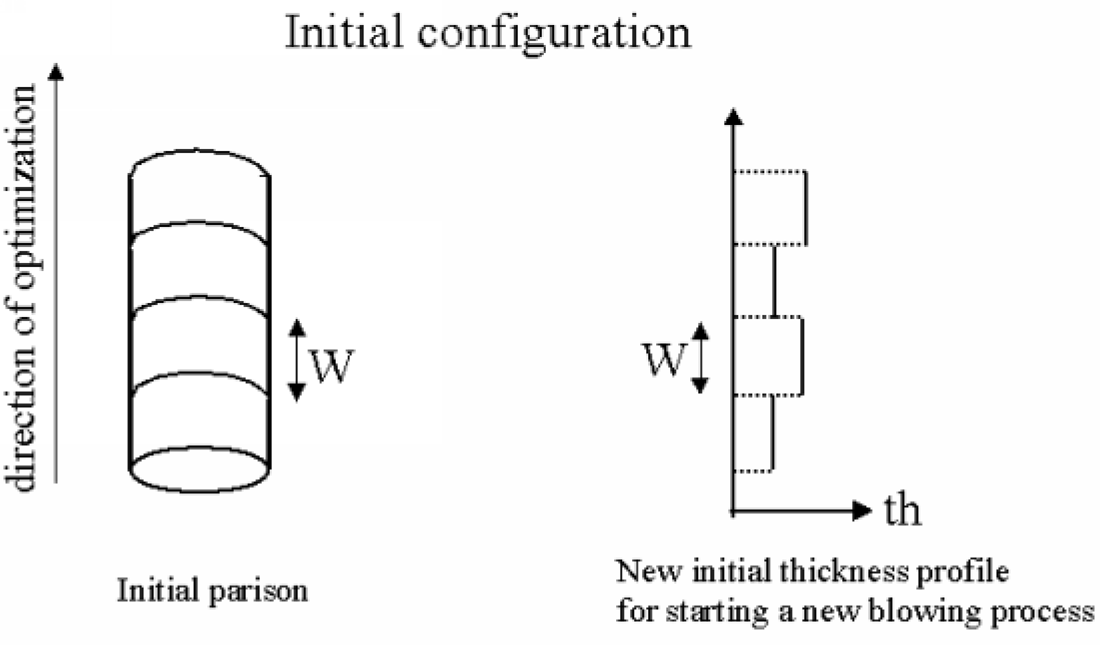
   Modify the list of pairs (Z, h)

9. The new initial thickness distribution is saved in a CSV file. This makes it easier for the next step of optimization, where you have to reread the data file and change the initial thickness in the blow molding sub-task.

   Modify the CSV filename = new_initial_thickness.csv

The result of the postprocessor is shown in *Figure 17.6* (p. 410).

Figure 17.6 Final Result


Initial configuration
direction of optimization
W
Initial parison
W
th
New initial thickness profile
for starting a new blowing process

Several new fields appear in the **Field management** menu at the end of the ANSYS POLYDATA session:

- **THICKNESS**: This field is evaluated by the shell model. It changes as the blowing process advances.
- **THICKNESS INI**: This is the initial thickness used to start the simulation.
- **THICKNESS DES**: This is the desired thickness profile at the end of the blowing process.
- **THICKNESS NEW**: This is the new initial thickness distribution evaluated by the postprocessor. This field is saved in a CSV file that will be used at the next step of optimization, to initialize the thickness field at time t=0.

Note

The last three fields carry a value equal to zero until the simulation reaches the time of activation. For the remaining time of the simulation, the postprocessor determines the values of the fields. Only the last CSV file should be used to start a new optimization step.

17.4.6. Inputs for Residual Stresses and Deformations

If you simulate a nonisothermal problem with a nonuniform temperature field, you can compute the residual stresses and deformations generated by the temperature field at ambient temperature. The procedure for setting up the calculation of the residual deformations and stresses is as follows:

1. Create a new steady task.

   Create a new task

2. Create a sub-task.

   Create a sub-task

   a. Select **Linear Thermo-Elastic stresses and deformations** as the sub-task type.

      Thermo-Elastic stresses and deformations

   b. When prompted, enter a name for this sub-task.

   c. Specify the domain where compute the residual stresses and deformation.

      Domain of the sub-task

   d. Specify the elastic data in the **Material data** menu.

      Material data

      Elastic data

      i. Specify the value of the Young's modulus.

         Modify Eo

      ii. Specify the Poisson's ratio.

         Modify MUo

         ### Important

         Note that while Poisson's ratio is referred to as **MU** in the previous menu item, it is elsewhere denoted by ν (e.g., *Equation 25–2* (p. 503)).

      iii. Specify the lineic dilatation coefficient.

         Modify Bo

         ### Important

         This is also commonly known as the coefficient of linear dilation. Its units are 1/K (where K = degree Kelvin).

iv. Specify the reference temperature for the dilatation. In this case, this reference temperature corresponds to the ambient temperature.

Modify Tref

v. You can choose whether or not you want to take into account the deformation of the mesh in the simulation. Specify your preference by clicking either **Switch to [no update of coordinates]** or **Switch to [update of the coordinates]**.

Switch to [no update of coordinates]

or

Switch to [no update of coordinates]

> **Important**
>
> If the coordinates are updated, the problem becomes nonlinear and more than one iteration is required to reach convergence. In addition to this, an evolution may be defined on the lineic dilatation coefficient, in order to address nonlinearities originating from the geometry update.

e. Specify the boundary conditions for the displacement. See *Elasticity Boundary Conditions* (p. 505) for more details about the boundary conditions for an elastic problem.

Displacement boundary conditions

3. Specify the numerical parameters.

Numerical parameters

a. Select the **Start from an old result file** menu item to initialize the temperature field.

Start from an old result file

b. When prompted, enter the name of the old result file.

17.4.7. Inputs for Temperature Programming

The general process for temperature programming is to first define your thermoforming simulation with a specified initial temperature distribution. Then you will create the temperature programming postprocessor.

1. Create a new sub-task.

Create a sub-task

a. Select **Postprocessor** as the sub-task type.

Postprocessor

b. When prompted, enter the name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Temperature Programming**.

   Temperature Programming

3. In the **Temperature programming for thermoforming** menu, specify the following parameters:

   a. Specify the initial sheet position.

      Modify the initial position of the sheet

      We assume that the initial sheet is parallel to two axes of the system of reference. The initial sheet can be in a plane parallel to the X and Y axes, to the X and Z axes, or to the Y and Z axes. If the initial sheet position is not parallel to two axes, you must change the system of reference (through rotations).

   b. Specify the under-relaxation factor α.

      Modify the under-relaxation factor

      As the technique of optimization is iterative, you must specify an under-relaxation factor (α) to avoid oscillations between successive solutions. The value for α ranges between 0 and 1 and has the default value of 0.9.

   c. Specify the size of the rectangles.

      Modify the size of the rectangles

      The initial sheet will be divided in rectangles. The postprocessor will evaluate a new single initial temperature per rectangle. This option allows you to specify the width and length of the rectangles. For example, if the initial sheet is oriented in the XY plane, then width is along the X axis and length along the Y axis. If the width and length are set to zero, a new initial temperature will be evaluated at each vertex of the initial sheet.

   d. Specify the time of activation of the postprocessor.

      Modify the time of activation

      The new initial temperature distribution is calculated at the end of the simulation, not in the beginning. It is recommended that you specify the time of activation as $\frac{3}{4}$ the time of simulation.

   e. Specify the desired thickness distribution.

      Modify the desired thickness distribution

      This option allows you to specify the thickness distribution that you want to get at the end of the thermoforming process. You must be careful when defining this distribution. Only limited improvements are possible by modifying the initial temperature distribution of the sheet. You can now specify a constant thickness (for all the sheet), a thickness distribution specified in a CSV file, or specify a thickness distribution by defining a set of zones (boxes or spheres, of specified dimensions, of specified law, or of thickness distribution constant or linear). In the last case, you must also specify a default thickness for vertices outside of defined zones.

Important

You must keep in mind that the desired thickness distribution must be defined at a position corresponding to the end of the thermoforming process (and not in the initial configuration).

f. Specify the output CSV file.

Modify the output CSV filename

The new initial temperature will be saved in a CSV file. This makes it easier for the next step of optimization, where you have to reread the data file and change the initial temperature in the shell sub-task.

g. Specify the minimum temperature.

Modify the minimum temperature

In order to avoid a shift of the solution during optimization steps, it is necessary to prescribe a minimum temperature. The result is that at every step of optimization, at least one node will be at that minimum temperature.

h. Specify the maximum temperature.

Modify the maximum temperature.

You should specify the temperature that the sheet should not exceed during the thermoforming process (to avoid degradation, e.g.). If the optimized initial temperature exceeds this maximum temperature, it is bounded to that value. If you observe that the new initial temperature reaches the maximum for several successive steps of optimization, you should look at the compatibility between initial thickness and desired thickness distributions. Moreover, you should also look at the under-relaxation factor, size of rectangles, and mesh refinement of the sheet.

i. Specify the search radius.

Modify the search radius

This option is available if the size of the rectangles is set to zero. In this case, the new initial temperature at one vertex will be the minimum of the new estimated temperature of points in the vicinity of the vertex (at a distance smaller than the search radius). The default search radius is set to the length of the largest segment existing in the initial mesh of the sheet.

Once all parameters in the **Temperature programming for thermoforming** menu have been specified, select the **Upper level menu** option to leave the menu and go to the **Restriction of layers by contact domain** menu.

4. In the **Restriction of layers by contact domain** menu, you will have the opportunity to restrict the calculation to the part of the sheet that is actually in contact with a part of the mold. If you do not want to restrict the calculation in this way, simply click on **Upper level menu** in the panel. If you want to restrict the calculation, follow these steps:

   a. Select the contact domain to which you want to restrict the calculation. By default, it will be preceded by **Ignore contact with**, which indicates that the calculation will be limited to the portion that is not in contact with this domain.

   b. Click **Modify**.

   c. Select **Restricting contact domain**.

   d. Repeat to restrict the calculation based on contact with other subdomains.

   e. Click **Upper level menu**to complete the definition.

   The calculation will now be limited so that only the initial temperature of the part of the thermoformed sheet that is in contact with the specified subdomain(s) will be optimized.

Chapter 18: Film Casting

ANSYS POLYFLOW provides techniques for solving film casting and film stretching problems. Information on these techniques is presented in the following sections:

18.1. Introduction

For the purposes of this chapter, a film is a sheet of polymer with a thickness that is several orders of magnitude smaller than its other dimensions. It is also assumed that the film is flat, so that effects due to film curvature are ignored.

Consider the problem of modeling the flow between a die exit and a take-up roll, where entrance and exit effects are ignored. The problem is truly three-dimensional, since the film will show necking along lateral sides as well as a thickness variation along the principal direction of stretching. However, because of the very small thickness, a 3D model would not be feasible: it would require a very large mesh which would mean a costly computation. Instead, a special 2D model has been developed in ANSYS POLYFLOW. This model can be compared to the planar stress approximation in elasticity, which reduces 3D problems to 2D.

18.2. Theory

18.2.1. Overview

The 2D calculation domain for film flow problems involves one or two free borders, which are subsequently referred to as free surfaces. The finite element mesh will be deformed throughout the calculation according to the kinematic condition on the free surface.

All conservation equations are averaged throughout the third dimension (i.e., the film thickness) and thickness-averaged velocity and temperature values are computed. The film thickness becomes a variable of the flow problem, which is governed by the mass conservation equation. Pressure is no longer a variable of the problem and is eliminated in the momentum equations.

Fluid models can be either generalized Newtonian or viscoelastic, isothermal or nonisothermal. Unlike for the shell model used in blow molding and thermoforming simulations, only steady-state modeling is available for film casting. Furthermore, for viscoelastic flows, both single- and multi-mode differential models are available. It is, however, important to note that only viscoelastic models with a constitutive equation written in terms of the extra-stress tensor, as well as the FENE-P and DCPP models, are available for film casting simulations.

For generalized Newtonian fluids, as described in *Generalized Newtonian Flow* (p. 207), the viscosity can depend on the temperature and the second invariant of the rate-of-deformation tensor. These dependences obey algebraic relationships identical to those described in *Generalized Newtonian Flow* (p. 207). For nonisothermal flow problems, it is important to recall that the temperature is considered to be constant throughout the film thickness, although the heat loss on both sides of the film can be taken into account.

For viscoelastic flows, since film casting is mainly an extensional flow, it is possible for viscoelastic effects to dominate. See *Viscoelastic Flows* (p. 225) for information on the available constitutive equations.

18.2.2. Flow Equations

The film is assumed to be geometrically described in the x- y plane. Let σ stand for the Cauchy stress tensor, and $\mathbf{v}$ for the velocity vector. On the upper and lower free sides of the film, zero traction conditions apply. In view of the small film thickness, it is assumed that this is true across the film thickness. From the vertical momentum equation, the pressure, p, can be found:

$$\sigma_{zz} = -p + \mathbf{T}_{zz} = 0\,, \quad \sigma_{xz} = \sigma_{yz} = 0 \tag{18–1}$$

That is,

$$p = \mathbf{T}_{zz} \tag{18–2}$$

where $\mathbf{T}_{zz}$ is the extra-stress component that obeys the fluid constitutive equation. Because the divergence of the velocity is equal to zero, the following equation holds:

$$\frac{\partial v_z}{\partial z} = -\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}\right) \tag{18–3}$$

where z is the coordinate in the film-thickness direction.

Let h be the thickness of the film, equal to the value of z at the upper side of the film. Mass conservation on a film element of thickness h yields

$$\frac{\partial\,(hv_x)}{\partial x} + \frac{\partial\left(hv_y\right)}{\partial y} = 0 \tag{18–4}$$

This partial differential equation (*Equation 18–4* (p. 418)) is of the hyperbolic type. From a mathematical viewpoint, a boundary condition for the thickness h must be specified along a line crossing all flow trajectories at their entrance into the computational domain. In the geometric model, this line corresponds to the die exit, and hence to the film entrance. Note that the thickness must be specified only along this line.

18.2.2.1. Boundary Conditions

When solving a film problem, the thickness-averaged velocity field, which is a function of x and y, will be calculated. As in any flow problem, flow boundary conditions must be specified. General information on boundary conditions can be found in *Boundary Conditions* (p. 173). When a nonisothermal film flow is solved, an average temperature equation will be calculated as well. This field obeys the energy equation. Boundary conditions must therefore be specified for the temperature field.

18.2.2.2. Stress Boundary Conditions for the DCPP Model in Film Casting

For the particular case of the DCPP model used within a film casting flow simulation, care may be required when imposing the boundary conditions for the viscoelastic variables at the inlet of the calculation domain. The inlet is identified as the border where the thickness condition is imposed. Here, instead of imposing values for all components of the orientation tensor, you will be asked about anisotropy described in terms of orientation and factor, as well as stretching amplitude. Imposing boundary conditions for the viscoelastic unknowns (orientation and stretching) is optional. If conditions are imposed, you will need to provide the following information:

- the direction of anisotropy of the orientation tensor (x or y)
- the anisotropy factor ranging between 0 (isotropic) and 1 (anisotropic)
- the inlet stretching

18.2.3. Multilayer Films (Coextrusion)

Some film casting applications involve several fluid layers. In view of the model, all fluid layers will overlap on the same computational domain. The velocity field, as well as the temperature field (for nonisothermal problems), will be uniquely defined on the computational domain, while an independent thickness field h will be assigned to each fluid layer. Also, for viscoelastic flow problems, an independent extra-stress tensor $\mathbf{T}$ will be assigned to each fluid layer.

18.2.4. Heating and Cooling of the Film

In the industrial process of film casting, air can be blown on the film to heat or cool it. You can model the effect of this in ANSYS POLYFLOW. In general, the surface heat flux q is obtained as a combination of the following:

- a constant heat flux q_0
- a heat transfer by convection characterized by a coefficient h and a reference temperature T_α
- a heat transfer by radiation characterized by a coefficient σ and a reference temperature T_σ

The heat flux can be written as

$$q = q_0 + h(T - T_\alpha) + \sigma((T + T_0)^4 - (T_\sigma + T_0)^4) \tag{18–5}$$

In polymer applications, however, the transfer through radiation is negligible with respect to the heat transfer by convection. Specification of the heat transfer coefficient applies for both sides of the film (the sum of the upper and lower heat transfer coefficients).

18.2.5. Stream Function

ANSYS POLYFLOW automatically computes a stream function based on the velocity and the thickness fields.

In view of the mass conservation equation, *Equation 18–4* (p. 418), a stream function ψ can be defined by

$$hv_x = \frac{\partial \psi}{\partial y}, \quad hv_y = -\frac{\partial \psi}{\partial x} \qquad \textbf{(18–6)}$$

where h is the local value of the total thickness of the film. Then, ψ obeys a Poisson equation:

$$\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} = \frac{\partial\,(hv_x)}{\partial y} - \frac{\partial\left(hv_y\right)}{\partial x} \qquad \textbf{(18–7)}$$

The solution of *Equation 18–7* (p. 420) is defined up to an arbitrary additive constant. Assigning one point where the value of the stream function is zero will determine this additive constant. By default, this is done at the node closest to the point $(0,0)$.

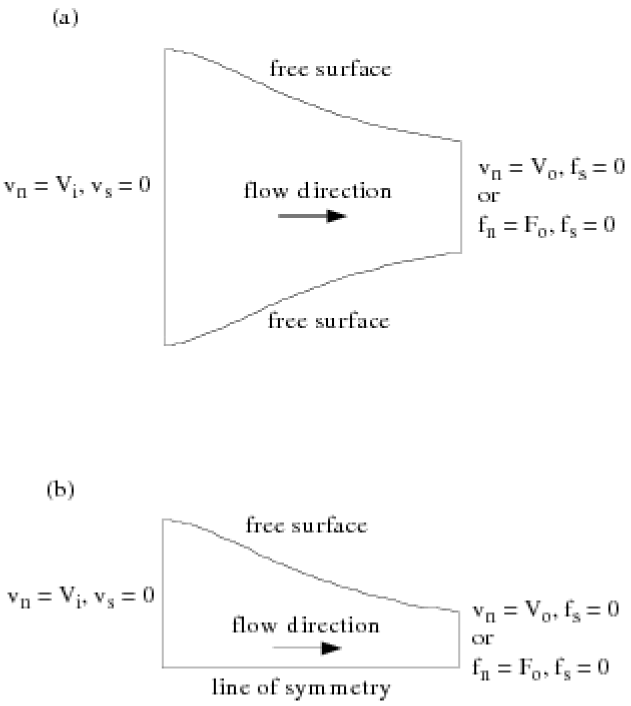
18.2.6. Film Problems and Nonlinearity

A film flow problem is characterized by several nonlinearities. At first, the problem involves one or two free boundaries, as in *Figure 18.1* (p. 420), which introduce a coupling between the velocity and the coordinates. Additionally, *Equation 18–4* (p. 418) couples the velocity to the film thickness.

Other nonlinearities can arise from the constitutive equations and the material data, such as shear-rate- or temperature-dependent viscosity. Nonlinearities can also be generated by the boundary conditions. This is especially true when the take-up velocity or force at the exit of the computational domain leads to strong changes in the geometry and/or in the thickness.

When the level of nonlinearity is significant, you should use an evolution scheme (e.g., on the longitudinal extension ratio). See *Evolution* (p. 517) for information on evolution.

Figure 18.1 Example of Flow Boundary Conditions for Film Casting



18.3. Inputs for Film Casting Problems

18.3.1. General Procedure

The steps for defining a film model are listed below:

1. Define a sub-task for the film model.

   Create a sub-task

   a. Select the appropriate problem type.

      Film model: Gen. Newtonian isothermal

      Film model: Gen. Newtonian non-isothermal

      Film model: Viscoelastic isothermal

      Film model: Viscoelastic non-isothermal

   b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. (nonisothermal flows only) Set the general parameters of the material.

   Material data

   The two items in this menu are **Average temperature** and **Heat source per unit volume**. The latter is constant, while the former can be constant, or a linear function of mesh coordinates.

4. Define the flow boundary conditions.

   Flow boundary conditions

   See *Flow Boundary Conditions* (p. 425) for guidelines, and *Boundary Conditions* (p. 173) for general information on boundary conditions.

5. Define the thermal boundary conditions (nonisothermal flows only).

   Thermal boundary conditions

   See *Boundary Conditions* (p. 173) and *Problem Setup* (p. 286) for details. See also *Thermal Boundary Conditions* (p. 425) for specific information for film problems.

   If you want to model the effect of heating or cooling air onto the film surface, select **Surface Heat source** in the **Thermal boundary conditions** panel.

   Surface Heat source

This will bring up a menu in which you can define the surface heat flux parameters q_c, α , T_α, σ , T_σ, and T_0 described in *Heating and Cooling of the Film* (p. 419).

6. Select a remeshing rule.

   #### Global remeshing

   For information on remeshing, including descriptions of available remeshing rules, see *Remeshing* (p. 343).

7. Modify the interpolation scheme (optional).

   #### Interpolation

   See *Controlling the Interpolation* (p. 218) for information on interpolation in generalized Newtonian flows, and *Selecting the Interpolation* (p. 246) for viscoelastic flows. See *General Procedure* (p. 286) for information on interpolation in nonisothermal flows.

8. Define the layers of the film.

   #### Define layers

   Follow these steps to define a new layer.

   a. Select **Creation of a new layer**.

   b. Define the physical properties for this layer (viscosity, density, etc.).

      #### Material data

      Each layer can have distinct physical properties.

   c. Set the boundary conditions for this layer.

      #### Thickness boundary conditions

      By default, ANSYS POLYFLOW specifies no boundary conditions for the film thickness.

      i. Select the side along which a thickness boundary condition will be imposed and click **Modify**.

      ii. Select **Thickness imposed** and specify film thickness as a constant, as a linear function of coordinates, or by means of a UDF.

   d. Set the initial thickness for this layer.

      #### Initial thickness

      The initial thickness of the film layer is used for initializing the unknown corresponding field. Usually, it is reasonable to use an average value of the specifications you employed when defining the **Thickness boundary conditions**. The initial thickness of the film layer can be set in one of four ways:

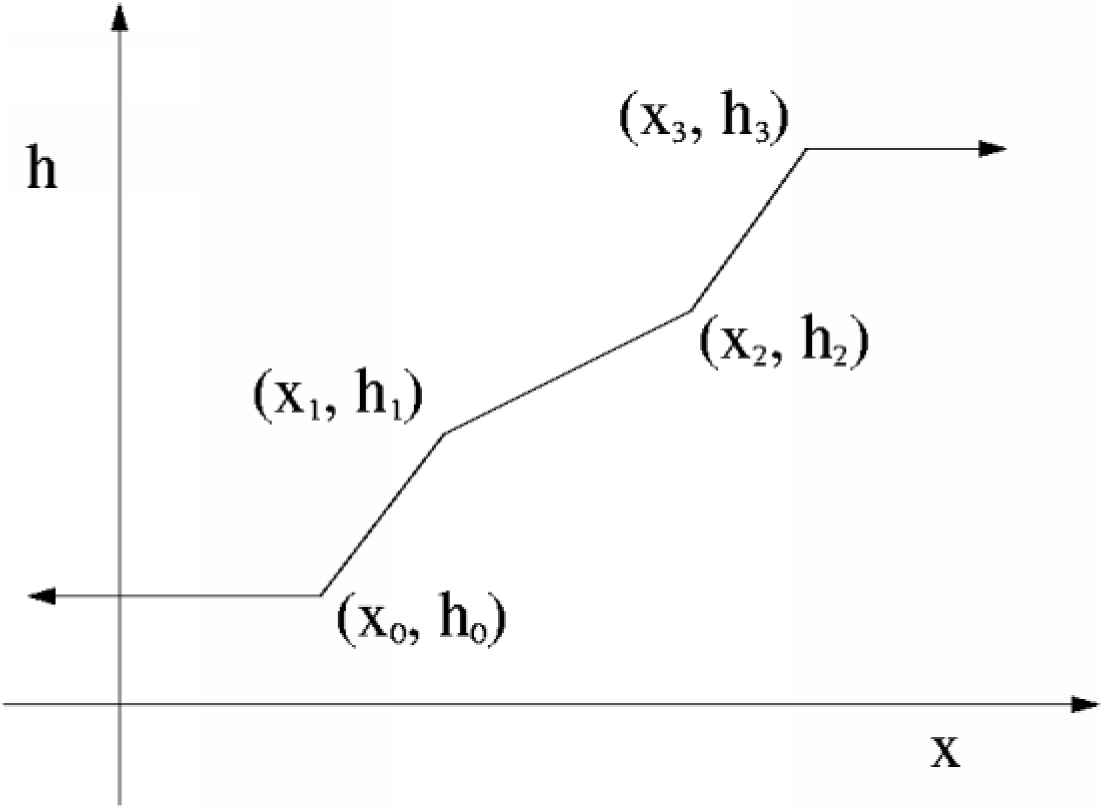
      - As a **Constant**. You must specify a value that will be the initial thickness over the entire fluid layer.

- As a **Linear function of coordinates**. Here, you will specify three values A, B, and C (in 2D) to define the thickness as

$$\text{thickness} = A + Bx + Cy \tag{18–8}$$

- As a **Multi-ramp function** of x or y. You will enter a sequence of pairs (x_i, h_i) $, i = 0, \ldots, n$ (or $\left(y_i, h_i\right)$ as appropriate) that defines the thickness as a piecewise linear function (see *Figure 18.2* (p. 423) for an example).

Figure 18.2 Example of a Multi-Ramp Function



i. Select **Define new pairs** and enter data pairs when prompted.
ii. Select **Check existing pairs** to check your inputs.
iii. To modify an incorrect pair, select **Modify mode** and then select the pair to be modified.
iv. To delete an incorrect pair, select **Delete mode** and then select the pair to be deleted.

The set of points you enter can be saved for future use using the **Save in a Dependence Data File** menu item, and recalled later with the **Read in a Dependence Data File** menu item.

Instead of interactively entering the data for the multi-ramp, the data can be read from a file, where the pairs of data must be written with the fixed (Fortran) format 2e14.7. This means that each pair of data (X_i, f (X_i)) is written on a single line, and fourteen digits are used for representing each data as ±0.1234567e ±89. Since the format is fixed, there is no need to use a separator. For a single data, the format e14.7 successively involves: a sign, a 0 digit, the decimal dot, seven digits (for the mantissa), the letter “e" standing for a power of 10 eventually given by its sign and the last two digits. In such a format, - π is then represented as -0.31415926e+01. If an entry $(X_i, f\ (X_i))$ in the file consists of (0.25,- π), for example, it will be represented as+0.2500000e+00-0.31415926e+01. Subsequent pairs of data are written on subsequent lines.

- From data in a CSV (Excel) file. In this case, values to be specified as a boundary condition are defined in a comma separated values (CSV) file, in a way similar to that described in *Results Interpolation Onto Another Mesh* (p. 150) for specifying data on the whole domain. ANSYS POLYDATA will prompt you for the name of the CSV file, as well as the case-sensitive tag used to identify the data for this boundary condition. Correspondence between mesh nodes and data points is not required; ANSYS POLYFLOW will interpolate the data to the mesh nodes.

e. For the DCPP model, set the inlet stress for the layer.

Enable stress condition

By default, the anisotropy direction is along the X axis, the anisotropy factor is 0, while the stretching scalar is 1. This can be changed when needed.

Switch to Y for direction of stretching

By selecting this option, the anisotropy will be defined with respect to the y direction.

Modify value of anisotropy factor

By selecting this option, you can modify the anisotropy factor s in the direction defined previously. Any value between 0 and 1 is valid. If $s = 0$ (isotropic), all three components of the orientation tensor are set to $\frac{1}{3}$; if $s = 1$, we have a 100% orientation along the selected direction of anisotropy and thus a unit value for the corresponding component of the orientation tensor, while the other components vanish.

Modify value of stretching

By selecting this option, you can modify the stretching amplitude in the direction defined previously. Any positive value is valid. However, recommended values should be at least equal to unity; values lower than unity are of course permitted, but would probably not match the physics in the die.

f. (optional) Give the layer a name. By default, all layers are simply called **new layer**.

Modification of title

To modify the definition of a layer you have already created, select its name from the **Define layers** menu and follow the steps above.

To delete a layer, select the **Deletion of a layer** menu item and then choose the layer to be deleted.

18.3.2. Guidelines for Setting Boundary Conditions

18.3.2.1. Flow Boundary Conditions

For a film problem, boundary conditions for the velocity should typically be set as follows:

- A uniform normal velocity profile should be specified at the inlet to the computational domain, while a take-up velocity or force should be imposed at the exit, together with a vanishing tangential force.
- For the two lateral sides, two free surfaces or one free surface and one line of symmetry should be defined. The free surface condition requires additional data (see *Free Surfaces and Extrusion* (p. 311) for information on defining free surfaces). Briefly, a fixed point for the free surface, and possibly some constraint on the directors should be specified. Also, it is usually advised to enable the upwinding for the kinematic equation on the free surface(s).

As an example, consider the computational domain displayed in *Figure 18.1* (p. 420).

It is important to note here that the velocity on the take-up roll usually differs from the entrance velocity by a factor of between 10 and 50. To get the problem to converge, the velocity at the exit is increased using an evolution scheme (see *Evolution* (p. 517)), starting from a rigid body translation problem, and evolving to the actual stretching problem.

18.3.2.1.1. Multilayer Film Casting Problems

In multilayer film flow problems, the various layers are overlapping on the same computational domain. Therefore, no interface boundary conditions are needed between layers. Continuity conditions are automatically satisfied in the model because of the unique definition of the velocity and temperature fields.

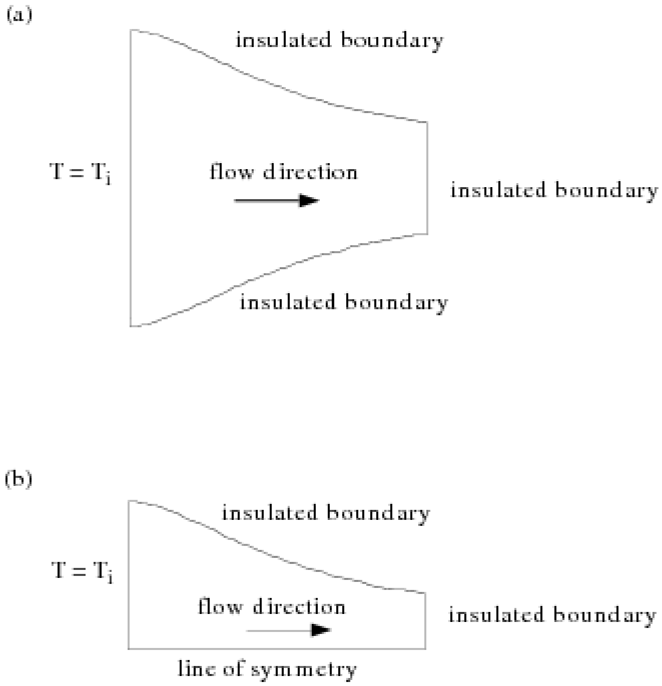
18.3.2.1.2. Viscoelastic Flow Film Problems

For viscoelastic flow problems, set the boundary conditions for the velocity field, but not for the extra-stress tensor. The solution will produce a stress field that is still in equilibrium according to the momentum equation, although additional data should be specified for the hyperbolic constitutive equation. However, the effect of not prescribing this part of the extra-stress tensor has very minor effects on the solution itself.

18.3.2.2. Thermal Boundary Conditions

For nonisothermal film flow problems, a temperature should typically be imposed at the inlet to the computational domain, as displayed in *Figure 18.3* (p. 426), while no heat exchange is assumed along the other boundary segments.

Figure 18.3 Example of Thermal Boundary Conditions for Film Casting



18.3.2.2.1. Multilayer Film Problems

In a multilayer film problem, several layers will overlap the same computational domain. However, no interface boundary conditions are needed between layers, since continuity conditions are automatically satisfied by the model, in view of the definition of an average temperature field. In particular, this is compatible with the basic assumption that there are no temperature variations throughout the thickness of the film.

18.3.2.3. Stress Boundary Conditions for the DCPP Model in Film Casting

Suggested boundary conditions for the viscoelastic variables of the DCPP model can only be defined along the side which is selected for thickness boundary conditions. Imposing boundary conditions for the viscoelastic unknowns (orientation and stretching) is optional. If conditions are imposed, you will need to provide the following information:

- the direction of anisotropy of the orientation tensor (x or y)
- the anisotropy factor ranging between 0 (isotropic) and 1 (anisotropic)
- the inlet stretching

By default, the anisotropy direction is along the X axis, the anisotropy factor is 0, while the stretching scalar is 1. This corresponds to a situation where the melt has not undergone deformation in the die before being extended. Actually, the flow in a die is dominated by shear, which may certainly affect the orientation tensor, while stretching will remain close to unity. Therefore, a typical inlet boundary condition for film casting would involve a nonzero anisotropy factor and a unit stretching. You can change this according to your needs. On the basis of the value entered for the anisotropy direction and anisotropy factor s, the value of the orientation tensor S at the inlet can be evaluated and assigned as:

$$S_{aa} = \frac{2s+1}{3}, \quad S_{bb} = S_{zz} = \frac{1-s}{3} \tag{18–9}$$

where a is the selected direction of anisotropy, b is perpendicular to it in the plane of the film; and the component $S_{xy} = 0$. If $s = 0$ (isotropic), all three components of S are set to $\frac{1}{3}$; if $s = 1$ (100% anisotropy), we have $S_{aa} = 1$, while the other components vanish. Any value between 0 and 1 is valid. For the stretching variable, it is recommended that the values should be at least equal to unity; values lower than unity are of course permitted, but would probably not match the physics in the die. When a multilayer system is defined, inlet boundary conditions can be defined independently for each layer, as is already the case for the thickness. In the case of multi-mode model for a given layer, all modes will undergo the same selected inlet conditions. Finally, it is worth mentioning that the solver exhibits better performances when reasonable inlet boundary conditions are selected for the viscoelastic unknowns of the DCPP model in film casting.

Chapter 19: Chemically Reacting Flows

Information about modeling chemically reacting flows is presented in this chapter.

19.1. Introduction

Generally, a chemically-reactive system involves a mixture of different components subject to several simultaneous chemical reactions. When chemical reactions are modeled together with flow (and possibly temperature) variables, it is necessary to drop the hypothesis that the chemical composition in the flow domain is homogeneous. On the contrary, the composition will vary from point to point under the influence of the transport phenomena inherent to all dynamic situations. Chemical “species" will be diffused and transported by the flow, and chemical reactions will act as sources or sinks of the chemical species. Establishing a global model for this process of transport and reaction is described in this chapter.

19.2. Theory

19.2.1. Overview of Reactions

Chemical reactions, which act as sources or sinks of species, are interactions between processes that result in the formation or destruction of new molecular species. Quite generally, they may take place in the bulk fluid or be restricted to surfaces or interfaces. The direct consequence of these reactions is a change in the mass of the various species in the system, along with generation or consumption of thermal energy.

Transport phenomena such as diffusion or advection are not unique to chemically reacting flows; they can be found in most models incorporated in ANSYS POLYFLOW. For chemical species, it is quite common for the advection mechanism to dominate the transport, and for diffusion under a gradient of concentration to be small. There is also sometimes a practical difficulty to the measurement of diffusion coefficients for chemical species. Therefore, it is not unusual to neglect the diffusive part of the transport.

Processes such as fusion, evaporation, and precipitation are physical reactions; there is no change in chemical composition associated with these processes. Physical reactions are intra- and inter-molecular interactions that result only in a phase change of the species involved. For instance, the melting of ice into water is a physical reaction. Ice and water share the same chemical composition, though they represent different phases. By treating each phase of a particular chemical species as a separate component, physical reactions can be treated as chemical reactions.

Reactions can be reversible or irreversible. Consider, for instance, a system of three reactions between four species, A, B, C, and D:

reaction #1:	2A + 3B	→	4C
reaction #2:	A	→	2D
reaction #3:	2D	→	A

The second and third reactions cancel when added together. These reactions are said to be reversible: species A forms species D, which, in turn, re-forms species A. We can express reactions #2 and #3 in a single expression using the following concise notation: A $\leftrightarrow$ 2D The first reaction, on the other hand, is said to be irreversible, since species C does not react to form species A and B.

The concentration of the various species may be expressed in many ways. In ANSYS POLYFLOW, mass fractions and diffusive mass fluxes relative to the mass averaged velocity of the mixture are used.

19.2.2. Definitions

Let ρ_i denote the mass concentration of species i, which is the mass of species i per unit volume of mixture. Given a total of N concentration species, the local density of mixture is defined as

$$\rho \equiv \sum_{i=1}^{N} \rho_i \tag{19–1}$$

In a flow of a mixture, the chemical species are moving at different velocities. Let $\mathbf{v}_i$ denote the velocity of species i with respect to the coordinate system. The local mass-averaged velocity is given by

$$\mathbf{v} \equiv \frac{1}{\rho}\left(\sum_{i=1}^{N} \rho_i \mathbf{v}_i \right) \tag{19–2}$$

This is the local velocity that would be determined by means of a Pitot tube or other methods that measure force or pressure, and corresponds to **v** as commonly used for pure fluids. Other definitions of velocity, such as molar- and volume-averaged velocities, can also be used. They have, however, the inherent disadvantage that they do not appear explicitly in the fluid-mechanics equations [*4*] (p. 715).

19.2.3. Advection-Diffusion Mechanism

A simple mass balance around species i gives the following advection-diffusion equation:

$$\frac{\partial}{\partial t}\left(\rho_i\right) + \nabla \cdot \mathbf{n}_i = R_i + S_i \tag{19–3}$$

Here, $\mathbf{n}_i$ is the mass flux of species i, R_i is the net mass rate of creation or depletion of species i, and S_i represents the contribution of any user-defined rate to the mass creation or depletion rate. The mass flux of species i is a vector quantity denoting the mass of species i that passes through a unit area per unit time. It corresponds to the sum of the mass flux of species i resulting from the bulk motion of the fluid together with a superimposed diffusion:

$$\mathbf{n}_i = \rho_i \mathbf{v}_i = \rho_i \mathbf{v} + \mathbf{j}_i \tag{19–4}$$

Hence, the conservation equation for a chemical species takes the following general form:

$$\frac{\partial}{\partial t}(\rho\omega_i) + \nabla\cdot(\rho\omega_i\mathbf{v}) + \nabla\cdot\mathbf{j}_i = R_i + S_i \qquad \textbf{(19–5)}$$

where $\omega_i = \frac{\rho_i}{\rho}$ is the mass fraction ($0 < \omega_i < 1$) of the chemical species i. In an N-component mixture, there are $N - 1$ independent species concentration equations of the form of *Equation 19–5* (p. 431), since the mass fractions sum to 1.

Diffusion is limited to situations where the diffusive flux is due to concentration gradients only (the Fickian effect):

$$\mathbf{j}_i = -\rho D_{im}\nabla\omega_i \qquad \textbf{(19–6)}$$

where D_{im} is the diffusion coefficient for species i in the mixture. In nondilute mixtures, the variation of D_{im} with the local mixture composition may adopt far more complicated forms that are not handled in ANSYS POLYFLOW. Temperature gradients, pressure gradients, and external forces may also contribute to the diffusion flux, although their effects are usually minor.

It is usually quite difficult to measure diffusion coefficients for the transport of chemical species. It is also difficult to find those values in the literature. However, in practical situations, the role of diffusion is frequently smaller than that of advection, which is always present, and it is not unusual to neglect the diffusion mechanism.

To close the system of equations, an equation of state relating density to the state of the mixture is required. For dilute liquid solutions, an acceptable equation of state is “constant density".

For gases at low pressure and high temperature, it is customary to make use of the ideal gas law, which relates density to pressure and temperature. More generally, however, equations of state are nonlinear algebraic relationships relating density to the field variables (such as temperature, pressure, and composition).

19.2.4. Chemical Reactions

The source of chemical species due to reaction, R_i, is computed as the sum of the reaction sources over the j reactions that the species may participate in:

$$R_i = M_i \sum_{j\ \mathrm{reactions}} v_{ij} r_j \qquad \textbf{(19–7)}$$

where M_i is the mole mass of species i

v_{ij} is the stoichiometric coefficient of the i-th species in the j-th reaction

r_j is the molar reaction rate of reaction j

Note that in *Equation 19–7* (p. 431),

$v_{ij} < 0$ for the reactants of the forward reaction

$v_{ij} > 0$ for the products of the forward reaction

The individual reaction rates r_j are themselves functions of the form

$$r_j = K_{j} \prod_{k \text{ reactants}} a_{kj}^{\phi_{kj}} \tag{19–8}$$

where K_j is a rate "constant", the terms $a_{kj} = \rho_{kj} / M_{kj}$ are the molar concentrations of the reactant species k, and ϕ_{kj} are a set of exponential powers.

The values of the rate constant and the exponential orders are dictated by a reaction mechanism that is an elementary step-by-step description of a chemical interaction.

The exponential orders may be different from the stoichiometric coefficients of the reaction itself.

For a given reaction mechanism, the influence of temperature on the rate constant can be expressed in terms of an Arrhenius type relationship involving the universal gas constant R and the absolute temperature T:

$$K_j = k_j \left(T^{\beta_j} \right) \left(e^{-\frac{E_j}{RT}} \right) \tag{19–9}$$

where k_j is a pre-exponential factor

β_j is a temperature constant

E_j is the activation energy

A change in temperature, however, may modify the reaction mechanism. *Equation 19–9* (p. 432) holds as long as a change in the system does not change the reaction mechanism. It is a subtle fact that a change in composition can also change the reaction mechanism.

Considering an unknown rate constant K at any temperature T, and a reference rate constant K_r at a reference temperature T_r, the pre-exponential factor can be eliminated from *Equation 19–9* (p. 432) to yield

$$K = K_r \left(\frac{T_a + T}{T_a + T_r} \right)^{\beta} e^{-\frac{E}{R} \left(\frac{1}{T_a + T} - \frac{1}{T_a + T_r} \right)} \tag{19–10}$$

T_a represents a shift in temperature. It is set to 0 by default.

This relationship has proven useful to relate K to T from appropriate reference values K_r and T_r. The temperature can also be expressed in a relative scale in conjunction with the absolute temperature with respect to the zero of the relative scale (T_a).

Chemical reactions are either exothermic or endothermic (i.e., they either generate or consume energy). Energy (positive or negative) associated with the chemical reaction acts as a source term of the energy equation as follows:

$$\sum_j r_j \Delta H_j \tag{19–11}$$

where ΔH_j is the heat of reaction j.

where ΔH_j is the heat of reaction j. Note that the temperature of the system strongly affects the rate of reaction; as a general rule, a change in temperature of about ten degrees changes the reaction rate by a factor of two.

19.3. User Inputs

The general procedure for modeling chemical reactions is as follows:

1. Create a new task of the appropriate type.
2. Define the species, as described in *Defining Chemical Species* (p. 433).
3. Define the reactions, as described in *Defining Chemical Reactions* (p. 434).
4. Define a sub-task to solve the transport equation for each species, as described in *Defining the Species Transport Equations* (p. 437) and *Defining a Closure Equation* (p. 440).

See also *Chemical Reactions and Evolution* (p. 441) for information about using evolution for a chemical reaction calculation.

19.3.1. Defining Chemical Species

The first step to defining a chemical reaction is the creation of the chemical species involved. Select the **Define species** menu item in the task menu and follow the steps below for each species you want to define.

Define species

1. Specify the new species.

   Create a new species

   ANSYS POLYDATA will ask for two names for each new species: a full name (referred to as the name) and a short name (referred to as the nickname). Nicknames will be used when you define the reactions for these species. Names and nicknames must be unique from species to species, though the name and nickname of a particular species can be the same.

   Names and nicknames can be any sequence of characters that meets the following criteria:

   - The first character is a letter.
   - The remaining characters (if any) are letters, digits, underscore symbols (_) or dollar signs.
   - Names can have at most 16 characters, while nicknames are limited to 4 characters.

For example, A, H2O, and i_c_ are acceptable nicknames, while 2ps (starts with a number), J&B (contains an ampersand), and water (has too many characters) are not.

ANSYS POLYDATA and ANSYS POLYFLOW are case sensitive, i.e., they distinguish between uppercase and lowercase letters.

You can delete a species at any time (using the **Delete a species** menu item) provided it is not involved in a chemical reaction or transport equation definition. If it is, you should delete that reaction or equation before deleting the species.

2. After you have given the species a name and nickname, it will appear in the **Define species** menu. Select it to go to the **Species parameters** menu, where you can set the molecular mass of this species.

Modification of mole mass

19.3.2. Defining Chemical Reactions

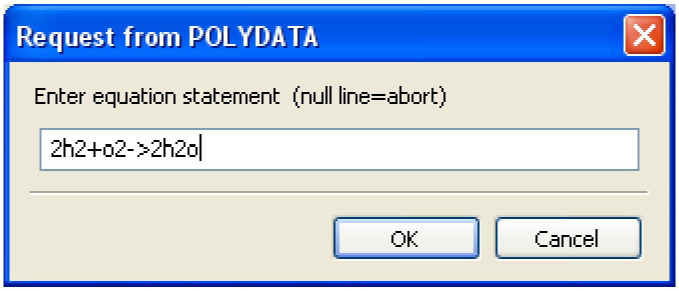
Once you have defined all of the relevant species, you can define the chemical reactions in which they participate. Select the **Define reactions** menu item in the task menu and follow the steps below for each reaction you want to define.

Define reactions

1. Specify the equation for the chemical reaction.

Create a new reaction

ANSYS POLYDATA will open a dialog box and request input of the reaction equation.



Enter equation statement (null line=abort)

2h2+o2->2h2o

An equation consists of references to species data fields, integers, operators (-> or <=>), and the plus sign +, and has the following form:

$$r_1R_1 + r_2R_2 + \cdots \text{->} p_1P_1 + p_2P_2 + \cdots$$

where	R	stands for a reactant
	r	is a stoichiometric coefficient for a reactant
	P	stands for a product
	p	is a stoichiometric coefficient for a product

There can be up to five reactants and five products in a reaction. Duplicate species are not allowed, and stoichiometric coefficients must be integers with four digits or less. If you don't specify a coefficient, a value of one will be assumed.

If your reaction is reversible, you can enter it using the <=> operator, or enter it as two separate reactions (the forward and the reverse). Note, however, that entering it as two separate reactions requires more input and uses up two of the ten places in the list of chemical reactions.

Embedded and trailing blanks will be ignored. Checks are performed from left to right. If a parsing error occurs, the formula is printed up to where the parser found the error, and you will see useful error statements like "equation statement is incomplete", and so forth. Use this information to correct syntax errors in the formula.

Once you have entered the equation correctly, the formula will appear in the reaction list. Some examples of valid and invalid equations, using three species with nicknames a1, a2, and a3 is as follows:

valid:

1 a1 + 2 a2	->	3a3
a1	<=>	2a3
a1	->	2 a3
2 a3	->	a1

invalid:

a1 + 2	->	a3	(missing species identifier)
a1 + a2	<=>	a1 + a3	(a1 appears twice)
a1 ++ a2	->	a3	(adjacent plus signs)
a1 + a2	->	a4	(a4 is not pre-defined)

2. Set the parameters for the reaction.

   a. Select the reaction equation in the **Define reactions** menu.

   b. Check the mass balance.

      Check mass balance

      A nonzero mass balance, while incorrect, will not yield an error message, nor will it cause the chemical reaction to be rejected. You should therefore be sure to check the mass balance yourself.

   c. Specify the domain of the reaction

      Domain of reaction

      This option allows you to select the subdomain(s) where the chemical reaction will take place. By default, the domain of the reaction is the entire mesh, but restriction to particular subdomains of the mesh is perfectly valid. Chemical reactions cannot be restricted to a boundary or an interface.

   d. If the reaction is reversible and you want to reverse it, select the **Reverse reaction** menu item.

      Reverse reaction

If the rate constant of either the forward or the reverse reaction is known, the reaction speed in the opposite direction can be calculated from the reaction equilibrium thermochemistry. The reaction mechanism for the reversed reaction is the same as for the original, with the products becoming the reactants and vice versa. If you want, you can modify the reaction parameters, using the reversed reaction as a shortcut for defining a new reaction.

e. Specify the reaction rate constant, by defining the parameters used in *Equation 19–10* (p. 432).

Rate ’constant’

Enter the values for **Ta**, **Tr**, **prefac** (K_r), **beta**, **Ea/R** (E/R), and **entalp** (enthalpy used in the energy equation; an exothermic reaction corresponds to a positive value for enthalpy, and an endothermic reaction corresponds to a negative value).

f. Specify a name for the reaction equation.

Modification of equation name

By default, the equations are given names like forward #1. You can use this menu item to give them more descriptive names for later reference.

g. If appropriate, switch the reaction orders from integer numbers (the default) to real numbers (optional).

Switch to real orders

Exponential orders are initialized on the basis of the species stoichiometric coefficients: the exponential orders of the direct (or reverse) reaction are assigned the values of the reactants’ (or products’) stoichiometric coefficients; as can be seen, the molar rate of the direct (or reverse) reaction can also depend upon the product (or reactant) species according to the so-called generalized Kamal’s model which accounts for all species present.

If the orders are integers, then a negative mole concentration will not cause a floating point error. If real orders are selected, then the signed modulus of the mole concentration is used when evaluating the molar rate. For most cases, there will be no reason to change to real orders.

Important

For computational reasons, real orders of reaction between 0 and 1 should be avoided. Also, evolution on the exponential orders, whether integer or real, is not permitted.

h. (optional) If necessary, modify the values of the exponential powers for the species orders (ϕ_{kj} in *Equation 19–8* (p. 432)).

Modification of power of c(species)

By default, the exponential power for a reactant is its stoichiometric coefficient and the exponential power for a product is zero.

Here is an example to illustrate how source terms are distributed among the various species transport equations present. Consider the following chemical system:

reaction #1:	2A + 3B	→	4C
reaction #2:	A	→	2D
reaction #3:	2D	→	A

reaction #1:

Source term in transport of A:	$-2M_A r_1$
Source term in transport of B:	$-3M_B r_1$
Source term in transport of C:	$4M_C r_1$

reaction #2:

Source term in transport of A:	$-M_A r_2$
Source term in transport of D:	$2M_D r_2$

reaction #3:

Source term in transport of A:	$M_A r_3$
Source term in transport of D:	$-2M_D r_3$

so that the net mass rate of production/destruction of the various species becomes

$$\begin{aligned} R_A &= M_A(-2r_1 - r_2 + r_3) \\ R_B &= M_B(-3r_1) \\ R_C &= M_C(4r_1) \\ R_D &= M_D(2r_2 - 2r_3) \end{aligned}$$

In parallel, the net contribution to the energy equation will be given by

$$r_1 \Delta H_1 + r_2 \Delta H_2 + r_3 \Delta H_3$$

19.3.3. Defining the Species Transport Equations

Once you have defined the reactions, you will next create a sub-task for each species transport equation (*Equation 19–5* (p. 431)). A transport equation is needed for at least each of the species involved in a chemical reaction (except as noted below). Species not involved in any reactions can also be given a transport equation.

Follow the steps below for each species transport equation:

1. Define a sub-task for the species transport equation.

   Create a sub-task

   a. Select the **Transport of species** sub-task type.

Transport of species

> **Important**
>
> To save CPU time, you can specify a simple **Closure** equation rather than a transport equation for one of the species. See *Defining a Closure Equation* (p. 440) for details. A **Transport of species** sub-task must be defined for all other species.

b. Select the species for which you want to solve a transport equation. This will give the sub-task the name of “Transport of species", where species is the species you have selected.

2. Define the domain on which you want this equation to apply.

Domain of the sub-task

When applicable, it is assumed that the computational domain for a species adjusts to the domain of the velocity field, so this menu item will not be available. As a result, a change in the domain for the flow problem also induces a change in the domain for the transport equation.

If there is no flow problem, then the domain of the transport equation is not a priori limited. All transport equations, however, must share a common domain defined here. If the domain of any of your transport equations is changed, then the domain of all other transport equations will be changed accordingly.

3. Define the details of the transport equation.

Material data

a. Specify the density of the species.

Density

b. Specify the average species concentration, if necessary.

Average concentration

If a flow problem has already been defined, the concentration is taken from the model for the concentration of the flow, which can depend on other variables, including species.

If no flow problem exists, you will need to specify the concentration here for both steady-state and transient problems. Here, zero values for the product mass fractions must be specified, along with homogeneous values for the reactant species.

These values will be used as initial guesses for the numerical scheme in a steady-state problem, or as initial values for a time-dependent simulation.

Note

If the product species appear in the reaction rate expression, no reaction will take place if the product species have a zero concentration. In this case, you should either modify the reaction mechanism or set the average concentration of the product species to a nonzero value.

c. (optional) Specify a constant volumetric source for the transport equation (S_i in *Equation 19–5* (p. 431)).

   Concentration source per unit volume

   This value will be in addition to the source term arising from the chemical reactions.

d. Specify a nonzero value for the species diffusivity.

   Diffusivity

   Transport of species by diffusion is often negligible as compared to that by advection. For numerical reasons, however, it is recommended that the diffusivity be a strictly positive number in steady-state simulations.

4. Define the boundary conditions for the species transport equation. Boundary conditions can be specified in terms of concentration (essential boundary condition) or in terms of concentration flux (natural boundary condition). The concentration variables for the transport equation are mass fractions.

   Concentration boundary conditions

   a. Select the boundary for which you want to set concentration conditions.

   b. Click **Modify**.

   c. Select the boundary condition type you want to impose. For each boundary, there are three or four possible conditions. By default, ANSYS POLYDATA assumes a zero mass fraction on all boundaries.

      - Choose **Mass fraction imposed** to specify a constant mass fraction or a linear function.

        Mass fraction imposed

      - Choose **Flux density imposed** to specify a mass fraction flux.

        Flux density imposed

      - Choose **Insulated boundary** to specify an insulated boundary.

        Insulated boundary

      - Choose **Source of the connected condition** to specify the “source" boundary in a pair of non-conformal boundaries that need to be connected.

        Source of the connected condition

Non-conformal boundary conditions are described in *Non-Conformal Boundaries* (p. 190). The inputs for non-conformal concentration boundary conditions are the same as those for flow boundary conditions, as described in *Connecting Non-Conformal Boundaries* (p. 191), except for the numerical parameters. The first two numerical parameters (element dilatation and amplitude of volume generation) are the same as for flow boundary conditions, but the stabilization factor for flow conditions is replaced by a smoothing factor for concentration conditions. The smoothing factor controls the species diffusion tangential to the surface of the connected boundaries. Ideally, there should be no species diffusion within the connected boundaries, but setting this parameter to zero can lead to numerical problems (e.g., zero pivot) or instabilities. It is therefore recommended to set the smoothing factor to a very small nonzero value.

Important

Changing the value of the smoothing factor from the default value is not recommended, except on the advice of your support engineer.

Repeat these steps for each species for which you want to define a transport equation.

An important point regarding the behavior of species near boundaries is the following. Suppose that the fluid sticks at the wall boundaries. Along such boundaries, residence time is arbitrarily large (for steady-state problems); the species do not move, so they eventually change from reactants to products.

Species near the wall boundaries, however, do move, and gradually change from reactants to products as the chemical reaction proceeds. Therefore, steep concentration gradients in the direction normal to the wall boundaries are generated. The closer to the entry section, the steeper the boundary layer. The natural remedy to this situation is to assume that the fluid slips at the wall boundaries. If slip is assumed, species characteristics are all evolving in a similar manner. You can select slip conditions at the wall using the **Flow boundary conditions** menu item.

19.3.4. Defining a Closure Equation

Optionally, for at most one species, instead of defining a transport equation, you can use the closure equation. By definition, a multicomponent system is said to be closed when all species, including the carrier, inert or not, are considered. Such is the case for nondilute mixtures. When systems are closed, the sum of the mass fractions of all species is one. Using this fact, the mass fraction of a single species can be computed if the mass fractions of all other species are known.

The steps for defining a closure equation for a species are as follows:

Create a sub-task

1. Select the **Closure** sub-task type.

   Closure

2. Select the species for which you want to solve the closure equation. This will give the sub-task the name of “Closure of species", where species is the species you have selected.

The closure equation does not require any material data or boundary conditions, so no further inputs are required.

19.3.5. Chemical Reactions and Evolution

Chemical reactions and fluid material properties often induce strongly nonlinear couplings among velocity, temperature, and concentration fields. Although source terms arising from chemical reactions and equations of state are sometimes complex, it is important to note that these relationships are always algebraic (though nonlinear) and never differential.

Other nonlinearities may originate from the constitutive equations and/or problems involving free surfaces, which introduce a coupling between the velocity and the coordinates of the mesh.

For chemically reacting flows, evolution on both the pre-exponential factor and the enthalpy is recommended. The chemically neutral “Stokes" solution is chosen as an initial guess. Since the Stokes solution is structurally different from the final result, a careful continuation procedure is needed to obtain convergence of the iterative scheme.

See *Evolution* (p. 517) for more details on using evolution.

Chapter 20: Volume of Fluid (VOF) Model

ANSYS POLYFLOW allows you to model a liquid with free surfaces using a so-called volume of fluid (VOF) technique for 2D and 3D problems. The VOF model is strictly an Eulerian formulation; this is in contrast to the combined formulation used in the Arbitrary Lagrangian-Eulerian (ALE) approach, which is the other means of simulating a free surface in ANSYS POLYFLOW.

The VOF model does not require remeshing. Because the numerical technique used for the VOF model does not need to reconstruct the interface (by splines or similar methods) or compute distances between a point and a line or a surface, it is more general and has less limitations than regularly used geometry-based algorithms.

Information about the VOF model is provided in the following sections.

20.1. Introduction

Volume of fluid (VOF) techniques offer a means of simulating a fluid flow with one or several free surfaces, and are popular for injection and filling-type problems. There are probably as many techniques as codes implemented, differing in the way that the “moving front" is handled and advanced. What is common among VOF techniques is the use of a time-dependent approach, which is applied on a fixed mesh; that is, the kinematic condition is not tracked and the simulation domain is not remeshed. Theoretically, a drawback of such techniques is that the location where the zero force condition is applied does not correspond to a region on which natural boundary conditions apply. Consequently, additional approximations are required, beyond those typically employed as part of the finite element method.

Because of the explicit nature of VOF algorithms, a so-called Courant type of limitation always occurs: the time step is limited to a fraction of the time it takes to transport information through an individual element (or cell). It is common to require hundreds of time steps to compute an entire simulation. Because each of these time steps are performed on a fixed domain with an inexpensive numerical technique, a VOF model can still require less CPU time than a moving mesh simulation, such as the Arbitrary Lagrangian-Eulerian (ALE) approach (described in *Free Surfaces* (p. 311)). The VOF model implemented in ANSYS POLYFLOW is intrinsically more robust than the ALE approach when modeling free surfaces that merge, separate, or are convoluted, and allows you to simulate problems that a remeshing-based technique simply could not attack (e.g., a complex cavity filling).

It should be noted that the VOF model is not a replacement for ALE methods in all cases. Because the VOF technique relies on partly filled cells (which means that the location of the free surface does not correspond to an element boundary), it is intrinsically less accurate then the ALE technique for flows with well defined interfaces, such as extrusion or multiple layer problems. The ALE method may be more appropriate when searching for a well defined steady-state free surface through a steady-state (or an evolution on a free surface) algorithm.

Important

The VOF model does not account for the effect of surface tension on the free surface.

Important

The VOF model has not been tested in combination with other ANSYS POLYFLOW models, such as the mesh superposition technique (MST), internal radiation (discrete ordinates model), etc. Due to the nonlinearities and dependencies involved in the various models, it is not possible to guarantee the convergence and accuracy of problems that combine VOF with other models. It is recommended that you attempt such combinations with caution.

20.2. Theory

ANSYS POLYFLOW's volume of fluid method tracks a liquid fluid on a domain Ω. A material property variable φ represents the fluid fraction. φ is used to identify where the fluid is present, and is governed by the following transport equation:

$$\frac{\partial \varphi}{\partial t} + \vec{v} \cdot \nabla \ (\varphi) \ = 0 \qquad \textbf{(20–1)}$$

where $\vec{v}$ represents the velocity of the fluid. Note that the values for φ that result from solving the previous equation are highly discontinuous, and hence difficult to simulate.

Equation 20–1 (p. 444) is governed by initial and inlet conditions:

- initial conditions

  You must define the φ variable for the domain at the initial time, in order to specify where fluid is initially present and not present. This necessarily introduces discontinuities.

- inlet conditions

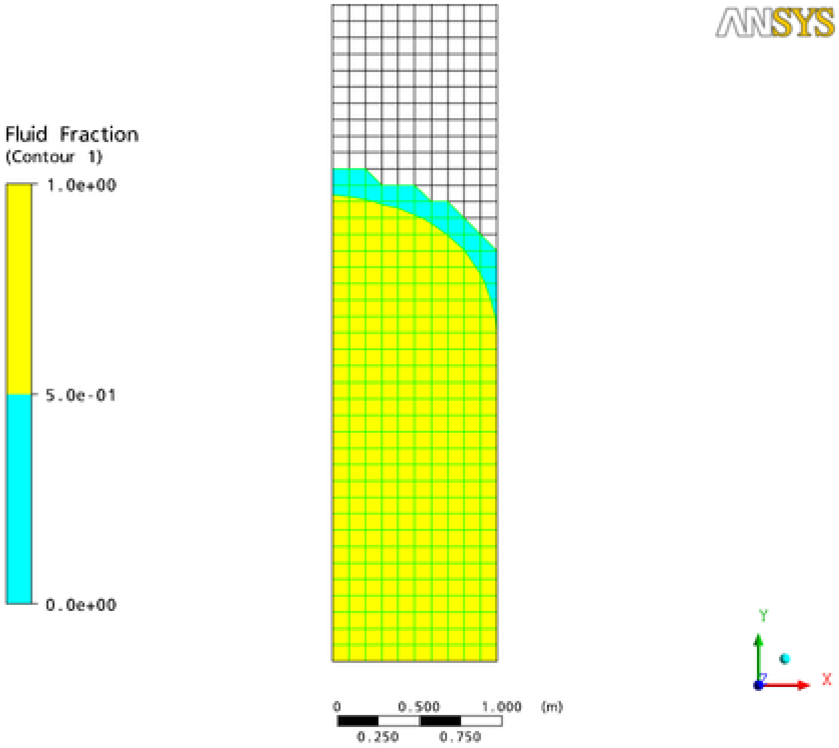
  You must define the φ variable for the inlet boundaries (i.e., boundaries where $\vec{v} \cdot \vec{n} < 0$, n representing the outward normal vector for the boundary), in order to specify that the inflow entering through this boundary is the fluid being tracked. When tracking a single fluid, the φ variable must be set to either 0 or 1 for inlets.

A highly accurate and consistent streamline-upwinding technique is used to integrate *Equation 20–1* (p. 444). The interpolation selected for φ makes use of linear subelements, to maximize numerical accuracy; this is necessary because the integration needs to be performed over a long time, and large φ variations are expected. The calculation of φ is decoupled from the flow calculation, so that calculating φ is not more expensive than computing the flow itself. In fact, the φ calculation is cheaper, because *Equation 20–1* (p. 444) is linear with regard to φ. When the flow is known, φ can be calculated directly in a single iteration.

Because of the usage of the variable φ on the domain, the previously described methodology should more appropriately be called a level set method. ANSYS POLYFLOW calls it a volume of fluid method as a reference to similar techniques in other flow codes.

Figure 20.1 (p. 445) displays the φ field distribution for a typical 2D VOF simulation that is tracking a single fluid. Note that the fluid is only determined to be present where the fluid fraction φ is above a threshold value of 0.5, which in *Figure 20.1* (p. 445) is the region colored yellow. Any region of the domain where φ is 0 (i.e., the uncolored region) or below the threshold value (i.e., the region colored blue) is not considered to contain fluid.

Figure 20.1 2D VOF Simulation Results



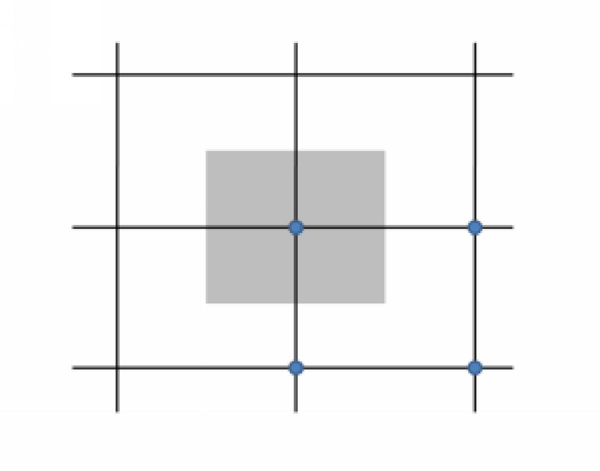
20.2.1. Volume Conservation

For VOF and related techniques, an essential accuracy criteria is the conservation of volume (or mass). When simulating an injection, for example, it is important that all of the fluid volume that enters the domain is reflected in the φ distribution. Volume loss can be an issue when a simulation involves many time steps, even with a highly accurate integration of *Equation 20–1* (p. 444). This is because the small volume errors associated with each time step can accumulate to yield a significant loss.

In order to minimize the volume loss, time is handled differently in VOF simulations than in an ordinary time-dependent scheme. First of all, ANSYS POLYFLOW attempts to minimize the number of time steps needed by varying the length of time used for the time steps (Δt). Though you set Δt for the initial time step of your VOF simulation, it is automatically determined for all subsequent time steps based on the outcome of the previous steps (see the section that follows for descriptions of various scenarios).

Another way that VOF simulations minimize volume loss is the manner in which the elapsed time is calculated. Consider the calculations that are performed at each time step: the flow field (i.e., the values for velocity/pressure and possibly other variables, such as the temperature or a chemical species concentration) is computed for a time step, followed by an integration of *Equation 20–1* (p. 444). When the new values for the φ field are available for the domain, a determination can be made for the subdomains that surround every node (e.g., the gray area in *Figure 20.2* (p. 446)) as to whether φ is above the threshold (which represents a “wet" node condition) or below the threshold (which represents a “dry" node condition). In this manner, the distribution of wet/dry nodes is known at each time step.

Figure 20.2 A Control Volume Surrounding a Node



Because it is not possible to numerically integrate discontinuous data without dissipation, the variation in the size of the wet domain is generally lower than the net volume of fluid that entered the domain through the inlets during the time step. To avoid translating this dissipation into a volume loss, ANSYS POLYFLOW calibrates the elapsed time for a successful time step based on the amount of fluid that entered the domain, rather than simply equating it with Δt. Note that this calibration is only performed when the flow rate of fluid entering the domain is nonzero; when no fluid enters (e.g., when no inlet is present), the elapsed time is set equal to the initially defined value of the time step. For this reason, you should define the initial time step conservatively.

20.2.2. Time Step Management

The following is a list of scenarios that can arise during the VOF simulation, and the resulting impact on the evaluation of Δt and/or the simulation:

- a positive net change in the number of wet nodes does not occur as a result of the time step, even though the inlet flow rate is not zero

  This is an indication that *Equation 20–1* (p. 444) has not been integrated over a period of time that is long enough to allow new nodes of the domain to reach the φ threshold, and so a larger time step needs to be applied. In this case, the time step index is not incremented, Δt is increased by 50%, and the calculations are run again to compute the φ field. This process can be repeated up to 10 times for a particular time step, after which point the simulation is terminated.

- the ordinary flow calculation returns an error (e.g., divergence is detected for a nonlinear problem)

  In this case, the time step index is not incremented and the flow calculation is attempted again using a smaller value of Δt. If no new wet nodes result and the calculation still diverges, the algorithm will eventually stop.

- the calculations for a time step yields a positive net change in the number of wet nodes

  This represents a successful outcome, and so the time step index is incremented. Having computed a new wet node flag vector (which yielded a new volume of fluid V_c in the domain) as a result of the inlet flow rate Q_{in}, a new value for Δt is calculated for the next time step as follows:

$$\Delta t = \frac{\varepsilon V_c}{Q_{in}} \quad (20\text{–}2)$$

where ε is a parameter you can set to control the accuracy of the scheme. Recommended values of ε are on the order of 0.2–0.5. Note that an extremely low value of ε can reduce the time step to such a degree that no new nodes become wet; consequently, the time step will be increased as described previously, and you then run the risk of engaging in an oscillatory "forward/backwards" marching scheme that calculates several useless iterations without improved accuracy. An inherent limitation of the VOF model is that it cannot process changes that are smaller than the element size.

Note that if the inlet flow rate Q_{in} for the time step was 0, *Equation 20–2* (p. 447) is ignored and Δt is set to the value you defined for the initial time step.

- the time step index is incremented beyond the maximum number

  In this case, the simulation is terminated. The maximum number of time steps is used to stop simulations that have values for Δt that are too low, and consequently run too long. Because of the Courant-type limitation described previously, you should expect a complete filling simulation to take several hundreds time steps and set the maximum accordingly. By default, the maximum number of time steps is set to 200.

- Δt is set to a value lower than the minimum

  The simulation is terminated when Δt falls below the minimum.

- the total outlet flow rate (integrated over all outlets) is 99% of the nonzero total inlet flow rate (i.e., the flow in the cavity is established, but almost all of the fluid is leaving the domain)

  By default, ANSYS POLYFLOW stops the simulation in such circumstances, even if the cavity has not been completely filled. In most cases, it is considered normal to terminate a VOF simulation when the flow exits the domain. The value of 99% is hardcoded in ANSYS POLYFLOW; while this value can be revised in the data file, it is recommended that you do not change it. This constraint is disabled for simulations in which there are no defined inlets, thereby allowing you to model the emptying of the cavity. You can also ignore this constraint by clicking the **Disable 'Stop When Full' criterion** menu item in the **VOF iterative parameters** menu, as described in step 8.(a)viii. in *Problem Setup* (p. 449).

- the fluid fraction variable φ exceeds the threshold value (e.g., 0.5 when tracking a single fluid) throughout the entire domain

  As stated previously, the fluid is determined to be present wherever φ is above the threshold value. Exceeding this value throughout the domain indicates that the volume of fluid has completely filled the cavity, and so the simulation is stopped by default. You can ignore this constraint by clicking the **Disable 'Stop When Full' criterion** menu item in the **VOF iterative parameters** menu, as described in step 8.(a)viii. in *Problem Setup* (p. 449).

20.2.3. Numerical Considerations

It should be noted that the methodology implemented in ANSYS POLYFLOW is different from a frequently used "two-fluid" approach, in which the region that does not contain the liquid being tracked is modeled as a low-viscosity fluid (e.g., "air") whose flow nevertheless needs to be computed. In ANSYS POLYFLOW, empty regions are simply excluded from the calculation. This has major advantages:

- The calculation time and required memory are significantly smaller at the beginning of the simulation when many nodes are still dry (and hence require no computations). Overall, this proves to be a significant saving.
- When the air viscosity is set to an extremely low value in a two-fluid approach, it can yield a Reynolds number that is high or even very high, such that it is by far more difficult (at least with velocity/pressure techniques in use in ANSYS POLYFLOW) to compute the flow of the otherwise absent material than the polymer. If you instead select a higher air viscosity, the flow of the air might interfere with the flow of the polymer by slowing it down or creating unrealistic “bubbles".

Another numerical consideration for the ANSYS POLYFLOW VOF model concerns the interpolation method. It is common for the elements to be partially filled, as a result of some of their nodes being wet and others dry (remember that the control volume around a node extends into neighboring elements). Consequently, it is not possible to use an incompressible velocity-pressure interpolation (e.g., the mini-element method) for the VOF model. Because it is necessary to accept any wet/dry node configuration within an element and a varying ratio between active velocity and pressure nodes, the LBB condition (see *Controlling the Interpolation* (p. 218)) cannot be satisfied. This is the reason why VOF simulations can only be performed using the stabilized linear velocity/constant pressure interpolation. The value of the stabilization coefficient is adjusted automatically.

20.2.4. Viscoelastic Fluids

It is possible to compute (differential) viscoelastic fluids in conjunction with the VOF model, by using a DEVSS formulation with a stabilized linear velocity/constant pressure interpolation. This section addresses the additional intrinsic difficulties that you must consider when attempting such modeling.

When a fluid model simulates a viscoelastic flow, more variables (e.g., stresses) are involved. As for any viscoelastic model, this means that more CPU time and memory is required. But the model also becomes severely nonlinear (as opposed to a generalized Newtonian flow, where the degree of nonlinearity is frequently moderate), and hence runs the risk of requiring significantly more iterations and experiencing trouble converging.

Fortunately, the VOF model is shielded from the effects of nonlinearity to some degree, owing to its time-dependent approach. It is often the case that a VOF simulation has small time steps, as a result of the Courant-type limitation; as the time steps become very small, the time derivative in the constitutive equation (e.g., *Equation 11–5* (p. 228)) dominates the other terms, including the nonlinear terms. This is the ideal case, however; while it holds true as you approach the limit of infinitely small time steps, in reality the other terms of the equation can still create difficulties with regard to nonlinearity and convergence. But if the time steps are reasonably small, you can expect that at the very least a linear term dominates. This is good news, as linear problems do converge. Therefore it can be said that from the point of view of the VOF model, the simulation of a viscoelastic flow does not pose major additional difficulties.

Important

Note that it is impractical to use a tool such as evolution with the VOF model (which again, inherently has small time steps) to handle the nonlinearity of a viscoelastic flow, as the computational expense of the resulting simulation would be extreme.

If you model a viscoelastic fluid as part of a VOF simulation, it is recommended that you take into account the inertia terms and introduce a nonvanishing density (this is not the default). This has the advantage of making the momentum equation evolutionary with respect to time (because in this case, time derivatives are maintained), and thus creates a link between the velocity fields at the current and previous time steps. You can expect this to facilitate most nonlinear problems in a VOF model, provided that the Reynolds number remains small (ideally lower than 1). This is a reasonable limitation, because the Reynolds number of a polymer flow is always low.

20.3. Problem Setup

In order to model a free surface using the VOF model, perform the following steps in ANSYS POLYDATA:

1. Create a VOF task by clicking **Create a new task**.

   Create a new task

   Then perform the following actions in the **Create a new task** menu:

   a. Specify that the VOF model is used for this task by clicking **Volume Of Fluid problem(s)**.

      Volume Of Fluid problem(s)

   b. Set up the geometry type.

   c. Click **Accept the current setup** to open the VOF task menu.

2. Create the flow sub-task in the usual manner.

   Create a sub-task

   Note the following:

   - Some of the menu items in the **Flow boundary condition along boundary <ID>** are not available for VOF tasks.
   - As a VOF simulation is transient, it is recommended that you use a nonvanishing density and take inertia terms into account (especially for viscoelastic fluids, for which inertia has a stabilizing effect). These settings can be made using the **Material Data** menu.

3. After the flow sub-task is defined, a fluid fraction transport sub-task named **F.E.M. Task Fluid Fraction Transport** is automatically created and listed in the task menu. Click this menu item to begin defining the sub-task.

   F.E.M. Task Fluid Fraction Transport

   The **F.E.M. Task Fluid Fraction Transport** menu will open.

4. Click **Domain of the sub-task** and define the domain of the fluid fraction transport sub-task in the usual way. Note that this domain represents the cavity in which the fluid will be tracked.

   Domain of the sub-task

5. Define where the initial fluid is present (i.e., the fluid already in the cavity when the simulation starts) by clicking **Material data**.

   Material data

   The **Material Data** menu will open, where you can perform the following steps.

a. To define the initial fluid fraction distribution, click **Initial Fluid Fraction**. When tracking a single fluid, the fluid is determined to be present wherever the fluid fraction is between 0.5 and 1; while this range of values is acceptable, it is recommended that you set the fluid regions to a value of 1 if possible. Similarly, the fluid is determined to be absent wherever the fluid fraction is lower than 0.5, but it is recommended that the empty regions have a value of 0. By default, the initial fluid fraction is set to 0 throughout the domain.

Initial Fluid Fraction

The **Initial Fluid Fraction** menu will open, allowing you to define the initial fluid fraction as being:

- a constant value

  You can specify that the value of the fluid fraction is the same throughout the domain by clicking **Constant**.

  Constant

  A panel will open where you can enter the fluid fraction for **New value**. Click **OK** to return to the **Initial Fluid Fraction** menu.

- a linear function

  You can specify that the fluid fraction is defined as a linear function of the coordinates ($f\ (x\ ,y\ ,z\ )$), as shown in the following equation:

$$f\ (x\ ,y\ ,z\ )\ = A + Bx + Cy + Dz \tag{20–3}$$

  Click **Linear function of coordinates**.

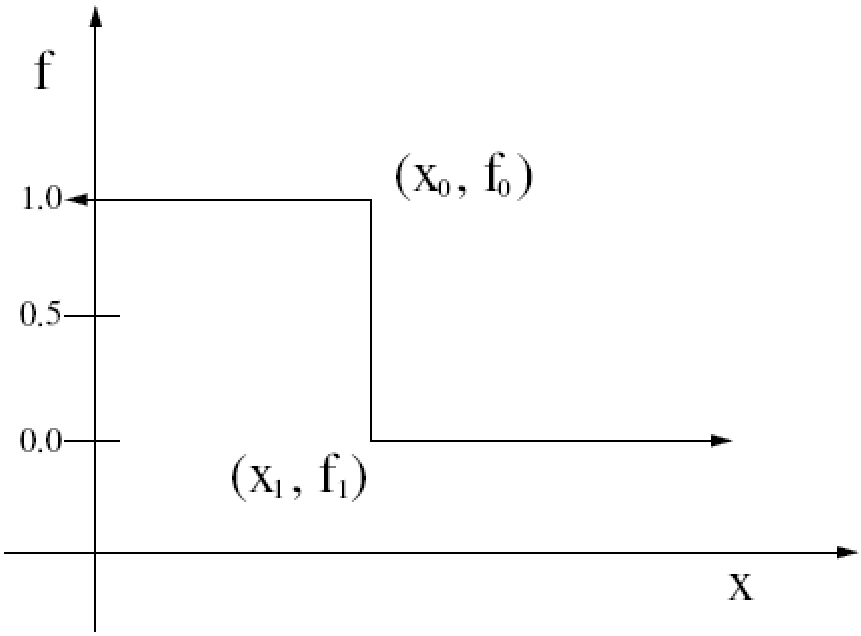
  Linear function of coordinates

  Panels will open where you can revise the **New value** for the parameters **A**, **B**, **C**, and **D** of the previous equation. Click **OK** in the final panel to return to the **Initial Fluid Fraction** menu.

- a multi-ramp function

  You can specify the fluid fraction behaves as a multi-ramp function that is defined by a sequence of coordinate pairs. For example, if you choose to vary the fluid fraction f as a function of the x coordinates, you will input pairs of coordinates $\left(x_i,f_i\right)$ (where i begins at 0 and is incremented for each additional pair). See *Figure 20.3* (p. 451) for an illustration of a multi-ramp function.

Figure 20.3 A Multi-Ramp Function for the Initial Fluid Fraction



Click the appropriate multi-ramp menu item based on the axis for which the values will vary.

Multi-ramp function of X coordinate

or

Multi-ramp function of Y coordinate

or

Multi-ramp function of Z coordinate

The **ffinit = Multi-ramp function on <direction> coordinate** menu will open, where you can perform the following:

- To input the coordinate pairs, select **Define new pairs** and enter the coordinate values in the panels that open.
- To review the coordinate pairs you defined, click **Check existing pairs**. You can delete a pair by clicking **Delete mode** and then clicking the pair. You can modify the values by clicking **Modify mode**, clicking the pair, and then entering values in the panels that open. When you have determined that the pairs are all correctly defined, click **Upper level menu** to return to the previous menu.
- To save the coordinate pair data for future use, click **Save in a Dependence Data File** and use the panel that opens to create a data file. This file can be recalled later by using the **Read in a Dependence Data File** menu item, as explained in the description that follows.
- To input the coordinate pairs using a previously written data file instead of interactively entering the pairs, click **Read a Dependence Data File** and select the file using the panel that opens. You could have created such a data file using the **Save in a Dependence Data File** menu item as described previously, or using another tool that writes the pairs of data

in the fixed (Fortran) format 2e14.7. Each data pair (e.g., x_i, f_i) is written on a single line, and fourteen digits are used for representing each datum as ±0.1234567e±89. Because the format is fixed, there is no need to use a separator. The following is a list (in order) of the components of a single datum in the e14.7 format: a sign, a 0 digit, the decimal point, seven digits (for the mantissa), the letter “e" (standing for a power of 10), a sign for the power, and two digits (for the power). For example, an entry in the file that represents (0.25,- π) would be written as +0.2500000e+00-0.31415926e+01. Subsequent data pairs are written on subsequent lines.

Click **Upper level menu** when finished defining the multi-ramp function to return to the **Initial Fluid Fraction** menu.

- interpolated values from a CSV (Excel) file

  You can interpolate fluid fraction values onto your mesh from a comma separated values (CSV) file, in a way similar to that described in *Results Interpolation Onto Another Mesh* (p. 150). The CSV file could have been created using a mesh that has different nodal locations, but the same overall geometry. Click **Map from CSV (Excel) file**.

  Map from CSV (Excel) file

  The **Map from CSV file** menu will open. Click **Modify the CSV file name** and use the panel that opens to select the file. Then click **Modify the label**, enter a case sensitive tag for **New value** in the panel that opens to identify the data, and click **OK**. Finally, click **Upper level menu** to return to the **Initial Fluid Fraction** menu

- a user-defined function (UDF)

  You can specify that the fluid fraction is defined using a UDF by clicking **User Defined Function**.

  User Defined Function

  See *User-Defined Functions (UDFs)* (p. 597) for information about UDFs.

Click **Upper level menu** to return to the **Material Data** menu.

b. To save the initial fluid fraction data defined using the **Initial Fluid Fraction** menu item for future use, click **Save in a Material Data File**.

   Save in a Material Data File

   A panel will open, which you can use to create a material data file. This file can be recalled later by using the **Read an Old Material Data File** menu item, as explained in the description that follows.

c. To input the initial fluid fraction using a previously written material data file (instead of inputting it using the **Initial Fluid Fraction** menu item), click **Read an Old Material Data File**. You could have created such a material data file using the **Save in a Material Data File** menu item, as described previously.

   Read an Old Material Data File

   A panel will open, which you can use to select the material data file.

Click **Upper level menu** to return to the **F.E.M. Task Fluid Fraction Transport** menu.

6. Define the boundary conditions for the fluid fraction transport sub-task.

   Fluid Fraction boundary conditions

   The **Fluid Fraction boundary conditions** menu will open, where you can perform the following:

   a. Select a boundary for which you want to define the boundary condition.

   b. Click **Modify** to open the **Fluid Fraction boundary condition along boundary <ID>** menu.

   c. Select the boundary condition type you want to impose:

      - Click **Inlet** if you want to specify that the boundary allows the fluid you are tracking to enter the domain. Note that all boundaries specified as an **Inlet** are taken into account when the mass balance is computed.

        Inlet

        Enter the fluid fraction for **New value** in the panel that opens and click **OK**. When tracking a single fluid, the **New value** must be set to `1` for an inlet.

        Then click **Upper level menu** to return to the previous menu.

        Note that it is possible to make your **Inlet** boundary condition variable with time by using the **EVOL** button (as described in *Problem Setup* (p. 524)), in order to simulate variable injection. You must make sure that at every time step the fluid fraction for the inlet is either 0 or 1.

      - Click **Fluid Fraction imposed** if you want to force a boundary to be either wet or dry, and thereby fine-tune a simulation to better match expected behavior. Note that all boundaries specified as a **Fluid Fraction imposed** are not taken into account when the mass balance is computed.

        Fluid Fraction imposed

        Enter the fluid fraction for **New value** in the panel that opens and click **OK**. When tracking a single fluid, the **New value** should be set to `1` for a wet boundary and `0` for a dry boundary.

        Then click **Upper level menu** to return to the previous menu.

      - Click **Wall / Symmetry** if you want to specify that the boundary acts as a wall for the fluid you are tracking, such that no fluid enters or exits (which is the equivalent of a symmetry boundary).

        Wall / Symmetry

      - Click **Outlet** if you want to specify that the boundary allows the fluid you are tracking to exit the domain. Note that all boundaries specified as an **Outlet** are taken into account when the mass balance is computed.

        Outlet

- Click **Source of connected condition** to specify that the boundary acts as a “source" boundary in a pair of non-conformal boundaries that are to be connected. It is recommended that the source be the coarser of the two meshes. Non-conformal boundary conditions are described in *Non-Conformal Boundaries* (p. 190).

  Source of connected condition

- Click **Target of connected condition** to specify that the boundary acts as a “target" boundary in a pair of non-conformal boundaries that are to be connected. This menu item is only available if you have defined a source boundary, as described previously. It is recommended that the target be the finer of the two meshes. Non-conformal boundary conditions are described in *Non-Conformal Boundaries* (p. 190).

  Target of connected condition

  The **Target of connected cond. along boundary <ID>** menu will open, where you must select a boundary from the list and click **Select**. Then the **Options for connected boundaries** menu will open, where you can set the parameters for the non-conformal boundary.

  The inputs for non-conformal fluid fraction boundary conditions are the same as those for flow boundary conditions, as described in *Connecting Non-Conformal Boundaries* (p. 191), except for the numerical parameters. The first two numerical parameters (element dilatation and amplitude of volume generation) are the same as for flow boundary conditions, but the stabilization factor for flow conditions is replaced by a smoothing factor for fluid fraction conditions. The smoothing factor controls the fluid fraction transport tangential to the surface of the connected boundaries. Ideally, there should be no fluid fraction transport within the connected boundaries, but setting this parameter to zero can lead to numerical problems (e.g., zero pivot) or instabilities. It is therefore recommended to set the smoothing factor to a very small nonzero value.

  > **Important**
  >
  > Changing the value of the smoothing factor from the default value is not recommended, except on the advice of your support engineer.

Click **Upper level menu** to return to the **Fluid Fraction boundary conditions** menu.

d. Repeat steps 6.(a)–6.(c) for each additional boundary you want to define.

e. Click **Upper level menu** to return to the **F.E.M. Task Fluid Fraction Transport** menu.

7. Click **Upper level menu** to return to the VOF task page.

8. Modify the numerical parameters for the VOF task.

   Numerical parameters

   The **Numerical parameters** menu will open.

   a. Define the iterative parameters for the VOF task, by clicking **Modify the VOF iterative parameters**.

      Modify the VOF iterative parameters

The **VOF iterative parameters** menu will open.

i. Specify the time at which the time-marching procedure begins.

   Modify the initial time value

   Enter the initial time for **New value** in the panel that opens and click **OK**.

ii. Specify the time by which the solution procedure must stop.

   Modify the upper time limit

   Enter the latest permissible end time for **New value** in the panel that opens and click **OK**.

iii. Define the value of Δt for the initial time step.

   Modify the initial value of the time-step

   Enter the initial Δt for **New value** in the panel that opens and click **OK**. Note that the elapsed time for a time step is set to this value when the inlet flow rate is 0, so it is recommended that you enter a conservative value.

iv. Define the minimum allowable value of Δt for any time step.

   Modify the min value of the time-step

   Enter the minimum Δt for **New value** in the panel that opens and click **OK**.

v. Define the maximum allowable value of Δt for any time step.

   Modify the max value of the time-step

   Enter the maximum Δt for **New value** in the panel that opens and click **OK**.

vi. Define the accuracy of the VOF scheme (ε in *Equation 20–2* (p. 447)).

   Modify the accuracy of the VOF scheme

   Enter the accuracy for **New value** in the panel that opens and click **OK**. It is recommended that the accuracy be on the order of 0.2–0.5.

vii. Define the maximum number of successful (i.e., converged) steps. The computation will be stopped when this number is reached, even if the upper time limit has not yet been reached. Note that you can always restart the scheme from the results file in order to continue the time-marching scheme without losing the benefit of the previous solutions. See *Starting an ANSYS POLYFLOW Calculation from an Existing Results File* (p. 109) for more information on these file types.

   Modify the max number of successful steps

   Enter the maximum number of steps for **New value** in the panel that opens and click **OK**.

viii. If you would like the simulation to run when the flow balance is zero (i.e., the total outlet flow rate is over 99% of the nonzero total inlet flow rate) or when the entire domain exceeds the fluid fraction threshold (i.e., the cavity is filled with fluid), you can disable the default constraint that limits this behavior by clicking the **Disable 'Stop When Full' criterion** menu item.

   Disable 'Stop When Full' criterion

   This menu option then becomes **Enable 'Stop When Full' criterion**, so that you can toggle it back if necessary.

ix. Select the method of integration by clicking one of the following:

   - **Use of the O-order method**
   - **Use of the implicit Euler method**
   - **Use of the Galerkin method**
   - **Use of the Crank-Nicolson method**

   See *Integration Methods* (p. 540) for details about the previous integration methods.

x. Click **Upper level menu** to return to the **Numerical parameters** menu.

b. Click **Upper level menu** to return to the VOF task menu.

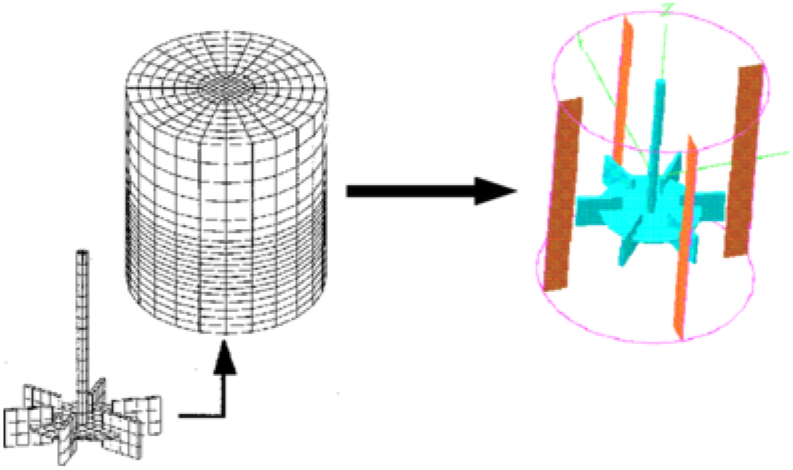
Chapter 21: Flows with Internal Moving Parts

Information on the modeling of flows with internal moving parts is presented in this chapter.

21.1. Introduction

ANSYS POLYFLOW incorporates a technique known as mesh superposition to simulate transient flows with internal moving parts. Applications include industrial processes such as stirring tanks, twin screw extruders, and pistons. An example is shown in *Figure 21.1* (p. 457).

Figure 21.1 Example of Mesh Superposition Technique



21.1.1. Advantages of the Mesh Superposition Technique

The mesh superposition technique has three major advantages:

- Mesh generation is much simpler since no complex intermeshing region must be generated.
- It is possible to define a library of moving parts, and to combine them with ANSYS POLYFUSE to generate new meshes for new simulations.
- The method is robust, since no remeshing algorithms are needed.

21.1.2. Limitations of the Mesh Superposition Technique

The mesh superposition technique also has several limitations:

- It can be used only for 2D planar and 3D models.
- Currently, it can be applied only to generalized Newtonian flow.

- The detailed variation of the velocity in the neighborhood of the moving part is not well resolved.
- As the physical boundaries do not match finite-element limits, the mass conservation equation $\nabla \cdot \mathbf{v} = 0$ cannot be satisfied in every element; as a result, you may see some limited fluid leakage.

21.2. Theory

The modeling of internal moving parts requires the modification of the Navier-Stokes equations, the mass conservation equation, and the energy equation. For simplicity, the discussion here assumes that there is a single moving part, although this is by no means a limitation.

21.2.1. Navier-Stokes Equations

The Navier-Stokes equations are modified to become

$$H(\mathbf{v} - \bar{\mathbf{v}}) + (1 - H)(-\nabla p + \nabla \cdot \mathbf{T} + \rho \mathbf{g} - \rho \mathbf{a}) = \mathbf{0} \tag{21–1}$$

where
H is a step function
$\mathbf{v}$ is the velocity
$\bar{\mathbf{v}}$ is the local velocity of the moving part
p is the pressure
$\mathbf{T}$ is the extra-stress tensor
$\rho \mathbf{g}$ is the volume force
$\rho \mathbf{a}$ is the acceleration term

For a generalized Newtonian fluid, the extra-stress tensor is defined to be

$$\mathbf{T} = 2\eta(\dot{\gamma}, T)\mathbf{D} \tag{21–2}$$

where
η is the viscosity
$\dot{\gamma}$ is the shear rate
T is the temperature
$\mathbf{D}$ is the rate-of-deformation tensor

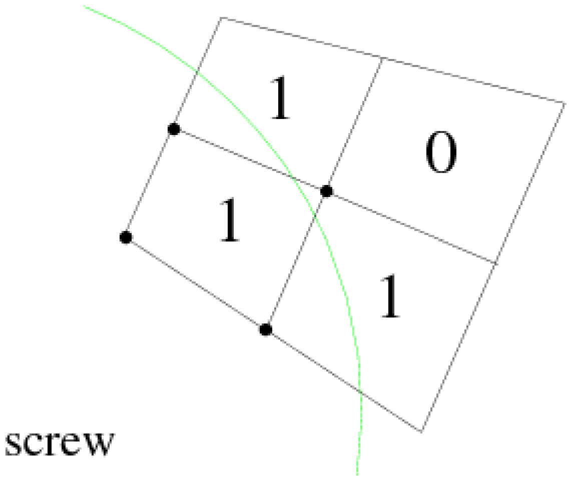
Equation 21–1 (p. 458) is then discretized for each node of the velocity field. For node i (at location $\mathbf{x}$), if it is outside the moving part, then H is equal to 0 and the usual Navier-Stokes equations are used. Otherwise, H is set to 1, and *Equation 21–1* (p. 458) degenerates into

$$\mathbf{v} = \bar{\mathbf{v}} \tag{21–3}$$

in order to impose the local velocity $\bar{\mathbf{v}}(\mathbf{x})$ of the moving part.

More specifically, before solving the Navier-Stokes equations, the “inside" field H is calculated for the flow domain. This field varies between 0 and 1. A subelement that is overlapped by the moving part has a value of $H = 1$, and a subelement outside the moving part has a value of $H = 0$. A node i (at location $\mathbf{x}$) is considered to be inside the moving part (i.e., $H = 1$) if $H(\mathbf{x})$ is greater than a threshold value. The threshold value is usually equal to 0.6, which indicates that more than half of the subelements neighboring the node are overlapped by the moving part. *Figure 21.2* (p. 459) shows a 2D finite element divided into 4 subelements. The subelements that are overlapped by the moving part are marked with a 1, and those that are outside the moving part are marked with a 0. The nodes for which $H = 1$ are indicated by filled-in circles.

Figure 21.2 "Inside" Field for a 2D Finite Element



The Navier-Stokes equation explained above considers a full stick condition along the borders of the moving part. To include a slip condition along the borders, a continuous linear inside field I (as opposed to inside field H, which defines constant values for the subelements) is evaluated at each node of the flow domain. I can take the following values:

- I=1, if the current velocity node is in the moving part
- I=0, if the node is outside the moving part

The gradient of I (∇I) has a nonzero value in a small zone of the flow domain surrounding the border of the moving part. This zone becomes thinner as the flow domain mesh is refined. In this zone, the local normal direction to the moving part boundary is evaluated.

With the slip condition, the Navier-Stokes equation modifies to become:

$$-\nabla p + \nabla \cdot \mathbf{T} + \rho \mathbf{g} - \rho \mathbf{a} + \mathbf{n}\mathbf{k}(\mathbf{v} - \bar{\mathbf{v}}) = \mathbf{0} \tag{21–4}$$

where

$\mathbf{k}(\mathbf{v}')$ is the slip law function of the local relative velocity v' at the moving part boundary

$\bar{\mathbf{v}}$ is the local velocity of the moving part

$\mathbf{n}$ is the local normal direction to the moving part

The additional term, ($\mathbf{n}\mathbf{k}(\mathbf{v} - \bar{\mathbf{v}})$), imposes a 'volume' force for all the velocity components at the nodes close to the border of the moving part. In its usual form, the $\mathbf{k}(\mathbf{v}')$ term involves only the tangential component of velocity for the nodes near the boundary. Hence, the velocities should be imposed such that they do not enter the moving part. This can be done using a penalty technique, where, for each node where ∇I is nonzero, a $((\mathbf{v} - \bar{\mathbf{v}}) \cdot \mathbf{n}) = \mathbf{0}$ condition is imposed.

In the calculation, all the elements of the flow domain completely overlapped by the moving part and not falling on the border are removed.

21.2.2. Mass Conservation Equation

In order to be able to calculate a physically meaningful pressure even in the zones where geometrical penetration occurs (i.e., to avoid pressure modes associated with the locking of the element), the mass conservation equation is modified to become

$$\nabla \cdot \mathbf{v} + \frac{\beta}{\eta} \Delta p = 0 \tag{21–5}$$

where β is a relative compression factor, and η is the local viscosity.

The relative compression factor β is a key aspect of the mesh superposition technique. If there are pressure peaks in regions where a large number of geometrical constraints exist, then the fluid cannot be considered incompressible. To prevent these pressure peaks, the mass conservation equation has been modified so that the fluid is slightly compressible.

The loss or gain of fluid volume per unit time is linked to the Laplacian of the pressure through the relative compression factor. It is absolutely essential to select the value of this factor carefully. If this factor is too small, pressure peaks will appear in tiny contact regions, especially when the mesh is so coarse that one element exists between the boundary and contact regions.

When the factor is too large, the fluid is unphysically compressible and all pressure gradients will be smoothed out, leading to an unphysically low pressure prediction. In ANSYS POLYDATA, the default value of 0.01 has been shown to be the best choice for this factor.

Since a constant pressure per element is assumed, *Equation 21–5* (p. 460) is discretized for each element of the flow domain.

21.2.3. Energy Equation

If the simulation is nonisothermal, the energy equation is modified to be

$$\begin{aligned} 0 = (1-H) &\left(\rho_f c_{pf} \frac{DT}{Dt} - r_f - \mathbf{T} : \nabla \mathbf{v} - \nabla \cdot \left(k_f \nabla T \right) \right) \\ &+ H \left(\rho_s c_{ps} \frac{DT}{Dt} - r_s - \nabla \cdot (k_s \nabla T) \right) \end{aligned} \tag{21–6}$$

where

ρ_f	is the fluid density
c_{pf}	is the fluid heat capacity
r_f	is the fluid heat source
k_f	is the fluid thermal conductivity
ρ_s	is the density of the moving part

c_{ps} is the heat capacity of the moving part

r_s is the heat source of the moving part

k_s is the thermal conductivity of the moving part

Equation 21–6 (p. 460) is discretized for each node of the temperature field. For each node i (at location $\mathbf{x}$), if it is outside the moving part, the step function H is equal to 0 and the energy equation with the fluid parameters if used. Otherwise, H is equal to 1 and the energy equation with the moving part parameters is used.

21.2.4. Interpolation

The following interpolants are used by default for flows with moving parts:

- Velocities are quadratic in 2D and mini-element in 3D. Note that linear velocities can be used as well, in both 2D and 3D.
- Pressure is constant per element in 2D and 3D.
- Temperatures are quadratic in 2D and linear in 3D.

ANSYS POLYDATA will select the appropriate method(s) automatically when you define a moving part. Only linear velocities are available if the slip model is imposed along moving parts.

21.2.5. Transient Moving Part Velocities

The motion of the moving parts can be both transient and complex. Any type of complex motion can be specified using a user-defined function. Using the specified translational and angular velocities, ANSYS POLYFLOW integrates the translational and angular positions with an implicit Euler scheme. At each time step, the new position is computed by a displacement (translation and rotation) of the moving part in its reference configuration.

21.2.5.1. Description of Motion

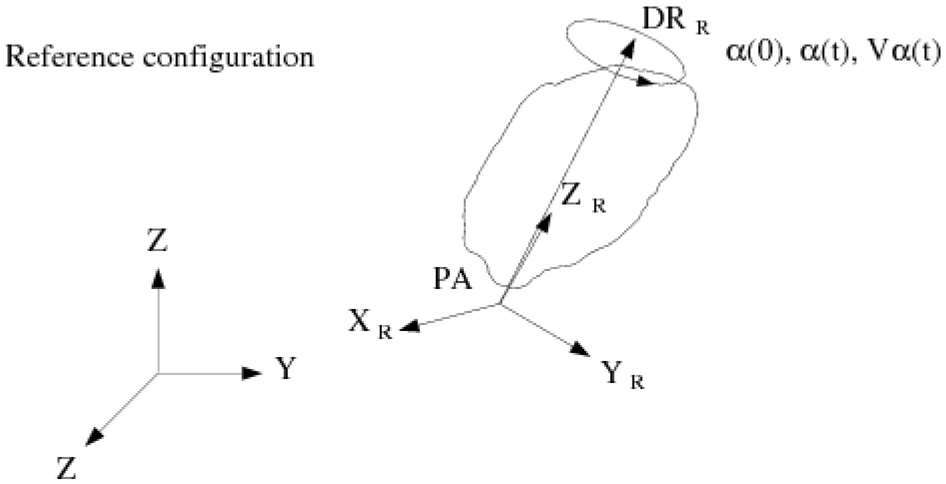
The motion is based on the reference position of the moving part, which is the initial position in the original mesh file. A reference frame ($X_{\mathrm{R}}, Y_{\mathrm{R}}, Z_{\mathrm{R}}$) is associated with the reference position, and the rotation motion is defined. The orientation of the axis of rotation can be changed by specification of a rotation around the Z axis, followed by another rotation around the new Y axis. Then a translation can be applied.

The motion is defined in three steps:

1. Define the rotation in the reference configuration (*Figure 21.3* (p. 462)):
   - Direction of the rotation axis in the reference frame: DR_{R}
   - Point of the rotation axis (i.e., origin of the ($X_{\mathrm{R}}, Y_{\mathrm{R}}, Z_{\mathrm{R}}$) reference frame): PA
   - Angular velocity: $V_{\alpha}(t)$
   - Initial rotation angle: $\alpha\ (0)$
   - Angle of rotation: $\alpha\ (t)$ given by

$$\alpha(t) = \alpha(0) + \int V_\alpha(t)\, dt \tag{21–7}$$

Figure 21.3 Description of Rotational Motion in the Reference Configuration



2. Define the change in direction of the axis of rotation, DR(t) by a rotation of angle $\theta(t)$ around the Z axis, followed by a rotation of angle $\phi(t)$ around the new Y axis (*Figure 21.4* (p. 463)):

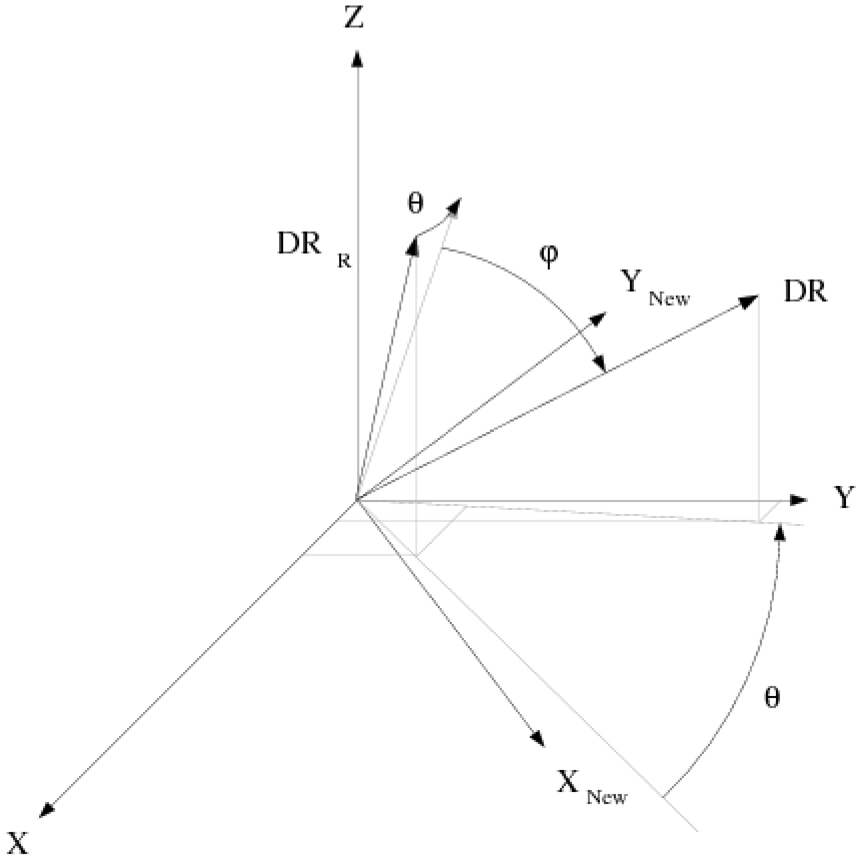
   - Initial angle of rotation around the Z axis: $\theta(0)$
   - Angular velocity around the Z axis: $V_\theta(t)$
   - Angle of rotation around the Z axis: $\theta(t)$ given by

   $$\theta(t) = \theta(0) + \int V_\theta(t)\, dt \tag{21–8}$$

   - Initial angle of rotation around the new Y axis: $\phi(0)$
   - Angular velocity around the new Y axis: $V_\phi(t)$
   - Angle of rotation around the new Y axis: $\phi(t)$ given by

$$\phi(t) = \phi(0) + \int V_{\phi}(t)\,dt \qquad (21–9)$$

Figure 21.4 Description of the Change of Orientation of the Rotation Axis

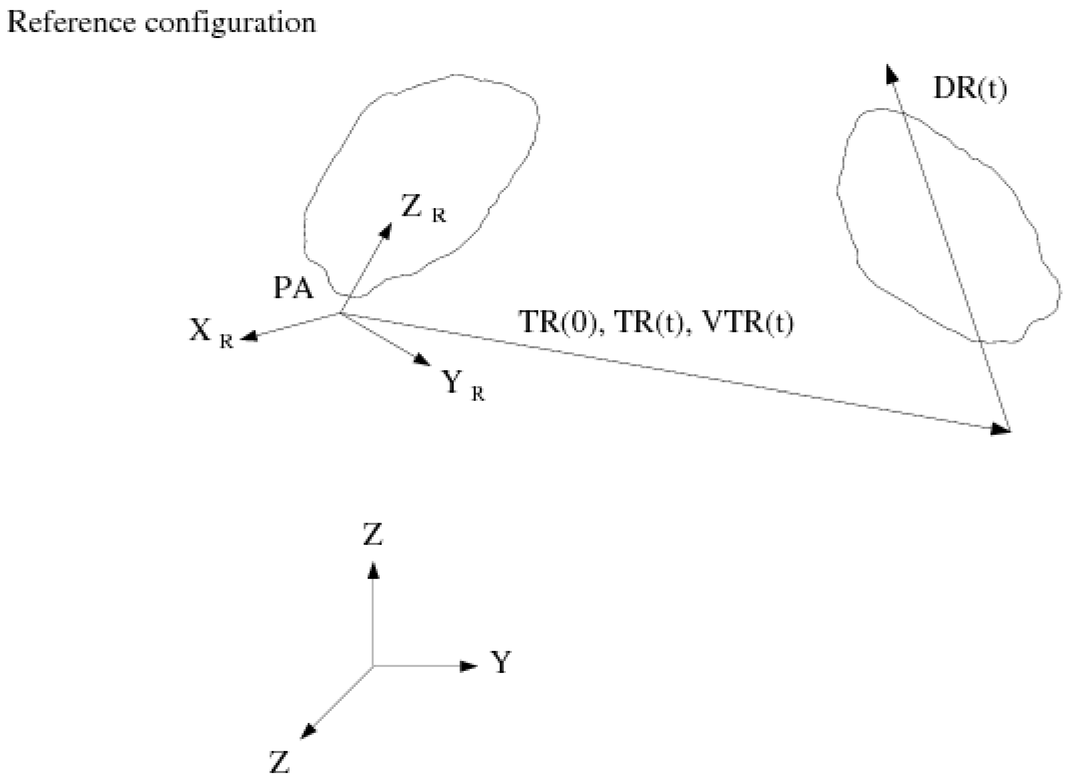


3. Define the translation (*Figure 21.5* (p. 464)):

   - Initial translation: TR(0)
   - Translation velocity: $V_{\mathrm{TR}}(t)$
   - Translation vector: TR(t) given by

$$\mathrm{TR}\,(t) = \mathrm{TR}\,(0) + \int V_{\mathrm{TR}}\,(t)\,dt \tag{21–10}$$

Figure 21.5 Description of Translation



The following quantities are specified by you during the problem setup in ANSYS POLYDATA:

- Direction of the rotation axis in the reference configuration, DR_{R}
- Point of the rotation axis, PA
- Angular velocity, $V_{\alpha}\,(t)$, in RPM
- Initial rotation angle, $\alpha\,(0)$, in degrees
- Angles $\theta\,(0)$ and $\phi\,(0)$, in degrees
- Angular velocities $V_{\theta}\,(t)$ and $V_{\phi}\,(t)$, in RPM
- Initial translation, TR(0)
- Translation velocity, $V_{\mathrm{TR}}\,(t)$

The positions (translation TR(*t*) and rotation angles $\alpha\,(t)$, $\theta\,(t)$, and $\phi\,(t)$) are integrated by an implicit Euler scheme:

$$\begin{aligned} \mathrm{TR}(t+\delta t) &= \mathrm{TR}(t) + V_{\mathrm{TR}}(t+\delta t)\,\delta t \\ \alpha(t+\delta t) &= \alpha(t) + V_{\alpha}(t+\delta t)\,\delta t \\ \theta(t+\delta t) &= \theta(t) + V_{\theta}(t+\delta t)\,\delta t \\ \phi(t+\delta t) &= \phi(t) + V_{\phi}(t+\delta t)\,\delta t \end{aligned} \tag{21–11}$$

This integration produces a vector of translation and a matrix of rotation. In order to guarantee the accuracy of the Euler scheme, each time step is divided into substeps of size on the order of 10^{-5}. The maximum number of substeps is 10,000. At each time step, the new position of the moving part is obtained by applying a translation followed by a rotation of the reference position. This technique avoids error accumulation that can lead to deformation of the moving part. Note that the substeps are used only to integrate the position (translation, angles of rotation), while the position itself is computed only once per time step.

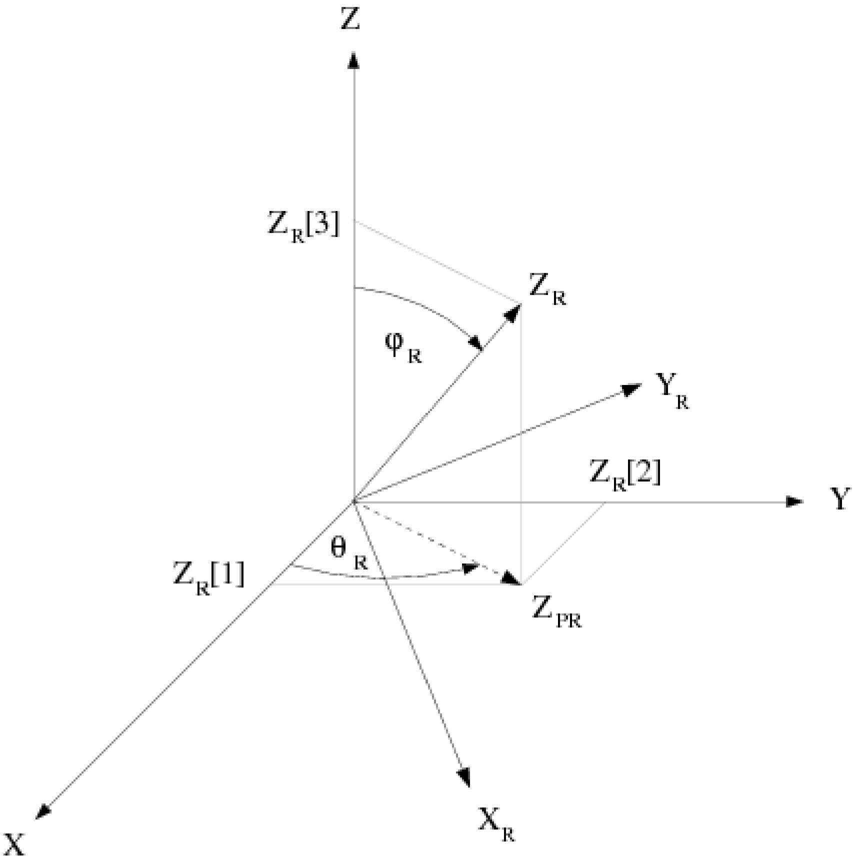
21.2.5.2. Computation of the New Position

In order to compute the new position, the translation vector and matrix of rotation are needed. In the reference configuration, the reference frame ($X_{\mathrm{R}}, Y_{\mathrm{R}}, Z_{\mathrm{R}}$) is defined such that the Z_{R} axis is aligned with the direction of the rotation axis DR$_{\mathrm{R}}$. The transformation matrix from (X, Y, Z) to ($X_{\mathrm{R}}, Y_{\mathrm{R}}, Z_{\mathrm{R}}$) is denoted by $\mathbf{M}_{\mathrm{R}}$ in the following relation:

$$\mathbf{X}_{\mathrm{R}} = \mathbf{M}_{\mathrm{R}} \cdot \mathbf{X} \tag{21–12}$$

$\mathbf{M}_{\mathrm{R}}$ is composed of two rotations: the first around the Z axis with an angle of θ_{R}, and the second around the Y_{R} axis with an angle of ϕ_{R}. *Figure 21.6* (p. 466) shows the reference frames and how the rotations are defined.

Figure 21.6 Transformation Between the (X, Y, Z) and (X_R, Y_R, Z_R) Frames



The transformation matrix for the first rotation is given by

$$\mathbf{M}_{\theta_R} = \begin{bmatrix} \cos(\theta_R) & \sin(\theta_R) & 0 \\ -\sin(\theta_R) & \cos(\theta_R) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{21–13}$$

and the transformation matrix for the second rotation is given by

$$\mathbf{M}_{\phi_R} = \begin{bmatrix} \cos\left(\phi_R\right) & 0 & -\sin\left(\phi_R\right) \\ 0 & 1 & 0 \\ \sin\left(\phi_R\right) & 0 & \cos\left(\phi_R\right) \end{bmatrix} \tag{21–14}$$

Thus, the transformation matrix for the combination of both rotations is given by

$$\mathbf{M}_{\mathrm{R}} = \mathbf{M}_{\phi_{\mathrm{R}}} \cdot \mathbf{M}_{\theta_{\mathrm{R}}} = \begin{bmatrix} \cos\left(\phi_{\mathrm{R}}\right)\cos\left(\theta_{\mathrm{R}}\right) & \cos\left(\phi_{\mathrm{R}}\right)\sin\left(\theta_{\mathrm{R}}\right) & -\sin\left(\phi_{\mathrm{R}}\right) \\ -\sin\left(\theta_{\mathrm{R}}\right) & \cos\left(\theta_{\mathrm{R}}\right) & 0 \\ \sin\left(\phi_{\mathrm{R}}\right)\cos\left(\theta_{\mathrm{R}}\right) & \sin\left(\phi_{\mathrm{R}}\right)\sin\left(\theta_{\mathrm{R}}\right) & \cos\left(\phi_{\mathrm{R}}\right) \end{bmatrix} \tag{21–15}$$

In *Equation 21–13* (p. 466), *Equation 21–14* (p. 466), and *Equation 21–15* (p. 467), the cosine and sine of the angles are given by

$$\begin{aligned} \cos\left(\phi_{\mathrm{R}}\right) &= \mathbf{Z}\cdot\mathbf{Z}_{\mathrm{R}} &&= Z_{\mathrm{R}}[3] \\ \sin\left(\phi_{\mathrm{R}}\right) &= \|\mathbf{Z}\times\mathbf{Z}_{\mathrm{R}}\| &&= \sqrt{\left(Z_{\mathrm{R}}[1]\right)^2 + \left(Z_{\mathrm{R}}[2]\right)^2} \\ \cos\left(\theta_{\mathrm{R}}\right) &= \frac{Z_{\mathrm{R}}[1]}{\sin\left(\phi_{\mathrm{R}}\right)} \\ \sin\left(\theta_{\mathrm{R}}\right) &= \frac{Z_{\mathrm{R}}[2]}{\sin\left(\phi_{\mathrm{R}}\right)} \end{aligned} \tag{21–16}$$

By definition, $\sin\left(\phi_{\mathrm{R}}\right)$ is always positive.

Z_{PR} in *Figure 21.6* (p. 466) is the projection of Z_{R} onto the plane XY. If this projection is a point, it is assumed that $\theta_{\mathrm{R}} = 0$.

Similarly, the transformation matrix $\mathbf{M}$ to align the reference axis of rotation DR$_{\mathrm{R}}$ with the current direction of the rotation axis DR(*t*) is given by

$$\mathbf{M} = \mathbf{M}_{\phi} \cdot \mathbf{M}_{\theta} = \begin{bmatrix} \cos[\phi(t)]\cos[\theta(t)] & \cos[\phi(t)]\sin[\theta(t)] & -\sin[\phi(t)] \\ -\sin[\theta(t)] & \cos[\theta(t)] & 0 \\ \sin[\phi(t)]\cos[\theta(t)] & \sin[\phi(t)]\sin[\theta(t)] & \cos[\phi(t)] \end{bmatrix} \tag{21–17}$$

The matrix of rotation in the reference frame is denoted by $\mathbf{R}_{\mathrm{R}}$ and is given by

$$\mathbf{R}_{\mathrm{R}} = \begin{bmatrix} \cos\,[\alpha(t)] & -\sin\,[\alpha(t)] & 0 \\ \sin\,[\alpha(t)] & \cos\,[\alpha(t)] & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{21–18}$$

To obtain the current position of a point P with the coordinates $\mathbf{X}$, the following transformations are applied:

1. Translate $-\mathbf{PA}$ to move the point in the reference frame:

$$\mathbf{X}_1 = \mathbf{X} - \mathbf{PA} \tag{21–19}$$

2. Change the frame from (X, Y, Z) to $(X_{\mathrm{R}}, Y_{\mathrm{R}}, Z_{\mathrm{R}})$:

$$\mathbf{X}_2 = \mathbf{M}_{\mathrm{R}} \cdot \mathbf{X}_1 = \mathbf{M}_{\mathrm{R}} \cdot (\mathbf{X} - \mathbf{PA}) \tag{21–20}$$

3. Apply the rotation angle $\alpha(t)$ in the reference frame:

$$\mathbf{X}_3 = \mathbf{R}_{\mathrm{R}} \cdot \mathbf{X}_2 = \mathbf{R}_{\mathrm{R}} \cdot \mathbf{M}_{\mathrm{R}} \cdot (\mathbf{X} - \mathbf{PA}) \tag{21–21}$$

4. Change the frame from $(X_{\mathrm{R}}, Y_{\mathrm{R}}, Z_{\mathrm{R}})$ to (X, Y, Z):

$$\mathbf{X}_4 = \mathbf{M}_{\mathrm{R}}^T \cdot \mathbf{X}_3 = \mathbf{M}_{\mathrm{R}}^T \cdot \mathbf{R}_{\mathrm{R}} \cdot \mathbf{M}_{\mathrm{R}} \cdot (\mathbf{X} - \mathbf{PA}) \tag{21–22}$$

5. Rotate around the Z axis by $\theta(t)$, and around the new Y axis by $\phi(t)$:

$$\mathbf{X}_5 = \mathbf{M}^T \cdot \mathbf{X}_4 = \mathbf{M}^T \cdot \mathbf{M}_{\mathrm{R}}^T \cdot \mathbf{R}_{\mathrm{R}} \cdot \mathbf{M}_{\mathrm{R}} \cdot (\mathbf{X} - \mathbf{PA}) \tag{21–23}$$

6. Translate PA:

$$\mathbf{X}_6 = \mathbf{PA} + \mathbf{X}_5 = \mathbf{PA} + \mathbf{M}^T \cdot \mathbf{M}_{\mathrm{R}}^T \cdot \mathbf{R}_{\mathrm{R}} \cdot \mathbf{M}_{\mathrm{R}} \cdot (\mathbf{X} - \mathbf{PA}) \tag{21–24}$$

7. Take into account the translation $\mathbf{TR}$:

$$\mathbf{X}_7 = \mathbf{TR}(t) + \mathbf{PA} + \mathbf{X}_6 = \mathbf{TR}(t) + \mathbf{PA} + \mathbf{M}^T \cdot \mathbf{M}_{\mathrm{R}}^T \cdot \mathbf{R}_{\mathrm{R}} \cdot \mathbf{M}_{\mathrm{R}} \cdot (\mathbf{X} - \mathbf{PA}) \tag{21–25}$$

21.2.5.3. Computation of the Velocity

For a point in the domain with coordinates $\mathbf{X}$, its velocity is composed of the translational and rotational velocities:

- Translational velocity: $V_{\mathrm{TR}}(t)$
- Velocity $VV_\theta(t)$ due to the rotation of $\theta(t)$ around the Z axis:

$$\mathbf{VV}_\theta = V_\theta(\mathbf{Z} \times (\mathbf{X} - \mathbf{PA} - \mathbf{TR})) \tag{21–26}$$

  where $\mathbf{Z}$ is the vector (0.0, 0.0, 1.0)

- Velocity $VV_\phi(t)$ due to the rotation of $\phi(t)$ around the new Y axis:

$$\mathbf{VV}_\phi = V_\phi\,(\mathbf{Y}_{\text{new}} \times (\mathbf{X} - \mathbf{PA} - \mathbf{TR})\,) \tag{21–27}$$

where $\mathbf{Y}_{\text{new}}$ is the vector $(-\sin[\theta(t)], \cos[\theta(t)], 0.0)$

- Velocity $VV_\alpha(t)$ due to the rotation of $\alpha(t)$ around $\mathbf{DR}(t)$:

$$\mathbf{VV}_\alpha = V_\alpha\left(\mathbf{M}^T \cdot \mathbf{DR}_{\text{R}} \times (\mathbf{X} - \mathbf{PA} - \mathbf{TR})\right) \tag{21–28}$$

where $\mathbf{Y}_{\text{new}}$ is the vector $(-\sin[\theta(t)], \cos[\theta(t)], 0.0)$

The total velocity is given by

$$\mathbf{V} = \mathbf{V}_{\text{TR}} + \mathbf{VV}_\theta + \mathbf{VV}_\phi + \mathbf{VV}_\alpha \tag{21–29}$$

21.3. User Inputs

21.3.1. Mesh Considerations

You can generate separate mesh files for the flow region and for each moving part. A geometric technique will be used to apply appropriate boundary conditions when penetration of the rigid part into the fluid region occurs. Element conformity and node matching is not required.

After generating these mesh files, use ANSYS POLYFUSE to combine them into a single new mesh representing the geometry of the problem you want to solve. Specify a different subdomain for each distinct part, moving or stationary. See *Combining Meshes with ANSYS POLYFUSE* (p. 151) for information about using ANSYS POLYFUSE to combine meshes.

21.3.2. Setting Up Your Problem in ANSYS POLYDATA

After creating an appropriate mesh in ANSYS POLYFUSE, follow these steps to define your problem in ANSYS POLYDATA:

1. Read in the combined mesh.

   Read a mesh file

2. Define a time-dependent task for your problem.

   Create a new task

3. Define a sub-task.

   Create a sub-task

   a. Select the appropriate problem type.

Important

The mesh superposition technique can be used only for a generalized Newtonian flow problem.

Generalized Newtonian isothermal flow problem

Generalized Newtonian non-isothermal flow problem

b. When prompted, specify a name for the sub-task.

4. Specify the stationary portion of the region where the sub-task applies.

Domain of the sub-task

Here, you should choose the subdomain(s) representing the *stationary* part of the domain. Each of the moving parts will be defined separately, as described in the next step.

5. Define the moving part(s).

Define moving parts

For each moving part, follow the steps below:

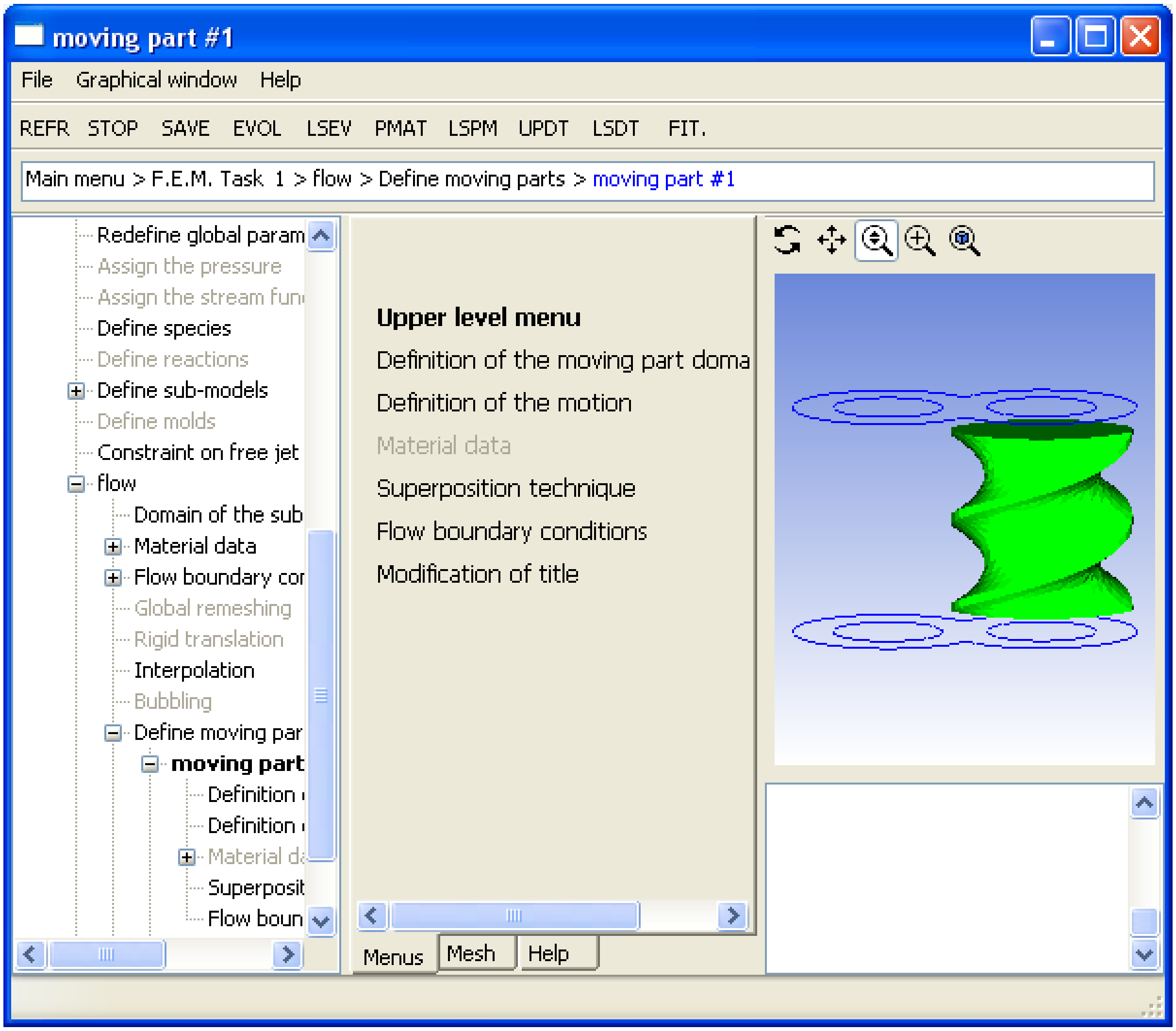
a. Create a new moving part.

Creation of a new moving part

If your mesh is 2D, ANSYS POLYDATA will inform you that quadratic coordinates are needed, and that the modification to quadratic coordinates has been done automatically. ANSYS POLYDATA will define the right interpolant for velocity, pressure, and temperature when a moving part has been defined. This is taken care of properly in 3D as well, although no message appears.

ANSYS POLYDATA will inform you that the tolerance for time marching has been increased to 10000. This is done in order to keep the time step Δt constant, as discussed in *Time-Dependent Parameters* (p. 473) . You can simply click **OK** to continue.

Figure 21.7 Menu for a Moving Part



b. Specify the subdomain(s) representing the moving part.

Definition of the moving part domain

c. Define the initial position and motion of the moving part. In general, the position of the moving part in the mesh file does not correspond to its initial position for the simulation, so you will need to specify an initial position as well as the motion. The simplest case involves rotation about a fixed axis with the reference position corresponding to the initial position. A more complex case involves rotation about a moving axis with a translation, with the initial position different from the reference position. See *Guidelines for Problems with Transient Velocities* (p. 473) for guidelines on setting up the motion when the axis of rotation is moving.

Definition of the motion

i. To specify rotation of the moving part, choose **Modify the point of local rotation axis**, **Modify the orientation of local rotation axis**, and **Modify the angular velocity**. The angular velocity can be dependent upon the evolution parameter S. (See *Evolution* (p. 517) for details about evolution.)

Important

Note that angular velocity must be entered here in RPM. (In other places, the angular velocity is given in rad/s.)

ii. To specify translation of the moving part, choose **Modify the translation velocity**. The translation velocity can be dependent upon the evolution parameter S. (See *Evolution* (p. 517) for details about evolution.)

iii. If the reference position of the moving part is not the initial position, specify the initial position by choosing **Modify the initial translation vector** and/or **Modify the initial angle of rotation**. If you specify an initial angle, it must be in degrees.

iv. If the axis of rotation is moving, specify its initial position (angles θ and ϕ) and the angular velocities (V_θ and V_ϕ) about the Z and Y axes, respectively, by choosing **Modify the initial theta angle (Z axis)**, **Modify the initial phi angle (Y axis)**, **Modify angular velocity Vtheta (Z axis)**, and **Modify angular velocity Vphi (Y axis)**.

The initial angles must be specified in degrees, and the angular velocities can be dependent upon the evolution parameter S. (See *Evolution* (p. 517) for details about evolution.)

v. If you would like to keep the moving part at the reference position (i.e., do not change the topology or symmetry of the moving part), select **Disable update of moving part coordinates**.

d. If your flow is non-isothermal, specify the material properties for the moving part.

Material data

To solve the heat conduction problem in the moving part, you must enter the **Density**, **Thermal conductivity**, and **Heat capacity per unit mass** of the moving part. Additionally, if your moving part is imparting heat to the flow, you will need to specify its **Heat source per unit volume**.

e. Accept the default values for the mesh superposition technique parameters.

Superposition technique

In the **Superposition technique** menu, there are two parameters: the threshold coefficient and the relative compression factor. In general, there is no reason to modify the default values of these parameters.

The threshold coefficient is used to separate the nodes of the flow domain into two categories: those inside the moving part and those outside. Its default value is 0.6; nodes with a value greater than this will be inside the moving part, as discussed in *Navier-Stokes Equations* (p. 458). Increasing the threshold will cause the finite-element representation of the moving part to be slightly thinner.

The relative compression factor (β in *Equation 21–5* (p. 460)) has the default value of 0.01, which should not be modified.

6. Next, you will have to define the flow boundary condition along the moving part. A stick condition, by default, will be applied on all the boundaries of the new moving part. To impose partial or a full slip condition, select

   Flow boundary conditions

   Click **Switch to slip condition**. Then click **Modify slip law and coefficients**. A new menu will present the available slip laws, as for regular boundaries of the flow domain. See *Slip Condition* (p. 176) for more details on the slip condition and the slip laws.

   The switch to the slip condition changes the current velocity interpolation to linear interpolation.

7. Give the moving part a name.

   Modification of title

   To modify the definition of a moving part you have already created, select its name from the **Define moving parts** menu and follow the steps above.

   To delete a moving part, select the **Deletion of a moving part** menu item and then choose the part to be deleted.

8. Define the other parameters of the flow (material data, flow boundary conditions, thermal boundary conditions, etc.) as usual.

9. Define the numerical parameters for the time-dependent task (one level above the sub-task menu).

   Numerical parameters

   See *Additional Guidelines* (p. 474) and *User Inputs for Time-Dependent Problems* (p. 544) for details about the inputs for time-dependent calculations. If you select **Modify the transient iterative parameters**, you will see that the tolerance has been modified for you, as mentioned above.

21.3.3. Time-Dependent Parameters

When you model a time-dependent flow with moving parts, keeping the time step Δt constant will make the analysis of results easier. ANSYS POLYDATA accomplishes this by setting the tolerance for the time-marching scheme to a value much greater than 1, as described in *Setting Up Your Problem in ANSYS POLYDATA* (p. 469).

21.4. Guidelines for Problems with Transient Velocities

For complex motion (i.e., where the direction of the rotation axis is changing), the description of the motion is not trivial. It is possible to have ANSYS POLYFLOW help you set up the complex motion of a moving part. To do this, set up the data file in ANSYS POLYDATA as usual, but do not complete the definition of the motion. Then, run ANSYS POLYFLOW on your data file (for example, `myfile.dat`) using the `-motion` option:

```
polyflow -motion < myfile.dat
```

When you use the `-motion` option, ANSYS POLYFLOW does not solve the flow problem or postprocessors; it just computes the positions of the moving parts and produces a listing file and one or more pairs of flum and flur files.

In the listing file, at the beginning of each step, you can check the values of the angles α, θ, and ϕ, the associated angular velocities, the translation vector, and the translational velocity. You can read the flum and flur files into FLUENT/Post and view the position graphically.

You can also define the motion in a series of steps, rather than defining all components of the motion at once. After each step, save a data file and run ANSYS POLYFLOW on it using the `-motion` option, as described above. You can then check the position after each step by reading the associated flum and flur files into FLUENT/Post.

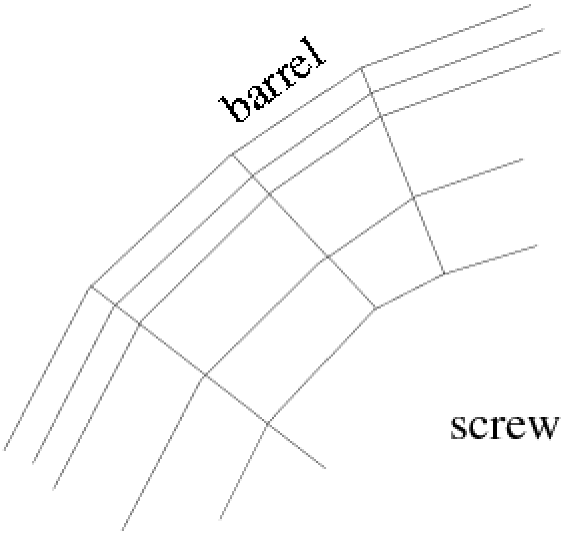
1. Impose a zero translational velocity and zero angular velocities. This allows you to check the initial position. Note that when you change the orientation of the rotation axis, the position of the moving part is also rotated.
2. Impose the angular velocities $V_\theta(t)$ and $V_\phi(t)$ to check the motion of the rotation axis and the resulting effect on the motion of the moving part.
3. Impose the angular velocity $V_\alpha(t)$ to include the rotation.

21.5. Additional Guidelines

Keep the following items in mind when planning and setting up a model with internal moving parts:

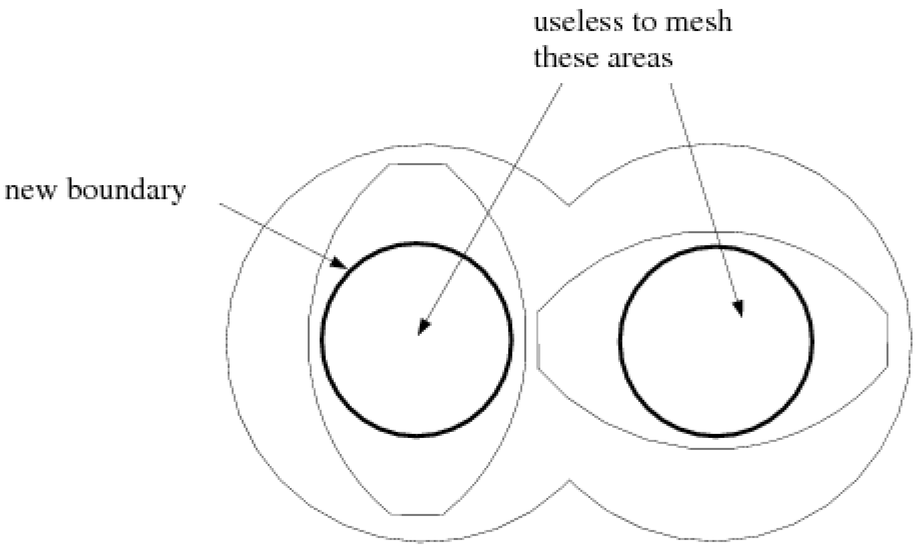
- The meshes for the flow domain and the moving parts should contain elements of about the same size.
- It is important that the mesh be fine enough in regions of small geometrical details; otherwise, the coarse mesh will smooth out the details. For the clearance between a screw and its barrel, at least two elements are recommended in the thickness direction (see *Figure 21.8* (p. 474)). Pressure peaks can also be generated due to a coarse mesh in the angular direction.

Figure 21.8 Elements Between a Screw and its Barrel



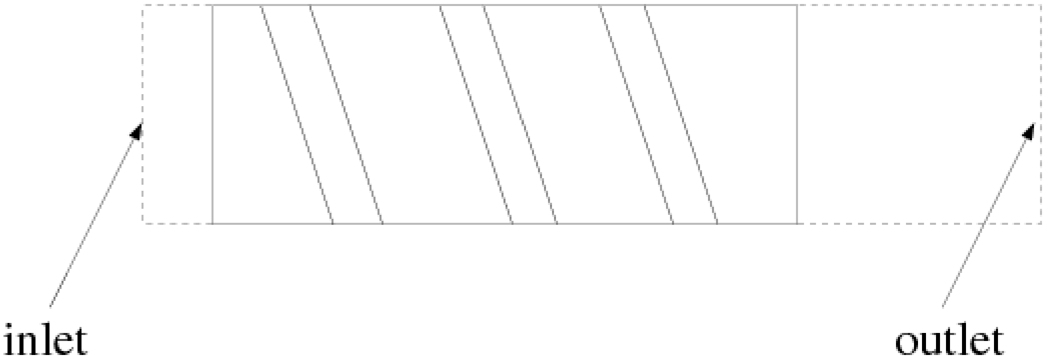
- Areas covered by moving parts that are *never* in the real fluid region do not have to be meshed. Modifying the mesh as shown in *Figure 21.9* (p. 475) will reduce its size and consequently the overall cost of the simulation.

Figure 21.9 Reducing Mesh Size



- If internal boundaries are overlapped by moving parts, make sure that the boundary conditions are compatible with the motion of the moving parts. Some boundary conditions can easily become incompatible with rotating screw constraints. In particular, this may be the case when a screw is rotating and touching the entry/exit sections where an outflow or inflow condition (which implies a zero tangential velocity) is imposed. In this case, you should either extend the flow domain, as shown in *Figure 21.10* (p. 475), or apply zero force conditions instead of outflow or inflow conditions.

Figure 21.10 Extending the Flow Domain to Avoid Incompatibilities in Boundary Conditions



- Make sure that the moving parts are rotating in the right direction. For a transient simulation, it is a good idea to perform a steady-state simulation first, in order to check the direction of rotation.
- Proper definition of the simulation time and the time step are very important. A time step that is too small will increase the CPU time unnecessarily.

  The simulation time should be the time (t_f, in seconds) required for the moving parts to come back to their initial position, which will depend on the symmetry in the simulation:

$$t_f = \frac{\text{number of turns} \times 60}{\text{number of turns per minute}} \tag{21–30}$$

The flow will be calculated at constant time steps (i.e., at constant angular steps). The number of steps (N) should be the number of elements crossed by the tip of the moving parts during the time of simulation, and the time step is determined by

$$\Delta t = \frac{t_f}{N} \tag{21–31}$$

- Using the mini-element for velocity with constant pressure will yield better results than linear velocity with constant pressure, but will be more expensive in terms of time and memory.
- In case of slipping simulation along moving parts, the mesh discretization of the moving parts (especially the boundaries) is very important. It is therefore mandatory to avoid distorted surfaces/volumes adjacent to the boundaries. If you try to modify the geometry of the moving part by dilating or squeezing it using “polyfuse", effects on the results becomes dramatic. You must therefore avoid using this tool to modify the shape of the moving part. The mesh of the fluid domain should be more refined if the slip model is used.

Chapter 22: Sliding Mesh Technique

This chapter demonstrates the use of the sliding mesh technique for solving transient flows with internal moving parts.

22.1. Introduction

ANSYS POLYFLOW incorporates a sliding mesh technique, which can be used in a similar manner to the mesh superposition technique (MST) to simulate transient flows with internal moving parts. The applications include industrial processes such as stirring tanks, single screw extruders, and non-intermeshing batch mixers.

22.1.1. Advantages of the Sliding Mesh Technique

The advantages of using a sliding mesh over MST are as follows:

- The sliding mesh technique is more accurate.
- It does not make any approximation on the shape of moving part. In MST, the shape of moving part depends on the mesh discretization of the flow region.

22.1.2. Limitations of the Sliding Mesh Technique

- You can solve only the simple rotation of a moving part around a fixed axis.
- It does not allow the intermeshing of moving parts.
- In this technique, surround each moving part by a cylinder. These cylinders should neither overlap nor cross boundaries of the flow domain during simulation. In 2D cases, the moving parts must be surrounded by circles.
- It is available for Generalized Newtonian fluids (isothermal or nonisothermal) and heat conduction problems. It is not available for viscoelastic fluids and transport of species.
- Sliding mesh motion is limited to rigid rotation. You can define an angular velocity varying with time, if required. Note that ANSYS POLYDATA will not check if the angular velocity of the sliding mesh is compatible with the boundary condition imposed along the internal boundary of sliding mesh.
- The mixing task is not compatible with the flow fields obtained with sliding mesh techniques. As the node positions of the sliding mesh change with time, the mesh of the flow domain becomes variable. This is a major limitation of the algorithm implemented to track particle paths in ANSYS POLYFLOW.

22.2. Examples

This section provides two examples: one with valid configuration and the other with invalid configuration.

- valid configuration

  Figure 22.1 (p. 478) represents a valid configuration for the sliding mesh technique. Two separate cylinders (`Cylinder #1` and `Cylinder #2`) are defined, which surround two cams. The axis of rotation of each cylinder corresponds to the axis of rotation of the cam. The union of the surrounding domain and of the two cylinders form the fluid region. You can mesh each region separately. It is not necessary to mesh the interior of moving parts.

- invalid configuration

  Figure 22.2 (p. 478) represents an invalid configuration for the sliding mesh technique. You cannot use this technique, because the cylinders surrounding the moving parts are overlapping.

Figure 22.1 Valid Configuration for the Sliding Mesh Technique

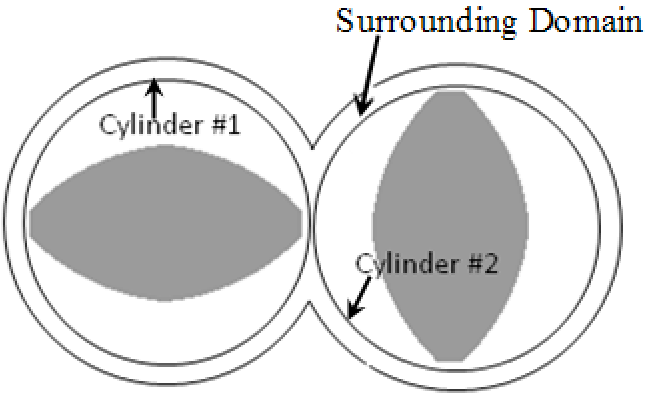
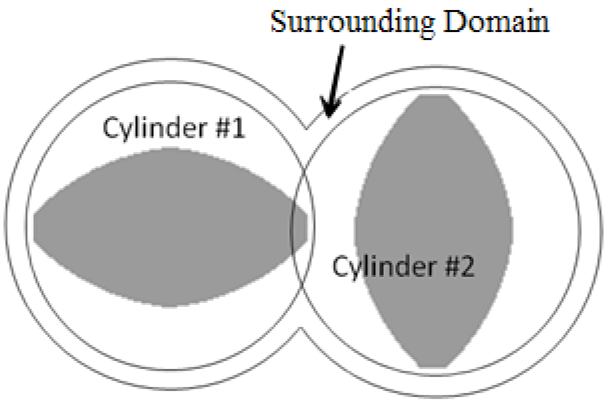


Figure 22.2 Invalid Configuration for the Sliding Mesh Technique



22.3. Meshing

When creating a sliding mesh simulation, mesh each part separately, and ensure that none of the meshes connect. In the valid configuration described previously, the following three mesh regions were required:

- the outer domain of the screw barrel (SD1)
- the left cam (SD2)

- the right cam (SD3)

Note that edge bs5 is separate and coincident to bs2. During the simulation, as nodes on edges bs5 and bs2 slide past each other, the solver will interpolate flow variables across the disconnected mesh. Specific interface boundary conditions are required to couple sliding meshes together. *Figure 22.3* (p. 479) shows the mesh of the surrounding domain (SD1).

Figure 22.3 The Mesh of the Surrounding Domain SD1

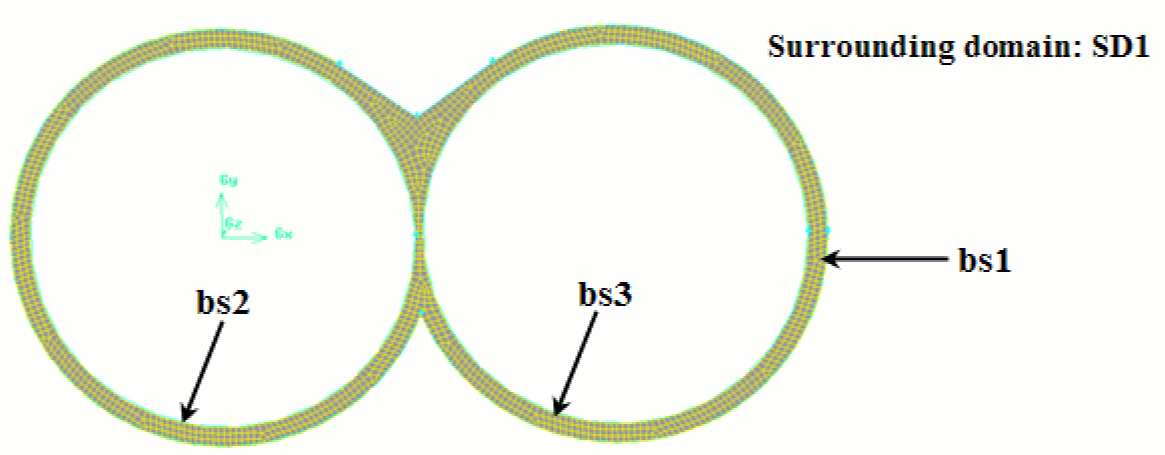
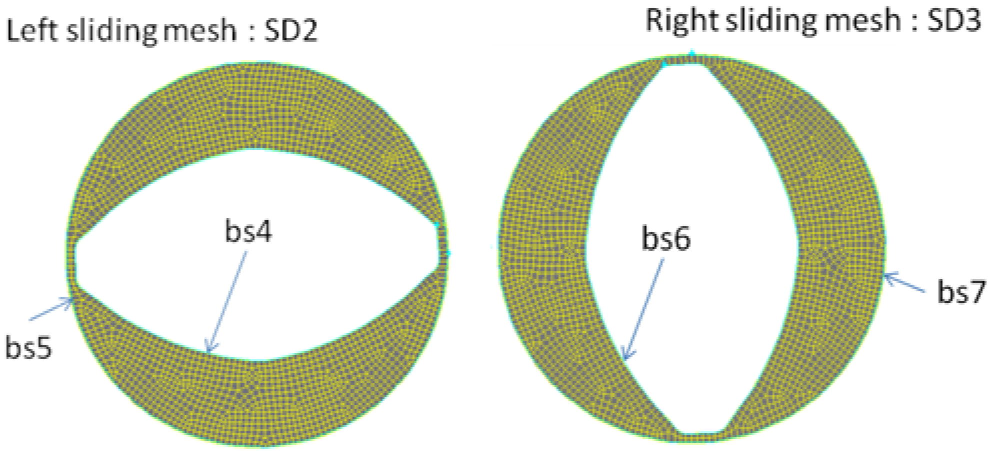


Figure 22.4 (p. 479) shows the mesh of the domains (SD2 and SD3) attached to each moving part. It contains two internal boundaries, bs4 and bs6, and two external boundaries, bs5 and bs7. The internal boundary bs2 of SD1 must be tangential to the external boundary bs5 of SD2.

Figure 22.4 The Sliding Meshes



Similarly, bs3 of SD1 must be tangential to bs7 of SD3. The element size on both boundaries should be same for more accuracy.

In the transient simulation, rotate the mesh SD2 (or SD3) at the same angular velocity prescribed along its internal boundary bs4 (or bs6). Use the non-conformal method to connect the velocity nodes of bs2 (or bs3) to the corresponding nodes of bs5 (or bs7). For additional information on the non-conformal method, see *Non-Conformal Boundaries* (p. 190).

Note

The algorithm ANSYS POLYFLOW uses to connect the boundaries is actualized at each time step as the sliding meshes rotate. The fluid exiting SD1 enters SD2 or SD3 (or vice-versa), and thus the total volume of the fluid is conserved. This flow rate balance is more accurate as the mesh is refined.

22.4. Equations

If inertia is considered, ANSYS POLYFLOW automatically adds a corrective term to the momentum and energy equations considering the rigid motion of sliding meshes. The equations solved on the surrounding domain are not modified.

Thus, the momentum *Equation 10–1* (p. 208) becomes:

$$-\nabla p + \nabla \cdot T + f = \rho a - \rho V_m \cdot \nabla_v \qquad \textbf{(22–1)}$$

while the energy *Equation 13–15* (p. 296) becomes:

$$\rho C_p DT/Dt - \rho C_p V_m \cdot \nabla T = r - \nabla \cdot q + (\sigma D) \qquad \textbf{(22–2)}$$

where Vm is the rigid velocity field corresponding to the rigid rotation of a sliding mesh.

22.5. Guidelines

Some guidelines for using the sliding mesh technique are as follows:

- You can remesh the surrounding domain of a sliding mesh. Do not remesh the sliding mesh domain or deform the internal boundary connected to a sliding mesh boundary.
- When you have several sliding meshes, ensure that the respective domains do not overlap or are not tangential. Maintain at least one element of the surrounding domain between two sliding mesh domains.
- In 2D, the sliding mesh domain must have a circular outer boundary. This domain will rotate around a fixed point, at the center of the circle. In 3D, the sliding mesh domain must have an axisymmetric outer boundary. This domain will rotate around a fixed axis, corresponding to the axis of symmetry of the outer boundary of the sliding mesh domain.
- To accelerate the algorithm that connects nodes, split the boundaries such that there is only one-to-one boundary mapping. *Figure 22.5* (p. 481) and *Figure 22.6* (p. 481) illustrate an example where splitting is useful. In this example, bs3 should be connected to bs6, and bs4 should be connected to bs5.

Figure 22.5 Stirring Tank: Boundaries of the Sliding Mesh

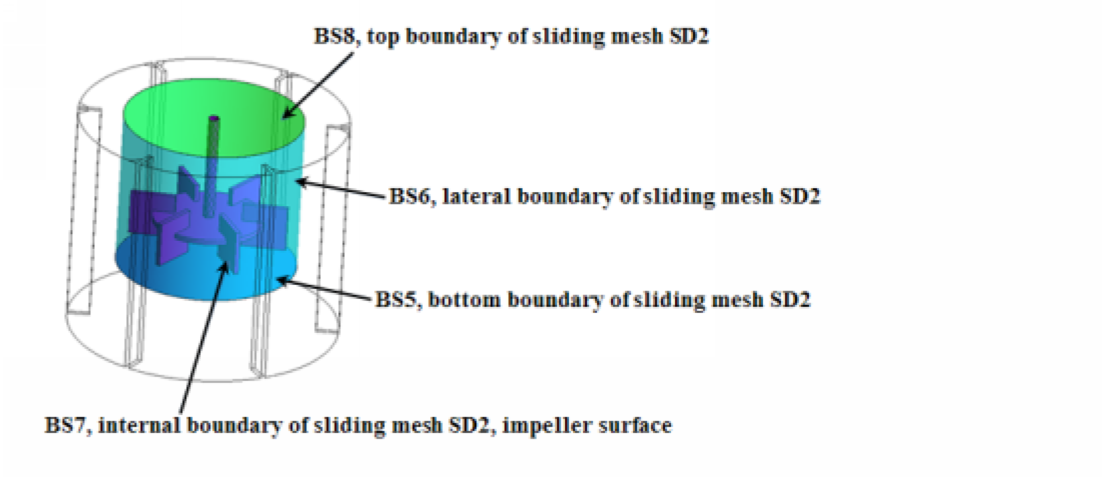
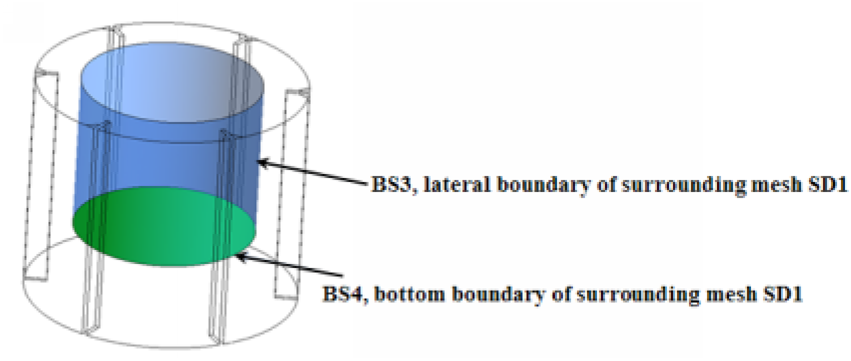

BS8, top boundary of sliding mesh SD2
BS6, lateral boundary of sliding mesh SD2
BS5, bottom boundary of sliding mesh SD2
BS7, internal boundary of sliding mesh SD2, impeller surface

Figure 22.6 Stirring Tank: Boundaries of the Surrounding Mesh


BS3, lateral boundary of surrounding mesh SD1
BS4, bottom boundary of surrounding mesh SD1

- For boundaries of a sliding mesh that are not connected, select a boundary condition that is compatible with the rotation of the sliding mesh.
- Select the time step carefully. The node displacement per time step on the outer boundary of the sliding mesh must be less than (or equal to) the element mesh size.
- Use the sliding mesh technique instead of MST, in case of slipping along the boundary of the moving part. Especially for open domains with large pressure variations between inlet and outlet sections of the flow domain, the balance of flow rates will be much better.
- Define different meshes in the mesher for the surrounding domain and the sliding meshes. You can use ANSYS POLYFUSE to combine them to generate new meshes for new simulations. For more details, see *Combining Meshes with ANSYS POLYFUSE* (p. 151).

22.6. User Inputs

The procedure to incorporate the sliding mesh technique is as follows:

1. Define a sub-task to model the flow domain that contains the surrounding and sliding domains.
2. Define the flow and/or thermal boundary conditions.
    a. Select the relevant boundaries of the surrounding domain to impose **Source of connected condition**.

Source of connected condition

b. Impose **Target of connected condition** on relevant boundaries of the sliding domains. In this menu, select the corresponding source boundary to create a pair of connected boundaries.

Target of connected condition

Note

The panel presents numerical parameters used by ANSYS POLYFLOW when connecting the boundaries. You should not revise these numerical values unless you have been requested to do so by the ANSYS POLYFLOW support team. For additional information about the non-conformal method, see *Non-Conformal Boundaries* (p. 190).

c. Define other boundary conditions, **Inflow**, **wall**, **Cartesian velocity imposed (vx, vy, [vz])**, **Temperature imposed**, **Heat Fluxes**, **symmetry**.

3. Select the **Define Sliding meshes** menu item.

   Define Sliding meshes

4. Click **Creation of a new sliding mesh**.

   Creation of a new sliding mesh

   This creates a new menu, **sliding mesh #1**. Then ANSYS POLYDATA will present further options in a new menu:

   - **Define the moving part domain**

     Specify the list of subdomains corresponding to the sliding mesh domain.

   - **Define the motion**

     Specify the parameters of the rigid rotation of the sliding mesh. You can specify an initial translation vector and/or an initial rotation to position the sliding mesh domain in the surrounding domain correctly.

     ### Note

     Specify the angular velocity in radians per time units, and the initial angle of rotation in radians.

   - **Modification of title**

     You can modify the default name of the sliding mesh.

5. Select the **Define Sliding meshes** menu to define another sliding mesh or to delete an existing one.

6. Click **Upper level menu** repeatedly to return to the task menu.

Chapter 23: Glass Furnaces and Electrical Heating

Information on modeling glass furnaces and electrically heated flows is presented in this chapter. See *Blow Molding and Thermoforming* (p. 379) for information about blow molding.

23.1. Introduction

ANSYS POLYFLOW provides several capabilities that are useful for glass furnace simulations:

- A library of finite elements (especially the 3D mini-element) which lead to significant CPU-time savings when compared to the full quadratic interpolation, while not suffering from spurious pressure modes as occurs with linear interpolation. This capability is described in *Finite-Element Interpolation for 3D Models* (p. 219).
- Subdivided elements for the temperature variable, embedded in a mini-element for the velocity field. Molten glass has a low conductivity and rather high viscosity. This means that the energy equation requires a finer mesh than the momentum equation, since advection-dominated problems are difficult to solve. Fortunately, the temperature is a scalar variable, whereas the velocity field has multiple components. A grid with finer interpolation is used for the temperature field, and this optimizes the number of variables needed for the simulation. This feature dramatically improves the CPU time and memory requirements.
- Automatic evolution schemes on all parameters in order to find the solution of highly nonlinear problems. For natural convection simulations of low-conductivity materials such as glass furnaces, a common practice is to start the simulation at a high value of the conductivity and to decrease it toward the solution of the problem. Typically, it is necessary to perform 10 evolution steps to lower the conductivity to its physical value.

Specific models have also been implemented in ANSYS POLYFLOW to improve glass furnace simulations. These are an electrical heating model, a Rosseland correction, and a macroscopic model to account for the influence of air bubblers on the flow field. They are described in the following sections.

23.2. Electrical Heating

23.2.1. Theory

Performing a local heating electrically in a glass furnace is a common industrial practice. A difference of electrical potential will be applied at given locations in the glass furnace, and an electrical current will heat the glass by Joule effect, which acts as a source term in the energy equation.

The numerical treatment of electrical heating does not present great difficulties. Under classical assumptions, the electrical potential field V is governed by a Laplace equation:

$$0 = \nabla \cdot (\sigma_{\varepsilon} (T) \nabla V) \tag{23–1}$$

where σ_{ε} is the electrical conductivity of the material, and can be a function of temperature T.

Equation 23–1 (p. 486) is associated to boundary conditions of the Dirichlet type (potential imposed) or of the Neumann type (electrical current imposed). In all practical cases, either the electrical potential is imposed as a constant, or the electrical current is zero. Each type of boundary condition can be prescribed in ANSYS POLYDATA along each part of the boundary.

The Joule contribution to the energy equation is written as

$$q_t = - \frac{\sigma_{\varepsilon} \nabla V \cdot \nabla V}{2} \tag{23–2}$$

where q_t stands for the heat source per unit volume in the energy equation.

When considering *Equation 23–1* (p. 486) and *Equation 23–2* (p. 486), it is important to note that the assumption is made of an electric current of the AC type, single phase. This is reflected through the factor $\frac{1}{2}$ in *Equation 23–2* (p. 486), which results from an averaged integration over unit of time. Therefore, peak values should be provided when entering data on potential.

23.2.2. User Inputs for Electrical Heating

The steps needed to incorporate electrical heating in your simulation are listed below:

1. Define a nonisothermal sub-task to model the flow in your problem.
2. Define a separate sub-task for the calculation of the electrical potential.

   Create a sub-task

   a. Select the potential problem type.

      Potential problem

   b. When prompted, specify a name for the sub-task.

3. Specify the region where the electrical potential sub-task applies (i.e., the domain for *Equation 23–1* (p. 486)).

   Domain of the sub-task

4. Specify the temperature dependence of the electrical conductivity.

   Material data

   a. Select **Electrical conductivity**.

      Electrical conductivity

   b. Specify the electrical conductivity in one of the following ways:

- **Constant conductivity**: $\sigma_{\varepsilon}(T) = a$ where a is a constant that you provide.
- **Linear conductivity**: $\sigma_{\varepsilon}(T) = a + b(T - T_0)$ where a, b, and T_0 are constants that you provide.
- **Exponential conductivity**: $\sigma_{\varepsilon}(T) = ae^{b - c/(T - T_0)}$, where a, b, c, and T_0 are constants that you provide. Note that this is a nonlinear temperature dependence, and may result in convergence difficulties.

5. Define the potential boundary conditions on the boundary sets (or along intersections with sub-domains) that form the boundary of the domain of the potential sub-task.

   Potential boundary conditions

   a. Select the boundary for which you want to set potential conditions.
   b. Click **Modify**.
   c. Select the boundary condition type you want to impose. For each boundary, there are two or three possible conditions. By default, ANSYS POLYDATA imposes an insulated condition on all boundaries.

      - Choose **Potential imposed** to specify the electric potential on the boundary.

        Potential imposed

        Select **Constant** to set a constant value.
      - Choose **Insulated boundary** to specify an insulated boundary.

        Insulated boundary

        No further inputs are required.
      - Choose **Interface** (available only for interfaces between sub-tasks) to guarantee continuity of the electrical potential and of the current across the interface.

        Interface

        No further inputs are required.

23.3. Radiative (Rosseland) Correction

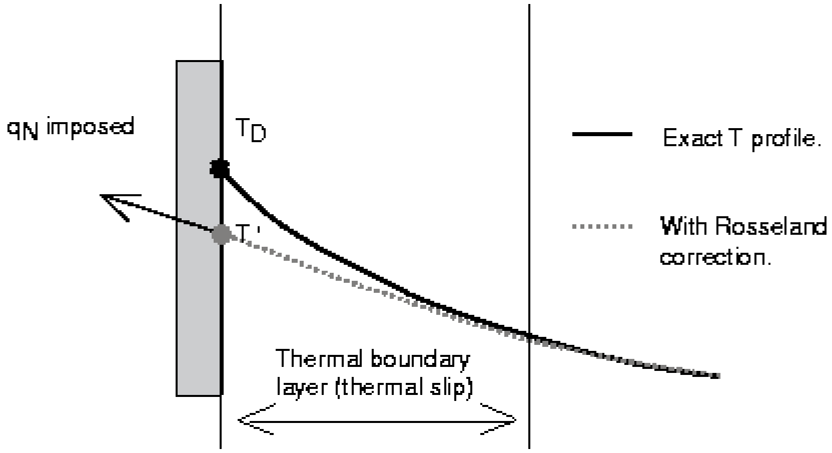
The importance of internal radiation in glass melts is well known. Because glass is a semitransparent medium, radiation must be taken into account in the fluid itself. The Rosseland approximation is a practical model that can be used for realistic 3D simulations. The correction leads to a modification of the apparent conductivity of the material. The so-called Fourier conductivity of the glass is modified by a third-order polynomial correction that models radiation in a semitransparent fluid [*24*] (p. 716).

If you can afford the computational expense, you will achieve more accurate results by defining an internal radiation sub-task rather than applying the Rosseland approximation. See *Internal Radiation* (p. 293) for details. Note that the Rosseland approximation should not be invoked for boundaries of a domain in which an internal radiation sub-task is defined.

23.3.1. Theory

In order to understand the modification of thermal boundary conditions, consider Figure 23.1 (p. 488), which details the temperature profile near a wall. The Rosseland approximation is unable to take into account the sudden modification of the temperature gradient at the boundary, the so-called “radiation slip". This means that the boundary condition $T = T_D$ can no longer be prescribed, although it is the physical condition. The Rosseland correction along a part of the boundary replaces T_D with a flux boundary condition that leads to a wall temperature equal to T'. The difference between T_D and T' is large enough to modify the amplitude of the thermoconvection cells in the whole domain.

Figure 23.1 Effect of the Rosseland Correction Near a Wall


qN imposed
TD
T'
Exact T profile.
With Rosseland correction.
Thermal boundary layer (thermal slip)

When a Rosseland boundary condition is selected, you will enter T_D as the “imposed" temperature. The flux q_N is then calculated on the basis of the refraction index n_{2_σ}, the average absorption a, the Fourier conductivity k (without the third-order correction for internal radiation), and the relative emissivity of the wall ε. The expression for q_N is a fourth-order polynomial function of the temperature T:

$$q_N = q_0 + q_1 T + q_4 \left(T^4 - T_\sigma^4 \right) \quad \textbf{(23–3)}$$

with

$$\begin{aligned} q_0 &= -C\left(kT + 4n_{2_\sigma}\frac{T^4}{3} \right) \\ q_1 &= Ck \\ q_4 &= C\frac{n_{2_\sigma}}{3} \end{aligned} \quad \textbf{(23–4)}$$

and

$$C = \frac{\frac{ak}{4n_{2_\sigma}T^3} + \frac{2}{3}\varepsilon}{\frac{1}{2} + \frac{4}{9}\frac{1-\varepsilon}{9}d_0} \qquad \textbf{(23–5)}$$

where d_0 is a material parameter described in [*12*] (p. 715) and [*15*] (p. 715).

23.3.2. Using Radiative Correction

Using the Rosseland correction requires a modification of Dirichlet temperature boundary conditions along the walls. Note that the correction only makes sense on the parts of the boundary along which you want to prescribe a temperature. No special action is needed along walls where a heat flux is imposed.

Important

Do not apply the Rosseland correction on the boundaries of a domain in which an internal radiation sub-task is defined.

All of the inputs for the radiative correction are entered within the thermal boundary condition specification.

Thermal boundary conditions

1. Along a wall where you want to impose a temperature, select **Rosseland Correction** instead of **Temperature imposed**.

   Rosseland Correction

2. Specify the wall temperature T_D (described in *Theory* (p. 488)).

   Modification of imposed temperature

3. Specify the refraction index (n_{2_σ} in *Equation 23–3* (p. 488) and *Equation 23–5* (p. 489)).

   Modification of refraction index

4. Specify the average absorption coefficient (a in *Equation 23–5* (p. 489)).

   Modification of average absorption

5. Specify the Fourier conductivity (k in *Equation 23–3* (p. 488) and *Equation 23–5* (p. 489)).

   Modification of Fourier conductivity

6. Specify the relative emissivity of the wall (ε in *Equation 23–5* (p. 489)).

   Modification of relative emissivity

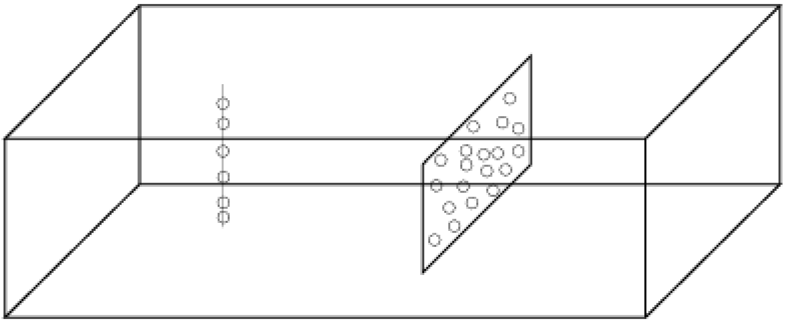
23.4. Bubblers

23.4.1. Introduction

Air bubblers have become an integral part of glass-melting tank design. The numerical simulation of bubbling can follow two different paths: a local description of the bubble trajectory or a global description of the action of the bubbles on the momentum equation. Due to the size of the bubbles (which are small compared to the absolute size of a glass furnace), and because of the huge numerical difficulty of adapting the mesh as a function of the bubble location, the model implemented in ANSYS POLYFLOW is based on a global balance introduced in the momentum equation governing the flow.

The ANSYS POLYFLOW bubbling model adds a force term along a line or plane located in the domain of the simulation, consistent with the finite-element formulation of the problem. For 2D simulations, the force always applies along a line. For 3D simulations, you can select either a line model (which uses a vertical line in the flow domain), or a surface model (to have a 2D surface in the 3D flow domain). See *Figure 23.2* (p. 490) for an illustration.

Figure 23.2 Bubbling Model with 1D and 2D PMeshes



Lines and surfaces must be identified in the mesh. GAMBIT, POLYMESH, PATRAN, and I-deas allow you to import collections of faces (internal or external), which will be identified in ANSYS POLYDATA and on which it will be possible to apply a bubbling model. Defining models on an internal flow region of the flow domain simplifies the domain topology because bubblers do not have to coincide with boundary sets or intersections between internal subdomains. Those 1D and 2D domains internal to an existing mesh are called PMeshes (for “pointer to mesh"). See *PMeshes* (p. 134) for more information about PMeshes.

Note that, since they are defined at the time of mesh generation, PMeshes always correspond to internal mesh lines or mesh faces.

Domains for bubbling are identified within a particular flow sub-task. ANSYS POLYFLOW scans the mesh to find 1D and 2D PMeshes, and it is possible to define a different model on each of these domains. For a mesh that does not include PMeshes, it is not possible to define a model for bubbling.

23.4.2. Theory

Along the lines or surfaces, a force density is calculated as a function of the average bubble radius R_b as follows:

$$\mathbf{f}=\hat{\mathbf{g}}\delta\left(\alpha\mathbf{v}_r+\beta\mathbf{v}_r^2\right) \tag{23–6}$$

where

$\hat{\mathbf{g}}$ = the direction of the gravity vector

δ = the number of bubbles per unit length (1D) or area (2D)

$$\mathbf{v}_r=\mathbf{v}_b-\mathbf{v}$$

$$\mathbf{v}_b=-\frac{2}{3}\hat{\mathbf{g}}\left(gR_b\frac{\rho-\rho_{air}}{\rho}\right)^{\frac{1}{2}}$$

$$\alpha=4\pi\eta R_b$$

$$\beta=\frac{4}{3}\pi\rho R_b^2$$

The quantity $\mathbf{v}_b$ is the bubble velocity, and $\mathbf{v}_r$ is the norm of the relative velocity between the bubble and the fluid. The glass viscosity and density are represented by η and ρ, respectively, while ρ_{air} is the density of air.

These algebraic expressions are derived from the Stokes drag formula for an air bubble, and are taken from [*29*] (p. 716). Using the global model for bubbling does not increase the CPU time of the simulation.

23.4.3. User Inputs for Bubblers

23.4.3.1. Mesh Requirements

In order to include a column of bubbles in your model, your mesh must contain at least one PMesh (a 1D domain within a 2D mesh, or a 1D or 2D domain within a 3D mesh, as shown in *Figure 23.2* (p. 490)). See *PMeshes* (p. 134) for information about PMeshes.

23.4.3.2. Defining a Bubble Column in ANSYS POLYDATA

Within ANSYS POLYDATA, the inputs for the bubble column are as follows:

1. Define a nonzero density for the glass material and a nonzero gravitational acceleration.

   Material data

2. Select the **Bubbling** menu item in the sub-task menu.

   Bubbling

3. In the resulting **Bubbling** panel, select the PMesh for which you want to define a bubble column.
4. Click **Modify**.
5. Specify the parameters for the bubble column:

   a. Set the radius of the bubbles (R_b in *Theory* (p. 490)).

      Modify radius of bubbles

b. Set the density of air (ρ_{air} in *Theory* (p. 490)).

   Modify air density

c. Set the number of bubbles per unit length (for 1D PMeshes) or area (for 2D PMeshes). This is the value of δ in *Theory* (p. 490).

   Modify bubbles density

If you want to remove a bubble column, select the **Delete this column of bubbles** menu item.

6. Repeat these steps for each bubble column you want to define.

Chapter 24: Residual Stresses and Deformations

This chapter provides information about calculating residual stresses and deformations in glass using the Narayanaswamy model.

24.1. Introduction

Residual stresses and deformations are usually encountered in all forming processes, after the shaped object is cooled. While they can be neglected or discarded in some applications, their prediction and anticipation can at times be of utmost importance. This is true in the glass industry in particular, where residual stresses and deformations may induce weaknesses and/or visual surface defects. It is therefore important to be able to identify and quantify these physical properties.

A significant part of residual stresses and deformations originate from the cooling phase in a process. The thermal history often differs according to the location, as well as the material response to such a thermal history. Typically, physical properties in regions that cool slowly will more often reach an equilibrium value, while they may be frozen in another state in regions that cool faster. This behavior is observed for the density, which may reach different values at room temperature, even though it was uniform within the forming process at high temperature.

This chapter will mainly address the behavior of glass during the cooling process, and with the resulting development of residual stresses and deformations. The simulation of the flow behavior of molten glass per se is described in other chapters. For additional information on residual stresses and deformations, see *Residual Deformations and Stresses* (p. 390) and *Inputs for Residual Stresses and Deformations* (p. 411).

24.2. Theory and Equations

24.2.1. Modeling

When dealing with the evaluation of residual stresses and deformations, two quantities of interest are the temperature and the displacement vector. They are respectively denoted by T and $\mathbf{u}$. In elasticity modeling, the deformation tensor ε is defined from the displacement vector $\mathbf{u}$ as follows:

$$\varepsilon = \frac{1}{2}\left[\nabla \mathbf{u} + (\nabla \mathbf{u})^{T} \right] \tag{24–1}$$

Following Chambers [5] (p. 715), the residual stress tensor σ is obtained as follows:

$$\begin{aligned}\sigma(t) \quad = \quad & \mathbf{I}\int_0^t K\,[\xi(t)-\xi(\tau)]\,\frac{d}{d\tau}(\mathrm{tr}(\varepsilon)-\Theta)\,d\tau \\ & +2\int_0^t G\,[\xi(t)-\xi(\tau)]\,\frac{d}{d\tau}\left(\varepsilon-\frac{\mathbf{I}(\mathrm{tr}(\varepsilon))}{3}\right)d\tau\end{aligned} \qquad \textbf{(24–2)}$$

where $K(t)$ and $G(t)$ are the bulk and shear relaxation moduli, respectively, Θ is the thermal strain, $\mathbf{I}$ is the unit tensor, and τ is the integration variable. The residual stress tensor must also satisfy the following momentum equation at all times:

$$\nabla\cdot\sigma=0 \qquad \textbf{(24–3)}$$

In a general form, the bulk and shear relaxation moduli are described by means of Prony series as follows:

$$K(t)=K_\infty+\sum_{p=1}^{P}K_p exp\left[-t/\tau_p\right] \qquad \textbf{(24–4)}$$

$$G(t)=G_\infty+\sum_{q=1}^{Q}G_q exp\left[-t/\lambda_q\right] \qquad \textbf{(24–5)}$$

where K_∞ and K_p are bulk elastic moduli, G_∞ and G_q are shear elastic moduli, and τ_p and λ_q are the relaxation times for each Prony component. In general, these moduli are characterized by P and Q modes, respectively, and by a steady value reached after relaxation. In *Equation 24–2* (p. 494), ξ is a reduced time variable which is given by the integral invoking a shift function suggested by Narayanaswamy [*21*] (p. 716):

$$\xi(t)=\int_0^t exp\left[Ax\left[\frac{1}{T_{ref}}-\frac{1}{T(\tau)}\right]+A(1-x)\left[\frac{1}{T_{ref}}-\frac{1}{T_f(\tau)}\right]\right]d\tau \qquad \textbf{(24–6)}$$

where T_{ref} and T_f are a reference temperature and the fictive temperature, respectively. In *Equation 24–6* (p. 494), A denotes the ratio of activation energy to the ideal gas constant, while x is a fractional parameter ranging between 0 and 1. Often, the temperature is expressed in a non-absolute scale, and the relationship for calculating ξ is then written as

$$\xi(t) = \int_0^t exp\left[Ax\left[\frac{1}{T_{ref}-T_0} - \frac{1}{T(\tau)-T_0} \right] + A(1-x)\left[\frac{1}{T_{ref}-T_0} - \frac{1}{T_f(\tau)-T_0} \right] \right] d\tau \tag{24–7}$$

where T_0 denotes the absolute zero temperature in the current scale (e.g., –273 when the temperature is given in Celsius).

The fictive temperature T_f is related to the actual temperature by means of the following equation:

$$T_f(t) = T(t) - \int_0^t M[\xi(t) - \xi(\tau)] \frac{dT(\tau)}{d\tau} d\tau \tag{24–8}$$

where $M(t)$ is a structural relaxation modulus, and is given by

$$M(t) = \sum_{r=1}^{R} M_r exp\left[-t/\mu_r \right] \tag{24–9}$$

where M_r are the individual weights for each Prony component of the structural relaxation function and μ_r are the corresponding relaxation times.

Finally, the thermal strain Θ in glass is a function of both temperature and fictive temperature histories. It is therefore obtained from an integral involving liquid and glassy properties described by means of the corresponding expansion coefficients a_l and a_g as follows:

$$\Theta = \int_{T_0}^{T_f} a_l(u)\, du + \int_{T_f}^{T} a_g(u)\, du \tag{24–10}$$

where the coefficients a_l and a_g can be constants, or up to third-order polynomial functions of the temperature.

24.2.2. Material parameters

As can be seen, the evaluation of residual stresses and deformations requires the knowledge of a large set of parameters. In particular, three relaxation spectra are needed: the bulk and shear relaxation moduli, as well as the structural relaxation modulus. These three moduli (given by *Equation 24–4* (p. 494), *Equation 24–5* (p. 494), and *Equation 24–9* (p. 495), respectively) involve $2P+1, 2Q+1$ and $2R$ parameters. Next, the reduced time expression shown in *Equation 24–7* (p. 495) involves four parameters—namely, the ratio A of activation energy to the ideal gas constant, a fractional parameter x, a reference temperature T_{ref}, and the absolute zero temperature T_0 expressed in the current temperature scale. Finally, liquid and glassy expansion coefficients a_l and a_g are required in *Equation 24–10* (p. 495). Note that they can be temperature dependent. Since the evaluation of residual stresses and deformations also involves a cooling phase, the energy equation is also solved, and this requires the knowledge of the density ρ, the thermal conductivity $k\ (T\ )$, and the heat capacity $C_p\ (T\ )$.

24.2.3. Boundary Conditions

The calculation of residual stresses and deformations requires solving the energy equation for the cooling phase and the above equations for the nonisothermal elastic case. These differential equations require appropriate boundary conditions, and the transient character of the process requires the assignment of appropriate initial conditions as well. From the point of view of the energy equation, the initial temperature will typically correspond of that of the glass at the end of the forming process, before cooling. Cooling will usually occur via convection, which requires the specification of a transfer coefficient and an outside temperature.

The momentum equation (*Equation 24–3* (p. 494)) also requires initial and boundary conditions. Since it is often assumed that thermal stresses develop from a stress-free state, initial conditions are set to zero. The assignment of boundary conditions is less obvious and depends on the individual case. The most typical conditions will probably be symmetry, stress-free or displacement-free border, though other choices are possible.

24.2.4. Physical Interpretation

Before entering into some details of the integration of the previous equations, it is interesting to consider a few comments on the physics that produces residual stresses and deformations. Imagine an initially stress-free glass sample at a high and uniform temperature. The sample is cooled via a heat exchange with the outside world (e.g., convection). The actual temperature T decreases nonuniformly at a rate that depends on the cooling conditions. We already understand that the thermal history of individual glass particles will differ. From *Equation 24–8* (p. 495), we see that a nonuniform fictive temperature field T_f develops, at the same time as a nonuniform thermal strain field Θ. Simultaneously, with the decay of both actual and fictive temperatures, the reduced time scale ξ is modified and slows down. Stresses develop in accordance with the constitutive *Equation 24–2* (p. 494); they relax at a decreasing speed, since the reduced time slows down.

It is important to note that the nonuniform thermal history of the individual glass particles creates a nonuniform field of thermal strain. Thus, even though elastic boundary conditions may be selected in such a way that no mechanical stress is generated, internal stresses will be created because of the thermal history. From *Equation 24–8* (p. 495), we also understand that residual stresses and deformations will be reduced when the cooling rate is low. Indeed, under such conditions, the fictive temperature T_f will remain close to the actual one. When the cooling rate is fast, the fictive temperature can be frozen at a high value, and this will then be accompanied by an extremely long relaxation mechanism due to the sharp decrease of the reduced time.

24.2.5. Numerical Treatment

As can be seen from *Modeling* (p. 493), the equations governing the development and relaxation of residual stresses and deformations are highly coupled and nonlinear. In addition, time integrations are also involved, which provide a time-dependent attribute to the whole system. Having said that, it is yet possible to derive differential expressions which make the numerical treatment significantly easier. For this, let us assume that all fields are known at time t^n, and that we want to calculate their values at the next time $t^{n+1} = t^n + \Delta t$.

Following Chambers [5] (p. 715), the stress tensor can be written as the following:

$$\begin{aligned}\sigma^{n+1} \;=\; & \mathbf{I}\sum_{p=1}^{P}\sigma_p^{n+1}+\mathbf{I}K_\infty\left[\varepsilon^{n+1}-\Theta^{n+1}\right]+\sum_{q=1}^{Q}\sigma_q^{n+1} \\ & +2G_\infty\left[\varepsilon^{n+1}-\mathbf{I}\varepsilon^{n+1}/3\right]\end{aligned} \tag{24–11}$$

where σ_p^{n+1} and σ_q^{n+1} denote the spherical and deviatoric contributions of the stress tensor, respectively, at time t^{n+1}. The expressions for these contributions are as follows:

$$\sigma_p^{n+1}=e^{-\left[\xi^{n+1}-\xi^{n}\right]/\tau_p}\left(\sigma_p^{n}\right)+K_p h_p\left(\Delta t\right)\left[\left[\varepsilon^{n+1}-\varepsilon^{n}\right]-\left[\Theta^{n+1}-\Theta^{n}\right]\right] \tag{24–12}$$

$$\begin{aligned}\sigma_q^{n+1}=e^{-\left[\xi^{n+1}-\xi^{n}\right]/\lambda_q}\left(\sigma_q^{n}\right)+2G_q g_q\left(\Delta t\right)\Big[\left[\varepsilon^{n+1}-\varepsilon^{n}\right] \\ -\mathbf{I}\left[\varepsilon^{n+1}-\varepsilon^{n}\right]/3\Big]\end{aligned} \tag{24–13}$$

In *Equation 24–11* (p. 497), Θ^{n+1} is the thermal strain given by

$$\Theta^{n+1}=\Theta^{n}+a_g\left(T^{n}\right)\left[T^{n+1}-T^{n}\right]+\left[a_l\left(T_f^{n}\right)-a_g\left(T_f^{n}\right)\right]\left[T_f^{n+1}-T_f^{n}\right] \tag{24–14}$$

As can be seen, *Equation 24–12* (p. 497) and *Equation 24–13* (p. 497) involve the reduced time ξ^{n+1}, which can be evaluated as

$$\xi^{n+1}=\xi^{n}+\int_{t^n}^{t^{n+1}} exp\left[Ax\left[\frac{1}{T_{ref}}-\frac{1}{T\left(s\right)}\right]+A\left(1-x\right)\left[\frac{1}{T_{ref}}-\frac{1}{T_f\left(s\right)}\right]\right]ds \tag{24–15}$$

where s is the integration variable. Two auxiliary functions are also found in *Equation 24–12* (p. 497) and *Equation 24–13* (p. 497), and are given by

$$h_p(\Delta t) = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \left(e^{-[\xi(t^{n+1}) - \xi(s)]/\tau_p}\right) ds \tag{24–16}$$

$$g_q(\Delta t) = \frac{1}{\Delta t}\int_{t^n}^{t^{n+1}} \left(e^{-[\xi(t^{n+1}) - \xi(s)]/\lambda_q}\right) ds \tag{24–17}$$

It is interesting to note that the *Equation 24–15* (p. 497) – *Equation 24–17* (p. 498) involve an integration over the time interval $\left[t^n, t^{n+1}\right]$, which can be performed via a simple sub-integration algorithm, over k subintervals.

Finally, following Markovsky et al. [*19*] (p. 715), the fictive temperature can be evaluated with the following formula:

$$T_{fr}^{n+1} = \frac{\mu_r T_{fr}^n + \Delta t \exp\left[Ax\left[\frac{1}{T_{ref}} - \frac{1}{T^n}\right] + A(1-x)\left[\frac{1}{T_{ref}} - \frac{1}{T_f^n}\right]\right] T^n}{\mu_r + \Delta t \exp\left[Ax\left[\frac{1}{T_{ref}} - \frac{1}{T^n}\right] + A(1-x)\left[\frac{1}{T_{ref}} - \frac{1}{T_f^n}\right]\right]} \tag{24–18}$$

with

$$T_f^{n+1} = \sum_{r=1}^{R} M_r T_{fr}^{n+1} \tag{24–19}$$

Once again, a sub-integration algorithm can be used over k subintervals.

When considering the relationships in *Equation 24–11* (p. 497) – *Equation 24–19* (p. 498), we see that several auxiliary fields are calculated, although they are not explicitly required by the user. These fields include the individual modes for the bulk and shear stress σ_p and σ_q, their corresponding functions $h_p(\Delta t)$ and $g_q(\Delta t)$, and the individual fictive temperature T_{fr}. All these quantities are involved in equations that are solved locally. Therefore, piecewise continuous interpolation can be selected, which is also the simplest and cheapest interpolation from the point of view of computation.

It is important to guarantee a stable behavior of the solver for the calculation of residual stresses and deformations for any bulk and shear relaxation spectrum. Therefore, a technique similar to the DEVSS [*14*] (p. 715) method is invoked. It consists of adding a term to the momentum equation that is expressed as a function of displacements gradient, and to remove its counterpart expressed as a function of the deformation tensor. The evaluation of the deformation tensor is included in the simulation.

24.3. Problem Setup

The basic procedure for setting up a simulation that uses the Narayanaswamy model to calculate the residual stresses and deformations is fairly straightforward. First of all, create a transient task with the appropriate geometric attributes. Then perform the following steps:

1. Create a sub-task for the simplified viscoelastic flow problem.

   Create a sub-task

   a. Select the appropriate problem from the **Create a sub-task** menu.

      Residual stresses and deformations (Narayanaswamy)

   b. When prompted, specify a name for the sub-task.

2. Specify the region where the sub-task applies.

   Domain of the sub-task

3. Define the material properties.

   Material data

   a. Specify the density.
   b. Specify the thermal conductivity.
   c. Specify the heat capacity per unit mass.
   d. Specify the average temperature, presently used as initial temperature.
   e. Specify the heat source per unit volume.
   f. Define the data for Narayanaswamy model.

      Data for Narayanaswamy model

      i. Define the bulk relaxation modulus (*Equation 24–4* (p. 494)).

         Bulk relaxation modulus

         The bulk relaxation modulus is defined as a Prony series (P terms) plus a constant (asymptotic) value, and requires $2P + 1$ parameters. You have to specify the asymptotic value, as well as the number of modes (up to 8) and the corresponding parameters. You have the option of changing the entire spectrum or one mode only.

      ii. Define the shear relaxation modulus (*Equation 24–5* (p. 494)).

         Shear relaxation modulus

         The shear relaxation modulus is defined as a Prony series (Q terms) plus a constant (asymptotic) value, and requires $2Q + 1$ parameters. You have to specify the asymptotic value, as well as the number of modes (up to 8) and the corresponding parameters. You have the option of changing the entire spectrum or one mode only.

iii. Define the structural relaxation modulus (*Equation 24–9* (p. 495)).

Structural relaxation modulus

The structural relaxation modulus is defined as a Prony series (R terms), and requires $2R$ parameters. You have to specify the number of modes (up to 8) and the corresponding parameters. You have the option of changing the entire spectrum or one mode only.

iv. Define the shift function (*Equation 24–6* (p. 494) and *Equation 24–7* (p. 495)).

Shift function

Specify the parameters for the Narayanaswamy function, together with the associated parameters—namely, the ratio A of activation energy to the ideal gas constant, the fractional parameter x, the reference temperature T_{ref}, and the absolute zero temperature T_0 expressed in the current temperature units.

v. Define the glassy dilation function.

Glassy dilation function

Specify the glassy dilation function, which can be a third-order polynomial function of the temperature.

vi. Define the liquid dilation function.

Liquid dilation function

Specify the liquid dilation function, which can be a third-order polynomial function of the temperature.

4. Define the thermal boundary conditions.

Thermal boundary conditions

By default, temperature is imposed on all boundaries. For all boundary sides, the only possibilities are **Temperature imposed**, **Flux density imposed**, or **Insulated boundary**. See *Boundary Conditions* (p. 173) and *Problem Setup* (p. 286) for details.

5. Define the displacement boundary conditions.

Displacement boundary conditions

By default, vanishing displacements are imposed on all boundaries. For all boundary sides, you can impose a combination of normal/tangential displacement and force, as well as symmetry (when geometrically relevant). See item 6 in *Problem Setup* (p. 511) for details.

Important

Note that inflow boundary conditions should be selected at inlet(s) with the flow rate, as this is the only way of imposing a relevant boundary condition for the viscoelastic variable.

6. (optional) Modify the interpolation scheme for temperature.

Interpolation

You only have access to the interpolation for the temperature. Low-level (and therefore computationally cheap) interpolation is selected for the stress unknowns and auxiliary fields.

Chapter 25: Fluid Structure Interaction (FSI) Model

ANSYS POLYFLOW can be used to solve problems involving nonisothermal elasticity in a solid using its Fluid Structure Interaction (FSI) Model. This model can be used as a postprocessor or it can be coupled with a flow problem.

This chapter provides information about the FSI model and contains the following sections:

25.1. Introduction

The FSI model in ANSYS POLYFLOW allows to you to take into account the elasticity of a solid region coupled to the flow in your simulation. Even very small geometry deformations can affect the flow. During die design, for example, for a thin die such as coat hanger the balance of the die can be sensitive to die deformation. Temperature has an effect on die deformation, and thus on the flow. The result is that the elastic model can produce unwanted stresses on the solid part that must be considered.

25.2. Theory

For elasticity in a solid region, ANSYS POLYFLOW solves the motion equation for displacement by:

$$\nabla \cdot \sigma + \mathbf{f} = \mathbf{0} \tag{25–1}$$

where σ is the stress tensor and $\mathbf{f}$ is the body force. Note that the transient term in *Equation 25–1* (p. 503) is neglected.

In ANSYS POLYFLOW small deformation is assumed and the stress tensor is given by the following thermoelastic constitutive equation:

$$\sigma = \frac{E}{1+v}\left(\frac{v}{1-2v}\mathrm{tr}\,(\varepsilon)\,I + \varepsilon\right) - \frac{E}{1-2v}\alpha\,(T - T_0)\,I \tag{25–2}$$

where E = Young’s modulus

v = Poisson’s ratio

α = lineic dilatation coefficient (coefficient of thermal expansion)

T = temperature

T_0 = reference temperature without stresses (reference temperature for dilatation)

The strain tensor ε is related to the displacement $\mathbf{d}$ by:

$$\varepsilon = \frac{1}{2}\left(\nabla \mathbf{d} + (\nabla \mathbf{d})^{T}\right) \tag{25–3}$$

E, v, and α can be constant or temperature dependent using an Arrhenius law or an approximated Arrhenius law. In 2D, ANSYS POLYFLOW always assumes planar deformation. When an elastic sub-task is defined, ANSYS POLYFLOW can also compute the elastic stresses as a postprocessor, using *Equation 25–2* (p. 503).

25.2.1. Effect of Elastic Sub-task on the Flow Domain

When an elastic domain is deformed, adjacent flow domains are modified. To reduce nonlinearities that may be introduced in this process, you can keep the flow domain unchanged by disabling the update of coordinates. When you choose to use updated coordinates, the coordinates for elastic deformation in the elastic domain are given by:

$$\mathbf{X}_{new} = \mathbf{X}_{old} + \mathbf{d} \tag{25–4}$$

To activate (or deactivate) this option, select **Switch to [update of coordinates]** or **Switch to [no update of coordinates]** in the material data menu of the elastic sub-task.

This option allows you to check deformations before using them on the flow domain under the following conditions:

- if the deformation has any effect, or a very minor effect on the flow behavior.
- if the deformation or the stresses are significant for your simulation.
- if there is any trouble with convergence.

In addition to using updated coordinates, a remeshing technique must be defined for the flow domain. Boundary conditions need to be defined in such a way that they are compatible with the boundary conditions of the elastic sub-task, at the intersection with the elastic solid. Any incompatibility in boundary conditions can lead to deformation of the mesh.

Figure 25.1 Compatibility of Boundary Conditions

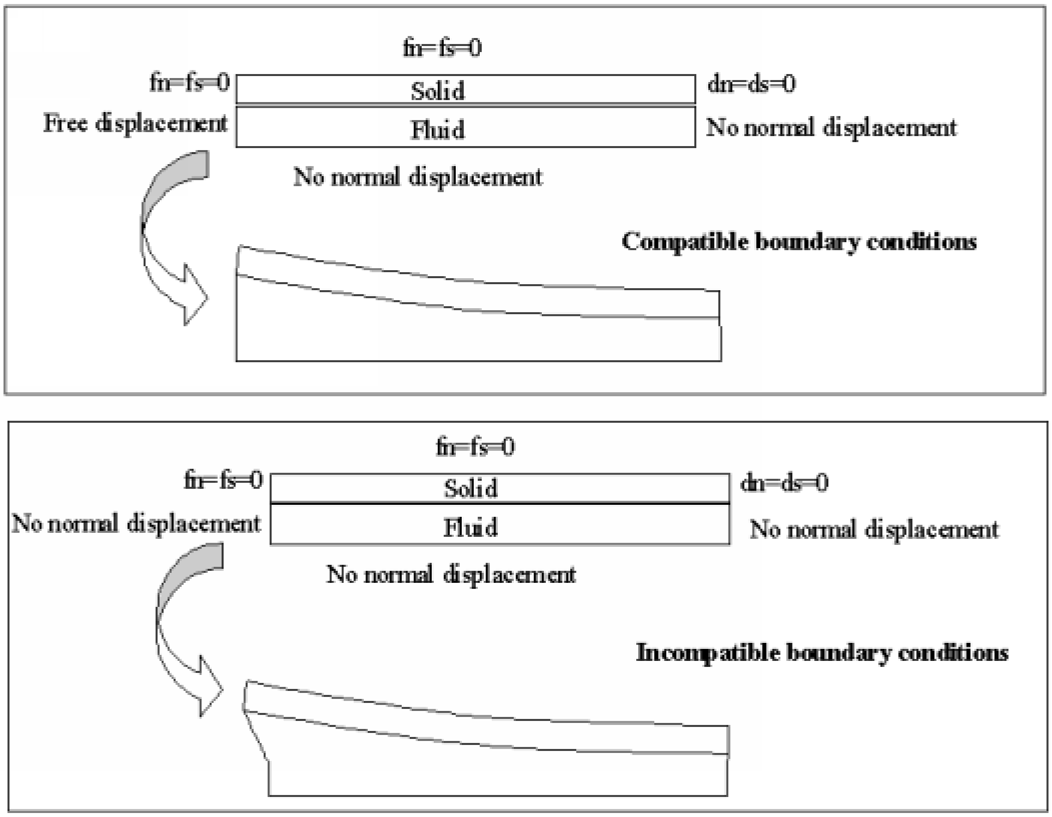


Figure 25.1 (p. 505) illustrates both compatible and incompatible boundary conditions (top and bottom views, respectively). For both cases, boundary conditions along the left boundary sections are considered (*Figure 25.1* (p. 505)). For compatible boundary conditions, the solid zone has zero force applied and the *free displacement* boundary condition is imposed on the flow domain for the remeshing technique. Due to the shear force along the wall, the solid endures a deformation in the direction of flow and the fluid follows this deformation.

For incompatible boundary conditions (*Figure 25.1* (p. 505)) the solid zone has zero force applied and *no normal displacement* imposed for the remeshing technique on the flow domain. Due to the shear force along the wall, the solid endures a deformation in the flow direction while the *no normal displacement* boundary condition prevents the fluid region from following this deformation. Instead, the element beside the interface is deformed, in response to the solid deformation.

25.3. Elasticity Boundary Conditions

To solve an elastic problem, ANSYS POLYFLOW requires information about the displacement (or force) along the boundary of the computational domain. To have a well-posed elastic problem, boundary conditions must avoid rigid motion.

Important

Rigid motion causes convergence problems and may result in a zero pivot error message.

The default condition defined by ANSYS POLYDATA on each section of a boundary is zero displacement (normal and tangential), except for the axis of symmetry in an axisymmetrical model. In this case, ANSYS

POLYDATA automatically creates a symmetry condition along the line r = 0. Where appropriate, keep the default boundary condition and specify new conditions for the rest of the boundary sections.

The boundary conditions available in ANSYS POLYDATA are:

- interface with solid.
- interface with fluid.
- normal and tangential displacement imposed.
- normal and tangential force imposed.
- normal displacement and tangential force imposed.
- normal force and tangential displacement imposed.
- plane or axis of symmetry.
- contact with parison.
- border of moving part.
- assign displacement at points.

When setting elastic boundary conditions in ANSYS POLYDATA, do the following:

1. Select the **displacement boundary conditions** in the sub-task menu. ANSYS POLYDATA will open the **Displacement boundary conditions** panel, displaying the default boundary conditions.
2. Select the appropriate boundary section.
3. Click **Modify** to change the boundary condition.

25.3.1. Interface With Solid

The *interface with solid* condition establishes continuity between two sides of the intersection between different elastic sub-tasks. For the elastic momentum equation, the *interface with solid condition* guarantees continuity of the displacement field and of the contact forces. The *interface with solid* condition is available only for the boundary section that is defined by an intersection with an adjacent subdomain.

If you define an *interface with solid* condition for one sub-task, you must also define this condition for the sub-task on the other side of the boundary section. To define an *interface with solid* condition between two elastic sub-tasks, choose **Interface with solid** from the list of boundary conditions.

Interface with solid

25.3.2. Interface With a Fluid

The *interface with a fluid* condition is used when the contact force produced by the fluid is applied as a force boundary condition along the intersection between fluid and solid domains. ANSYS POLYFLOW computes the force along the wall required to satisfy the velocity condition for the flow from the fluid momentum equation. It applies this force as boundary condition for the elastic momentum equation.

If you give the *interface with a fluid* condition for the elastic sub-task, you must give the *Interface with elastic solid* condition on the other side of the boundary for the flow boundary condition. For details, see *Interface with Elastic Solid* (p. 186).

To define this condition between an elastic and a flow sub-task, choose the **Interface with a fluid** menu item in the list of boundary condition choices.

Interface with a fluid

Important

On the elastic side, this condition does not require other inputs.

25.3.3. Normal and Tangential Displacement Imposed

The *normal and tangential displacement imposed* condition allows you to specify the normal and tangential displacement components on the boundary section. The normal displacement component is denoted by d$_n$ and the tangential component by d$_s$. This condition is used when the boundary section is either fixed or embedded. Though the default condition corresponds to d$_n$ = d$_s$ = 0, you can assign nonzero values for one or both of the components.

Important

In 3D, d$_s$ is a vector with two components.

Note the sign convention for d$_n$ and d$_s$. A positive d$_n$ signifies that the displacement emerges from the domain of the sub-task.

- A positive d$_s$ is oriented in 2D along the anticlockwise direction on the closed curve surrounding the domain of the sub-task for an external boundary. For internal boundaries of 2D domains that are not simply connected, the orientation is reversed.
- It is complicated to assign a value of d$_s$ in 3D because the definition of d$_s$ depends on conventions of orientation that are not easy to define.

  Therefore in 3D, the normal and tangential displacement condition is used for a zero tangential displacement only.

To define the normal and tangential displacement on a boundary section you will need to do the following:

1. Select **Normal and tangential displacement imposed (dn, ds)** from the list of boundary conditions.

   Normal and tangential displacement imposed (dn, ds)
2. Specify d$_n$ as **Constant**, **Linear function of coordinates**, **Map from CSV (Excel) file**, or **User Defined Function**.
3. Select the **Upper level** menu. ANSYS POLYDATA will open the menu where you can specify d$_s$ in a similar manner (2D only).

25.3.4. Normal and Tangential Force Imposed

The *normal and tangential force imposed* condition allows you to specify the normal force (f$_n$) and tangential force (f$_s$) components on the boundary section. This condition is used for free boundaries. Both forces are expressed in units of Pascal.

The default condition corresponds to $f_n = f_s = 0$, but you can assign nonzero values for one or both of the components. The normal and tangential directions are defined in *Normal and Tangential Displacement Imposed* (p. 507).

For the tangential displacement in 3D, it is not possible to assign a nonzero value for f_s due to orientation conventions.

To define the normal and tangential force on a boundary section you will need to do the following:

1. Select **Normal and tangential force imposed (fn, fs)** from the list of boundary conditions.

   Normal and tangential force imposed (fn, fs)

2. Specify f_n as **Constant** or **Linear function of coordinates**.
3. Select the **Upper level** menu. ANSYS POLYDATA will open the menu where you can specify f_s in a similar manner (2D only).

25.3.5. Normal Displacement and Tangential Force Imposed

The *normal displacement and tangential force imposed* condition combines displacement and force boundary conditions described in *Normal and Tangential Displacement Imposed* (p. 507) and *Normal and Tangential Force Imposed* (p. 507), respectively. The default condition corresponds to $d_n = 0$ and $f_s = 0$, but you can assign nonzero values for the normal displacement d_n. It is not possible to assign a nonzero value for the tangential force f_s due to orientation conventions in 3D.

To define the normal displacement and tangential force on a boundary section, you will need to do the following:

1. Select **Normal displacement and tangential force imposed (dn, fs)** from the list of boundary conditions.

   Normal displacement and tangential force imposed (dn, fs)

2. Specify d_n as **Constant**, **Linear function of coordinates**, **Map from CSV (Excel) file**, or **User Defined Function**.
3. Select the **Upper level** menu. ANSYS POLYDATA will open the menu where you can specify f_s as a **Constant** or a **Linear function of coordinates** (2D only).

25.3.6. Normal Force and Tangential Displacement Imposed

The *normal force and tangential displacement imposed* condition combines displacement and force boundary conditions described in *Normal and Tangential Displacement Imposed* (p. 507) and *Normal and Tangential Force Imposed* (p. 507), respectively. The default condition corresponds to $f_n = 0$ and $d_s = 0$, but you can assign nonzero values for the normal force f_n. It is not possible to assign a nonzero value for the tangential displacement d_s due to orientation conventions in 3D.

To define the normal force and tangential displacement on a boundary section, you will need to do the following:

1. Select **Normal force and tangential displacement imposed (fn, ds)** from the list of boundary conditions.

Normal force and tangential displacement imposed (fn, ds)

2. Specify f_n as **Constant** or **Linear function of coordinates**.
3. Select the **Upper level** menu. ANSYS POLYDATA will open the menu where you can specify d_s as **Constant**, **Linear function of coordinates**, **Map from CSV (Excel) file**, or **User Defined Function** (2D only).

25.3.7. Symmetry Condition

The *symmetry* condition is equivalent to imposing zero values for the normal displacement and tangential force components, respectively. For axisymmetric models, ANSYS POLYDATA identifies the axis of symmetry (r=0) automatically and imposes the symmetry condition on that section.

To define symmetry condition, you will need to select **Plane of symmetry (fs=0, dn=0)** from the list of boundary conditions.

Plane of symmetry (fs=0, dn=0)

25.3.8. Contact with the Parison

The *contact with the parison* condition is available only for elastic postprocessor sub-tasks that relate to a contact problem. For example, the postprocessor can be used to compute the deformation of the mold and elastic stresses generated during a blow molding process.

This condition applies the force produced by the contact, as the boundary condition for the elastic subtask. After detecting the contact, the force produced by the contact is applied to the part of the mold that is in contact with the parison, while a zero-force is applied to the remaining surface contact of the mold.

To define this condition on a boundary section, you will need to select **Contact with a parison** from the list of the boundary conditions. Further inputs are not required.

Contact with a parison

25.3.9. Border of a Moving Part

The *border of a moving part* condition is available only for elastic postprocessor sub-tasks that relate to moving parts. For example, the postprocessor can be used to compute the deformation of a screw in an extruder and the elastic stresses generated in the screw. This condition applies the force produced by the flow on the moving part as the boundary condition.

To define this condition on a boundary section, you will need to select the **Border of a moving part** from the list of the boundary conditions. Further inputs are not required.

Border of a moving part

25.3.10. Assign Displacement at Points

The *assign displacement at points* condition allows you to impose the displacement component at certain points within the computational domain. This condition is used to avoid rigid motion of a solid.

You will need to specify the coordinates of the points where the displacement is imposed, which component is involved, and the value of the displacement. Note, that the displacement will be imposed at the closest node. To specify this condition, do the following:

1. Click **Assign displacement at points** in the **displacement boundary conditions** panel.

   Assign displacement at points

   This will open the **Assign displacement at points** panel.

2. In the **Assign displacement at points** panel, select **Creation of a new constraint** menu item to create a new constraint.

   Creation of a new constraint

   A constraint with a default name **Constraint #I** will be created, where `I` is the default integer index (e.g., 1, 2, 3,....). Note that you can change the name of the constraint, if you choose. By default, no constraint is applied.

3. Select the newly-created constraint by selecting **Constraint #I** (or the label you have chosen, e.g., **myconstraint**) from the list of constraints to change the default or the current constraint.

   Constraint #I

4. Select **Modify coordinates of point** to change the location of the constraint.

   Modify coordinates of point

   Specify the value of **X** and **Y** (or **X**, **Y**, and **Z** in 3D).

5. Select **Impose displacement-X** (-Y or -Z as appropriate) to impose the displacement of a component.

   Impose displacement-X

   Note that this option changes and becomes **Do not impose displacement-X** (-Y or -Z) if **Modify displacement-X** (-Y or -Z) has been enabled.

6. Click **Do not impose displacement-X** (-Y or -Z) to remove the constraint on the X (Y or Z) component.

7. Click **Modify displacement-X** (-Y or -Z) to specify the value of the displacement component.

   Modify displacement-X

8. Select the **Deletion of a constraint** option to delete the unwanted constraints.

   Deletion of a constraint

   Specify the constraint to be deleted.

9. Click the **Assign displacement at points** button on the **displacement boundary conditions** panel to see the constraints again.

   After creating constraints for displacement, when you return to the **displacement boundary condition** panel, it will not be possible to verify constraints for displacement previously defined. To review the constraints, you will need to select **assign displacement at points**.

Assign displacement at points

25.4. Problem Setup

The procedure for setting up an elastic problem for a solid region coupled with a flow problem, is described below. This process assumes that you know how to define a flow problem in ANSYS POLYFLOW.

1. Create a sub-task for the elastic problem.

   Create a sub-task

   a. Select the appropriate problem type from the **Create a sub-task** menu.

      - For an isothermal elastic problem in a solid region coupled with flow problem, select:

        Elasticity (isothermal)

      - For a nonisothermal elastic problem in a solid region coupled with flow problem, select:

        Elasticity (non-isothermal)

      - For an isothermal elastic postprocessing in a solid region, select:

        Postprocessor Elasticity (isothermal)

      - For a nonisothermal elastic postprocessing in a solid region, select:

        Postprocessor Elasticity (non-isothermal)

   b. When prompted, enter a name for the sub-task.

2. Specify the region where the elastic sub-task is to be applied.

   Domain of the sub-task

3. Define the material properties.

   Material data

4. Define the elastic data.

   Elastic data

   a. For an elastic problem, enter values for the Young's Modulus (E) and the Poisson's ratio (ν), as defined in *Equation 25–2* (p. 503).

      Modify E

      Modify Mu

Important

Note that while Poisson's ratio is referred to as **Mu** in the previous menu item, it is elsewhere denoted by ν (e.g., *Equation 25–2* (p. 503)).

By default the effect of an elastic deformation on the flow domain is active. Select **Switch to [no update of coordinates]** to ignore this effect.

Switch to [no update of coordinates]

b. For a nonisothermal elastic problem, specify the lineic dilatation coefficient (α) and the reference temperature for dilatation (T_0), as defined in *Equation 25–2* (p. 503).

Modify B0

Modify Tref

i. Specify the temperature dependencies of the Young's Modulus, the Poisson's ratio and the lineic dilatation coefficient. Note that the default setting is no temperature dependence.

+ temperature dependence on E

+ temperature dependence on Mu

+ temperature dependence on B

ii. Select **Modify temperature dependence law**, and

Modify temperature dependence law

choose from the following:

Switch to [no temperature dependence] law

Switch to [Arrhenius approximate] law

Switch to [Arrhenius] law

iii. Specify heat conduction parameters: thermal conductivity (k) (*Equation 13–2* (p. 283)), density (ρ), specific heat capacity (C_p), and heat source per unit volume (r) (*Equation 13–4* (p. 284)).

Density

Thermal conductivity

Heat capacity per unit mass

Heat source per unit volume

iv. Add an average temperature to improve convergence of nonlinear problems.

Average temperature

5. For a nonisothermal elastic problem, define the thermal boundary conditions.

   Thermal boundary conditions

   Follow the procedure for the heat conduction problem in *Problem Setup* (p. 286).

6. Define the displacement boundary conditions.

   Displacement boundary conditions

   a. Select the boundary you want to set the displacement conditions for.

   b. Click **Modify**.

   c. Select the boundary condition type you want to impose. By default, ANSYS POLYDATA imposes zero displacement for each external boundary.

      - Choose **Interface with a solid** to specify an interface between 2 elastic solids.

        Interface with a solid

      - Choose **Interface with a fluid** to specify an interface between an elastic solid and a flow domain.

        Interface with a fluid

      - Choose **Normal and tangential displacement imposed (dn, ds)** to specify the normal and tangential displacement on the boundary.

        Normal and tangential displacement imposed (dn, ds)

        For each component, select the appropriate specification method:

        – **Constant** to impose a constant value for the current component of the displacement.

        – **Linear function of coordinates** to impose a linear function of the form d=A+Bx+Cy+Dz for the current component of the displacement.

        – **Map from CSV (Excel) file** to impose a displacement profile contained in a CSV file for the current component of the displacement.

        – **User Defined Function** to impose a displacement profile using a user-defined function for the current component of the displacement.

      - Choose **Normal and tangential force imposed (fn, fs)** to specify the normal and tangential forces, respectively, along the boundary.

        Normal and tangential force imposed (fn, fs)

        For each component, select the appropriate specification method:

  - **Constant** to impose a constant value for the current component of the force.
  - **Linear function of coordinates** to impose a linear function of the form d=A+Bx+Cy+Dz for the current component of the force.

- Choose **Normal displacement and tangential force imposed (dn, fs)** to specify the normal displacement and zero tangential force, respectively, on the boundary.

  Normal displacement and tangential force imposed (dn, fs)

  For the normal displacement, select the appropriate specification method:

  - **Constant** to impose a constant value for the normal displacement.
  - **Linear function of coordinates** to impose a linear function of the form d=A+Bx+Cy+Dz for the normal displacement.
  - **Map from CSV (Excel) file** to impose a displacement profile contained in a CSV file for the normal displacement.
  - **User Defined Function** to impose a displacement profile using a user-defined function for the normal displacement.

- Choose **Normal force and tangential displacement imposed (fn, ds)** to specify the normal force and zero tangential displacement, respectively, on the boundary.

  Normal force and tangential displacement imposed (fn, ds)

  For the normal force, select the appropriate specification method:

  - **Constant** to impose a constant value for the normal force.
  - **Linear function of coordinates** to impose a linear function of the form d=A+Bx+Cy+Dz for the normal force.

- Choose **Plane of symmetry (fs =0, dn =0)** to specify the symmetry condition. No further inputs are required.

  Plane of symmetry (fs=0,dn=0)

- Choose **Contact with a parison** to apply the contact force from the parison on the contact surface of a mold. Note that this option is available only for an elastic postprocessor.

  Contact with a parison

- Choose **Border of a moving part** to apply the contact force from the fluid on border of a moving part. Note that this option is available only for an elastic postprocessor.

  Border of a moving part

- Choose **Assign displacement at points** to impose a displacement at designated points, in the computational domain.

  Assign displacement at points

- Create a new constraint.

  Creation of a new constraint

- Enter a name of the constraint, and specify the position by selecting **Modify coordinates of points**.

  Modify coordinates of points

- Specify which component you want to impose (-X, -Y or -Z).

  Impose displacement-X

- Specify the value of the displacement component (-X, -Y or -Z).

  Modify displacement-X

7. Create a sub-task for the flow problem.

   Create a sub-task

   Define your flow sub-task (e.g., **domain**, **material data...**). Note that ANSYS POLYDATA detects interfaces with fluid boundary conditions for which a flow problem has yet to be defined and prompts you to define a flow sub-task.

8. Define the flow boundary conditions (e.g., inflow, wall, outflow) for all boundary conditions except between the flow and solid domains.

   Flow boundary conditions

   a. Select the boundary you want to set the interface with solid condition for and click **Modify**.

   b. Select the **Interface with elastic solid** condition.

      Interface with elastic solid

      i. Choose **Switch to slip condition** to apply a slip boundary condition. By default, a stick condition (zero velocity) is applied.

         Switch to slip condition

      ii. Depending upon the type of boundary condition you choose (stick or slip), specify the stability coefficient (stick) or the slipping and penalty coefficient (slip).

          Modify stability coef.

          Modify Fslip

          Modify penalty coef.

9. If you have selected the **update of coordinates** option for a material parameter of the elastic sub-task, you must specify the global remeshing of the flow domain.

   Global remeshing

   Define the elastic remeshing that is required, paying particular attention to the compatibility between boundary conditions of the remeshing and boundary conditions of the elastic sub-task.

25.4.1. Numerical Parameters in Elastic Problem Coupled with Flow Problem

When modeling elasticity in a solid or a flow problem, it is not typically necessary to use an evolution scheme because equations are usually linear. However, note that even if separate problems converge without evolution, some convergence problems could occur if the retrospective effect of the solid deformation on the flow domain is taken into account.

- If you are encountering convergence difficulties, there are three ways to overcome them:
  - Check the deformation of the solid by selecting the **no update of coordinate** option in the **Material data** menu of the elastic sub-task.
  - If the difficulty is related to the deformation of the solid part, then it is recommended that you use an evolution scheme which decreases the value of the Young's modulus (see *Evolution* (p. 517) for details).
  - An alternative solution is to use an evolution scheme on flow parameters (e.g., flow rate or viscosity) which dominates the force produced by the fluid and applies it on the solid interface.
- If deformations are dominated by dilatation, then it is recommended that you use an evolution scheme which increases the dilatation coefficient, or the difference between the temperature and the reference temperature.
- For a large-scale problem, select the **Decoupled computation of the moving surfaces** option in the **numerical parameters** panel.

  Decoupled computation of the moving surfaces

  This option will iteratively compute the flow problem and the elastic sub-tasks. An advantage of utilizing this option is reduced memory consumption, but the convergence rate will be slower.

Chapter 26: Evolution

Evolution is a powerful tool provided by ANSYS POLYFLOW to assist you in solving nonlinear flow problems. Information about evolution is provided in the following sections:

26.1. Introduction

Despite the fact that there are more linear techniques and methods than nonlinear ones, the physical world is a nonlinear place. Fluid mechanics frequently exhibits nonlinearity, including the following:

- The material parameters used in modeling heat conduction, such as conductivity or specific heat, may be temperature dependent, and thus cause a nonlinearity.
- In Newtonian flows, inertia terms are quadratic in terms of the velocity components; also, the shear-rate dependence of the viscosity causes nonlinearity.
- In nonisothermal flow problems, heat convection generates terms that are products of velocity components and temperature gradients.
- In viscoelastic flows, the constitutive equations are highly nonlinear.

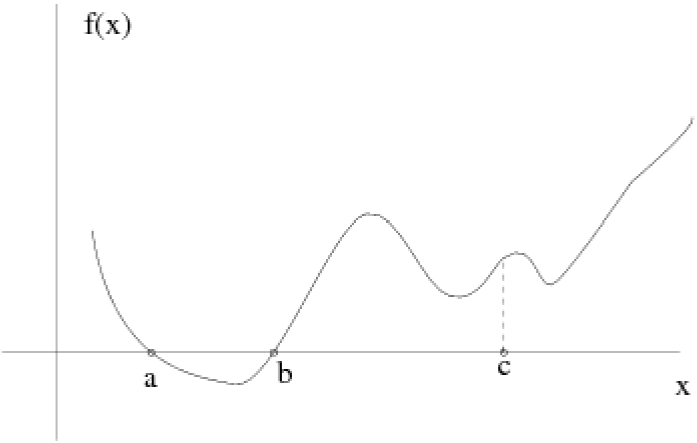
The evolution procedure in ANSYS POLYFLOW greatly simplifies the handling of the computational difficulties of nonlinear problems.

26.2. Nonlinearity and Evolution

For several reasons, nonlinearity often makes computations difficult. Nonlinear problems frequently have multiple families of solutions. Also, the solution for a given set of parameters cannot generally be extrapolated from an initial, linear solution, since dramatic changes in the nature of the flow can occur with a slight change of flow parameters.

Typically, the solution of a nonlinear problem is surrounded by a domain of convergence. You can expect to reach the solution only if your initial guess lies within this domain. An example of this is the following. Suppose you want to find a solution to $f(x) = 0$ where $f(x)$ is the function graphed in *Figure 26.1* (p. 518). There are two solutions, $x = a$ and $x = b$, but it will be difficult to find $x = a$ when starting from $x = c$.

Figure 26.1 A Nonlinear Problem: $f(x) = 0$



In most cases, the only way to solve a nonlinear problem is to begin with a solution to a simpler problem, one in which nonlinearity is not as troublesome. From this solution, you then solve a sequence of problems of increasing nonlinearity, using the solution of one problem as the initial condition for the subsequent problem. Ultimately, the sequence should lead to the original problem and its solution.

Suppose that, in the original problem, a certain parameter (such as density) that has value x is found (or is suspected) to be largely responsible for the nonlinear nature of the problem. You can then consider a sequence of problems, in each of which this parameter is given the value xS, where $0 \leq S \leq 1$. The sequence begins with $S = 0$; if a solution is obtained with the parameter having value $xS = 0$, then S is increased, by a small amount, to $S' = S + \Delta S$, and a new problem is created with parameter value xS'. If a solution is obtained, then the procedure is continued, using a larger value of the increment, such as $\Delta S_{next} = 1.5 \Delta S$. If the problem diverges, then the previous solution is returned to and a smaller increment, such as $\Delta S_{next} = 0.5 \Delta S$, is tried.

At each step, there is a limit to how much S can be increased. Each problem solution can be used successfully as the initial condition for only those problems with sufficient proximity to the initial condition. Increasing S by too much will result in a problem setup that will diverge with the previous solution used as an initial condition.

The evolution procedure in ANSYS POLYFLOW is more general than that described above. Instead of assigning the parameter the value xS, you can choose a function f and use the value $xf(S)$ for the parameter. You can also have S increase over any interval, instead of requiring $0 \leq S \leq 1$. Though evolution can be applied to more than one parameter, it is generally most effective when applied to a single parameter (with some exceptions, such as when applied to flow rate and pulling velocity).

26.3. Available Evolution Functions

The choices of the evolution function f are listed below:

- $f(S) = S$

  This is the default function.

- $f(S) = 1/S$

When using this function, you must make sure that S is never zero by setting the initial value and upper limit of S appropriately in the **Numerical parameters** menu. This function is recommended when the power law index is used as an evolutive parameter. It is also recommended in order to decrease the thermal conductivity for nonisothermal flows with a high Péclet number.

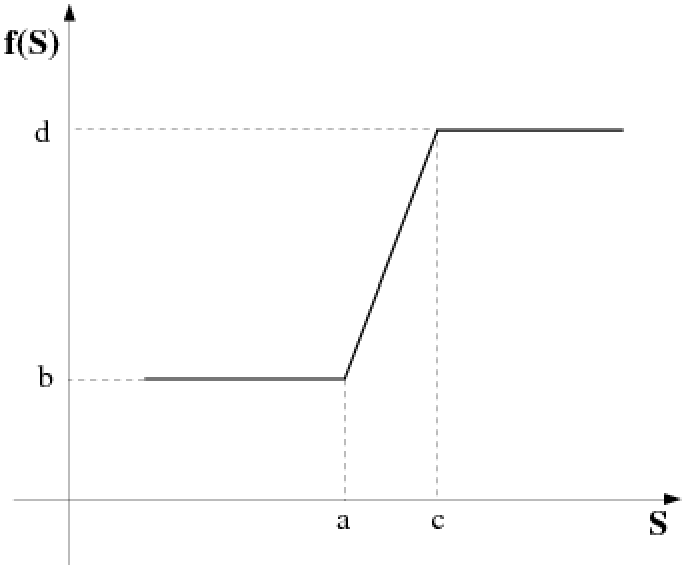
- $f(S) = a + bS + cS^2 + dS^3$

  The values of a, b, c, and d can be set, allowing a wide variety of polynomial functions.

- $f(S) =$ ramp function

  This is a function with four parameters (a, b, c, and d) that define the corners of a ramp function as illustrated in *Figure 26.2* (p. 519).

Figure 26.2 Ramp Function



- $f(S) = a\cos(bS + c) + d + eS$

  This is a function with five parameters (a, b, c, d, and e) that combines a linear component and a periodic component.

- $f(S) = aS^b + cS^d$

  This function has four parameters (a, b, c, and d) that control its rate of growth.

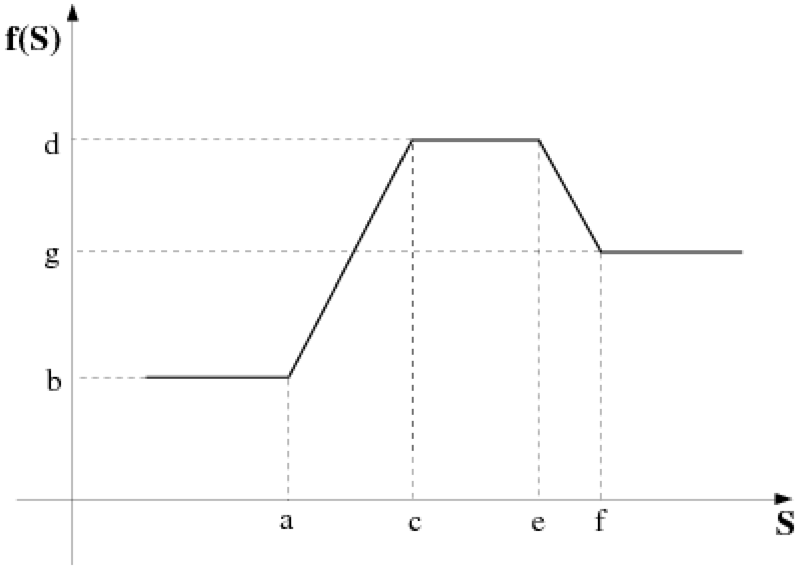
- $f(S) = ae^{bS} + c + dS$

  This function is suggested when wall slip is used as an evolutive parameter.

- $f(S) =$ double ramp function

  This is a function with seven parameters (a, b, c, d, e, f, g) that define the corner values of a double ramp function as illustrated in *Figure 26.3* (p. 520).

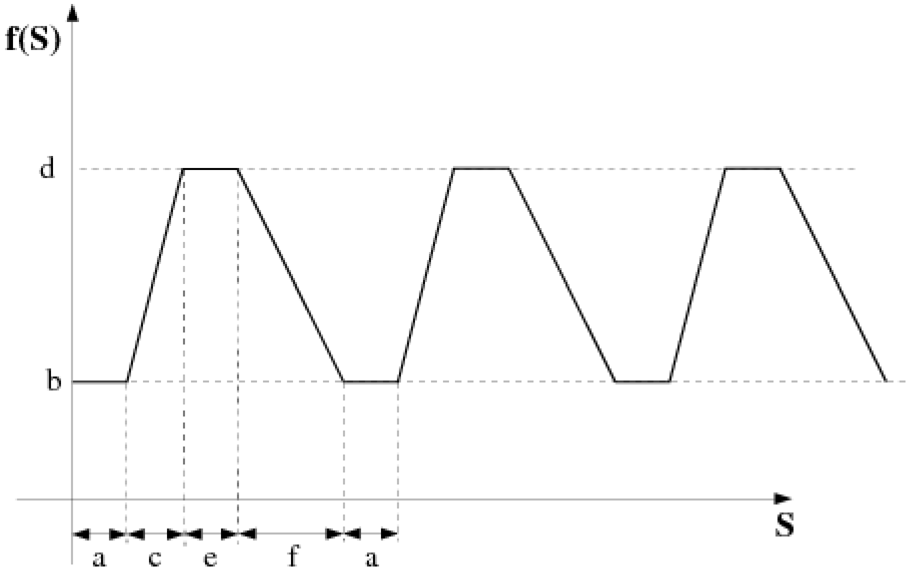
Figure 26.3 Double Ramp Function



- $f\ (S)\ =$ trapezoidal wave

This is a function with six parameters (a, b, c, d, e, f) that define a trapezoidal wave as shown in *Figure 26.4* (p. 520).

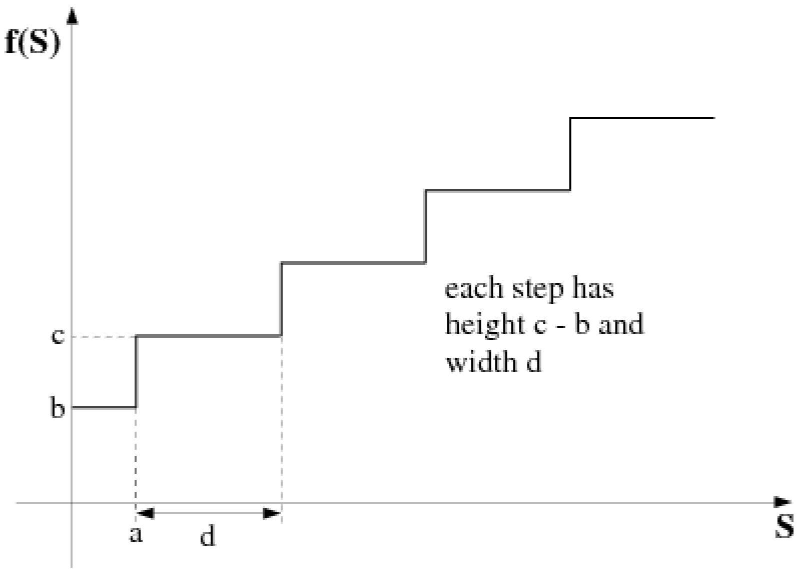
Figure 26.4 Trapezoidal Wave Function



- $f\ (S)\ =$ H step function

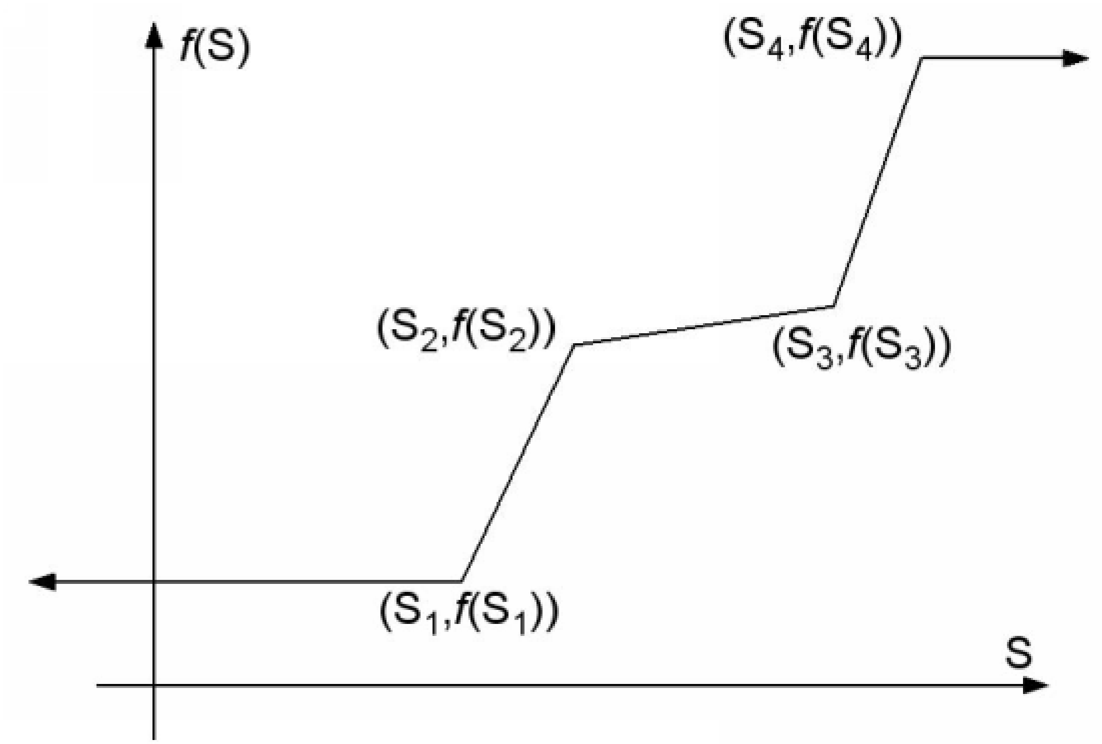
This is a step function with four parameters (a, b, c, d). It is illustrated in *Figure 26.5* (p. 521).

Figure 26.5 H Step Function



- $F(S)$ = multi-ramp function. The multi-ramp function is a generalization of the simple ramp. It can also be converted into a piecewise linear function, and it is thus defined by means of pairs of data (S_i, $f(S_i)$), where all S_i are given in ascending order. *Figure 26.6* (p. 521) illustrates such a function.

Figure 26.6 Multi-ramp Function Defined with Four Pairs of Data



The definition of such a function can be done interactively by introducing the sequence of pairs of data; the results of which can be saved into a dependence file. Alternatively, the data for the multi-ramp function can be read from a file, where the pairs of data must be written with the fixed (Fortran) format 2e14.7. This means that each pair of data (S_i, $f(S_i)$) is written on a single line, and fourteen digits are used for representing each data as ±0.1234567e ± 89. Since the format is fixed, there is no need to use a separator. For a single data, the format e14.7 successively involves: a sign, a 0 digit, the decimal dot, seven digits (for the mantissa), the letter "e" standing for a power of 10 eventually given by its sign and the last two digits. In such a format, - π is then represented as -0.31415926e+01. If an entry (S_i, $f\ (S_i)$) in the file consists of (0.25,- π), for example, it will be represented as +0.2500000e+00-0.31415926e+01. Subsequent pairs of data are written on subsequent lines.

26.4. Using Evolution

26.4.1. When to Use Evolution

Evolution should be used only when a problem will not otherwise converge. You should always try solving the problem without evolution before deciding if you need to use it.

If your problem does not converge, try simplifying the problem. For instance, if your problem is nonisothermal, try solving the isothermal version first and then start the nonisothermal calculation using the isothermal solution as a starting point. If this doesn't help, or if your problem is already isothermal, you should consider using evolution.

26.4.2. Determining an Appropriate Evolution Parameter

The key to the evolution procedure is to identify the parameters governing the nonlinearity of the problem. Possible choices include the following:

- density in problems with inertia
- flow rate in a viscoelastic flow
- relaxation time in a viscoelastic flow
- moving boundaries
- power index for a power law fluid
- heat conductivity in a heat transfer problem
- slip coefficient for an extrusion process
- surface tension in a free surface problem

Before starting an ANSYS POLYDATA session, you should decide which material parameters or boundary conditions will be governed by evolution. These parameters and boundary conditions may concern different sub-tasks, though the evolution in S (e.g., S_0, S_1, δS, the number of iterations, and other numerical parameters) is a global task attribute. A list of suggested evolution parameters for various types of problems is given in *Table 26.1: Suggestions for Evolution Parameters* (p. 522).

To determine which parameter should be governed by evolution, try increasing (or decreasing) one parameter by a factor of 10 (e.g., decrease flow rate, increase the power law index, etc.). Do this until you are able to get the problem to converge (this may require modifying more than one parameter).

Table 26.1 Suggestions for Evolution Parameters

Problem	Evolution Parameter
Large viscous heating	Flowrate Q

Problem	**Evolution Parameter**
Power law index less than 0.7	n, or Picard scheme
Bird-Carreau, low index	n or Q, or Picard scheme
Differential viscoelastic fluid	Flowrate Q or relaxation time λ
3D free surface without surface tension (extrusion and coextrusion)	Evolution on moving boundaries; alternatively, evolution on the slip coefficient
Free surface with surface tension (straight free surface is acceptable)	Surface tension coefficient σ; use quadratic coordinates
Free surface with surface tension (recirculations present on the free surface)	Switch to time-dependent mode; use quadratic coordinates
Moving contact point	Surface tension coefficient σ
Natural convection, low conductivity	Thermal conductivity k or gravity g
Large Péclet number (low conductivity) with temperature dependence of the viscosity	Flowrate Q or thermal conductivity k
Integral viscoelastic problem	Evolutive viscosity
Radiation	Reference temperature T_σ
Forced convection or inertia	Flowrate Q or density ρ
Strong temperature dependence of viscosity	α or Q
Interface between two fluids, $\frac{\eta_1}{\eta_2}$ large	η_1, with η_2 fixed so that $\frac{\eta_1}{\eta_2}$ increases with S
Fiber spinning	Evolution on the pulling velocity
Extrusion with intense cooling	Heat transfer coefficient

Now that you have determined the parameter that is causing your problem to fail to converge, you should define an evolution problem on that parameter. To do this, you will assign a value, x, and a function f of S, to this parameter, and select an interval of S, $S_1 \leq S \leq S_2$, so that

- in the problem you want to solve, this parameter has value $xf\ (S_2)$
- when the parameter is given the value $xf\ (S_1)$, a solution can be obtained

26.4.3. Problem Setup

After you have determined the parameter on which you will apply evolution, follow the steps below to set up the problem:

1. In the **Create a new task** menu, specify an evolution problem, and define the geometry type, etc., in the usual way.

   Evolution problem(s)

2. To set an evolutive parameter, use the following steps to define the parameter's value as $xf\ (S)$:

   - For an evolutive material parameter, follow these steps:

     a. Go to the menu appropriate for setting the value of this parameter. For instance, if the parameter is density, select the **Density** menu item in the **Material data** menu.

        Density

     b. Click the **EVOL** button at the top of the ANSYS POLYDATA menu to enable the evolution inputs.

        The **EVOL** button will change to **EVOL [on]** to indicate that the evolution inputs are enabled.

     c. Select the menu item appropriate for the evolutive parameter. For density, this would be **Modification of density**.

        Modification of density

     d. Enter x when ANSYS POLYDATA prompts for the **New value**.

     e. Click **OK** to accept this new value.

     f. A menu will appear from which you will select the function f. The available functions are described in *Available Evolution Functions* (p. 518).

        If several parameters are evolutive, a different function f can be chosen for each one.

     g. Select **Upper level menu** to complete the specification.

     h. Click the **EVOL** button again to disable the evolution inputs.

        The **EVOL** button will change to **EVOL [off]** to indicate that the evolution inputs are disabled.

   - For an evolutive boundary condition, follow these steps:

     a. Select **Flow boundary conditions** or **Thermal boundary conditions** from the sub-task menu, or **Fluid Fraction boundary conditions** from the **Material Data** menu of the fluid fraction transport sub-task.

b. Select the boundary condition you want to set, and click **Modify**.

c. Click the **EVOL** button at the top of the ANSYS POLYDATA menu to enable the evolution inputs.

   The **EVOL** button will change to **EVOL [on]** to indicate that the evolution inputs are enabled.

d. Select the type of boundary condition you want to impose. For instance, for a flow boundary condition, you might choose **Normal and tangential velocities imposed (vn, vs)**.

e. After you enter the value of each numerical parameter (such as v_n) associated with this boundary condition, clicking **Upper level menu** will cause a menu to appear, from which you can select the function f. The available functions are described in *Available Evolution Functions* (p. 518).

   Boundary conditions can be position dependent (such as a linear function of coordinates) and evolutive (S dependent). That is, x may be a constant, or a function of coordinates. If several parameters are evolutive, a different function f can be chosen for each one.

f. Select **Upper level menu** to complete the specification.

g. Click the **EVOL** button again to disable the evolution inputs.

   The **EVOL** button will change to **EVOL [off]** to indicate that the evolution inputs are disabled.

Repeat this step for each evolutive parameter.

3. Continue setting up your problem in the usual way.

4. After all sub-tasks have been defined, select **Numerical parameters** from the **F.E.M. Task** menu.

Numerical parameters

In the **Numerical parameters** menu, select **Modify the evolution parameters**.

This will bring up the **Evolution parameters** menu, where you can modify the following parameters of the evolution:

- **initial value of S**

  This is the value of S used to calculate the parameter value(s) for the first computation.

Important

If $f\ (S)\ = 1/S$ is being used for any parameter, the initial value and upper limit of S should be chosen to prevent S being set to 0.

- **upper limit of S**

  This value of S should be chosen to give parameter value(s) that correspond to the value(s) in the problem you want to solve.

- **initial value of delta-S**

  This is the starting value for the S increment ΔS.

- **min value of delta-S**

  If ΔS is too large, the problem obtained will not converge. When this happens, ΔS will be decreased, and a new problem tried. To prevent an infinite loop, you should select a lower bound for ΔS here. If this lower bound is reached, the evolution process is terminated.

- **max value of delta-S**

  This sets an upper bound for ΔS.

- **max number of successful steps**

  Here you can set the upper bound on the number of steps used to reach the original problem.

You can also choose the integration method for the evolution calculation. These methods are equivalent to those used in time-dependent flows, and are described in *Theory* (p. 539) :

- **0-order method** (explicit Euler)
- **implicit Euler method** (the default scheme)
- **Galerkin method**
- **Crank-Nicolson method**

26.4.4. Initial Conditions

By default, ANSYS POLYFLOW automatically calculates an initial solution by evaluating at $S = S_{init}$ all parameters for which an S dependency has been declared. If $S_{init} = 0$ (the default value in ANSYS POLYDATA), and you are using the evolution function $1/S$, ANSYS POLYDATA will automatically change S_{init} to 0.001 and issue a warning message.

In free surface problems, an initial solution is calculated on the initial mesh; the kinematic condition $\mathbf{v} \cdot \mathbf{n} = 0$ is dropped but the force boundary condition is maintained. See *Theory and Equations* (p. 311) for details.

If a previously-saved results file (called `res`, by default) is available, then ANSYS POLYFLOW can read the initial solution from it. See *Starting an Evolution or Time-Dependent Calculation from Existing Results and Restart Files* (p. 109) for details. With a corresponding restart file (which is automatically saved during an evolution calculation and called `rst` by default), ANSYS POLYFLOW can also read the S derivative of the solution. Without an `rst` file, ANSYS POLYFLOW will not be able to perform a first-order or second-order continuation, and will not find the old value of S. Should you want to continue the calculation starting from the `res` file anyway, you need to first modify the initial values of S and dS in the **Numerical parameters** menu. Specifying the values that correspond to the `res` file is almost equivalent to using an `res` file and an `rst` file. The only difference is that the S derivative of the solution is not available and the convergence of the continuation may be different.

With an `rst` file, ANSYS POLYFLOW will find the last value of S and be able to restart exactly at the level where the program was interrupted. Typically, when the second-order Adams-Bashforth scheme is used, the S derivatives at two previous evolution steps are used to extrapolate the solution. When the first-order explicit Euler scheme is used, the derivative at the previous evolution step is used for the extrapolation.

When the zero-order scheme is used, in order to avoid small evolution step, the initial evolution step, `dsini` is the maximum between the value given by the user and the one read in the `rst` file. You should never modify problem data such as model parameters or boundary conditions when starting from an `rst` file. Any deviation will result in an ill-posed problem.

26.4.5. Output of Results for Evolution Problems

For evolution calculations, you can ask ANSYS POLYFLOW to save the output files at specified intervals during the calculation. This will allow you to examine the progress of the solution, and study the evolution history of solution variables. See *Output for Time-Dependent and Evolution Calculations* (p. 126) for more information.

26.5. Interrupting Evolution

During an evolution or a transient scheme, you may be interested in interrupting the evolution when one or several criteria related to a field are satisfied. Examples of such criteria include when the minimum of temperature reaches a lower limit:

$$\min\left(T \text{ on } \Omega\right) < T_{min}$$

or when the norm of the velocity along the wall reaches a lower limit during an evolution on the slip coefficient:

$$\min\left(\|\mathbf{v}\| \text{ on wall } \right) < v_{min}$$

In the sections that follow, the criteria are related to fields, and the interruption of the evolution due to convergence trouble will not be addressed.

The criteria are evaluated at the end of a successful step. If the criteria are satisfied, the evolution is interrupted. Otherwise, the evolution continues.

26.5.1. Criterion Definition

A criterion for interrupting an evolution scheme is based on the values of a field $\mathbf{X}$ in a domain Ω. The set of values of $\mathbf{X}$ in the domain Ω will be noted as $\mathbf{X}_\Omega$.

Important

Note that though fields can be either a vector or a scalar, for the sake of simplicity they will always be denoted as $\mathbf{X}$ in this section and the sections that follow.

If the field is not a scalar, you have to use a restriction function $R\left(\mathbf{X}_\Omega\right)$ to convert the non-scalar value to a scalar one, which is then called the restricted value and noted r_n. The subscript n refers to the node index.

The value to check is obtained by applying a function $F\left(R\left(\mathbf{X}_\Omega\right)\right)$ on the restricted value $R\left(\mathbf{X}_\Omega\right)$ or r_n.

Eventually, an inequality test $T\left(F\left(R\left(\mathbf{X}_\Omega\right)\right), L\right)$ is applied to check the value of the field with respect to a limit L.

Thus, a criterion is written has follows:

$$T\left(F\left(R\left(\mathbf{X}_{\Omega}\right)\right),L\right) \tag{26–1}$$

For the criteria $\min\left(T \text{ on } \Omega\right) < T_{min}$:

- the field is temperature on the domain Ω: $\mathbf{X}_{\Omega} = T \text{ on } \Omega$
- there is no restriction function,
- the function for obtaining the check value is the minimum, $F\left(r_n\right) = \min\left(r_n\right)$
- the inequality test is “less than".

For the criteria $\min\left(\|\mathbf{v}\| \text{ on wall}\right) < v_{min}$:

- the field is velocity along the wall: $\mathbf{X}_{\Omega} = \mathbf{v} \text{ on wall}$
- the restriction function is the norm: $R\left(\mathbf{v}\right) = \|\mathbf{v}\|$
- the function for obtaining the check value is the minimum, $F\left(r_n\right) = \min\left(r_n\right)$
- the inequality test is “less than".

26.5.2. Available Fields (X) for Criteria

The fields available for a criterion for interrupting an evolution scheme depend upon the physical models involved in the simulation. The following fields can be selected for a criterion:

- Output fields of a main task:
  - Coordinates
  - Velocity
  - Pressure
  - Temperature
  - Thickness for shell elements or films
  - Velocities for porous media
  - Pressure for porous media
  - Displacement
  - Electric potential
  - Species
  - Density (if PMAT for density has been defined)
- Output fields of postprocessors:
  - Stream function (*Stream Function* (p. 552))
  - Local shear rate (*Local Shear Rate* (p. 553))
  - Viscosity (*Viscosity* (p. 553))
  - Viscous heating (*Viscous Heating and Dissipated Power* (p. 555))
  - Residence time (*Residence Time* (p. 556))

- Material points (*Tracking of Material Points* (p. 558))
- Material property (*Tracking of a Material Property* (p. 560))
- Flow rate (*Flow Rate* (p. 563))
- Joule effect (*Joule Effect* (p. 563))
- Mixing index (*Mixing Index* (p. 563))
- Vorticity (*Vorticity* (p. 564))
- Convected heat (*Convected Heat* (p. 565))
- Eigen values (*Calculation of Eigenvalues* (p. 568))
- Stress component along the velocity direction (*Calculation of the Stress Component Along the Velocity Direction* (p. 569))
- Flow balance (*Quantification of Die Balancing* (p. 569))

26.5.3. Restriction Functions R

The restriction function type depends upon the tensor type of field $\mathbf{X}$. The following are the restriction functions $R(\mathbf{X}_{\Omega})$:

- Current value: $R(\mathbf{X}) = \mathbf{X}$

  This function consists of the current value of $\mathbf{X}$. This function can only be applied when $\mathbf{X}$ is a scalar field. This is selected automatically, since the values of the scalars are detected.

- Norm of vector: $R(\mathbf{X}) = \|\mathbf{X}\| = \sum_{i=1}^{N} (\mathbf{X}_i)^{2}$

  This function computes the norm of the vector X. This function can only be applied when $\mathbf{X}$ is a vector field.

- i^{th} component of vector or tensor: $R(\mathbf{X}) = \mathbf{X}_i$ This function extracts the i^{th} component of a vector or a tensor.

26.5.4. Functions for Obtaining the Check Value F

The following is a list of the functions that can be used to obtain the value to check:

- Minimum: $F(r_n) = \min(r_n)$

  This function computes the minimum of the restricted values r_n.

- Maximum: $F(r_n) = \max(r_n)$

  This function computes the maximum of the restricted values r_n.

- Average: $F(r_n) = \overline{r_n} = \frac{\sum_{i=1}^{N} r_n}{N}$

  This function computes the average of the restricted values r_n. This average is based on the nodal values: the sum of nodal values divided by the number of nodal values (N).

- At closest node: $F(r_n) = r(x, y, z)$

This function extracts the value of restricted values r_n at the node closest to the point of coordinates (x, y, z).

- Weighted average: $F(r_n) = \overline{\langle r_n \rangle} = \dfrac{\int_\Omega r d\Omega}{\int_\Omega d\Omega}$

  This function computes the weighted average of the restricted values r_n. This weighted average is obtained from the ratio of the integral of r over the domain Ω and the length-surface-volume of Ω.

- Integral: $F(r_n) = \langle r_n \rangle = \int_\Omega r d\Omega$

  This function computes the integral of the restricted values r_n over the domain Ω.

Note that the function for computing the check value is irrelevant for fields having only one value. This is the case for fields whose interpolation type is **constant over the domain** (flow rate, viscous heating), or if the domain over which the criteria is evaluated is a point.

26.5.5. Coordinate Functions

Besides the functions listed in the previous section, there is a special function for the coordinates. This function computes the length, the area, or the volume of a domain with 1, 2, or 3 dimensions, respectively. This function is only available if the coordinates have been coordinates for the field.

Table 26.2: List of Check Values (p. 530) lists the values to check by combining the restriction functions R and the functions F to obtain the check values.

Table 26.2 List of Check Values

		Restriction Functions		
		X (Scalar)	$\|\mathbf{X}\|$	$\mathbf{X}_i$
Functions for Obtaining the Extracted Value	Minimum	$\min(X)$	$\min(\|\mathbf{X}\|)$	$\min(\mathbf{X}_i)$
	Maximum	$\max(X)$	$\max(\|\mathbf{X}\|)$	$\max(\mathbf{X}_i)$
	Average	$\overline{X}$	$\overline{\|\mathbf{X}\|}$	$\overline{\mathbf{X}_i}$
	Weighted Average	$\overline{\langle X \rangle}$	$\overline{\langle \|\mathbf{X}\| \rangle}$	$\overline{\langle \mathbf{X}_i \rangle}$
	Closest Node	$X(x,y,z)$	$\|\mathbf{X}(x,y,z)\|$	$\mathbf{X}_i(x,y,z)$
	Integral	$\langle X \rangle$	$\langle \|\mathbf{X}\| \rangle$	$\langle \mathbf{X}_i \rangle$
Coordinate Functions	Length			
	Area			
	Volume			

26.5.6. Inequality Tests

The choices for the inequality tests are listed below:

- $F\ (r_n)\ \ > L$ (greater than)
- $F\ (r_n)\ \ \geq L$ (greater than or equal to)
- $F\ (r_n)\ \ < L$ (less than)
- $F\ (r_n)\ \ \leq L$ (less than or equal to)

26.5.7. Multiple Criteria

You can define several criteria for interrupting an evolution scheme. When this is the case, these criteria are gathered in one or several groups of criteria. A group of criteria will be considered satisfied if all criteria belonging to the group are satisfied. This corresponds to an “and" condition between criteria within a group. For the evolution to be interrupted, at least one group of criteria must be satisfied. This corresponds to an “or" condition between the groups.

26.5.8. Inputs for Criteria to Interrupt the Evolution

Note that in ANSYS POLYDATA, the group of criteria described in *Multiple Criteria* (p. 531) is called “criterion", while the criterion described in *Criterion Definition* (p. 527) is called “sub-criterion".

After you have defined your simulation, you can specify the criteria to interrupt the evolution in the **Numerical parameters** item from the **F.E.M. Task** menu.

Numerical parameters

In the **Numerical parameter** menu, select one of the following:

Modify the evolution parameters

or

Modify the transient iterative parameters

These open the **Evolution parameters** menu and the **Transient iterative parameters** menu, respectively, where you can select the **Management of interruption criteria** item. This will bring up the **Management of interruption criteria** menu.

1. If you want to add a new group of criteria, click the **Add a criterion** item.

   Add a criterion

   a. The new group of criteria receives a name: **Stop # [index]**. You can change the name of the group of criteria by clicking **Change name of the criterion**.

      Change name of the criterion

      When prompted, enter the new name of the group of criteria.

b. To add a new criterion to the group, click the **Add a sub-criterion** item.

Add a sub-criterion

A criterion is defined by default. For example:

If the minimum of norm of field COORDINATES on domain R,

is lower or equal to 0,

the criterion becomes TRUE.

This is just an example, because the default definition for the criterion depends upon the fields computed during the simulation. In order to modify the criterion, perform the following steps:

i. Click the **Select a field** item, to select or modify the $\mathbf{X}$ field.

Select a field

ANSYS POLYDATA will provide you with choices from different fields. Click the item corresponding to your choice, and then click the **Upper level menu** item on the top of the menu after you have selected the field.

ii. Click the **Select a restriction of the field** item to select or modify the restriction function R. This item is available for non-scalar fields only.

Select a restriction of the field

- If you have selected the coordinates for the field, you can compute the length, area, or volume of the domain by clicking the **length/surface/volume of domain** item.

  length/surface/volume of domain

- Select the **Norm (magnitude)** item if you want to compute the norm of a vector for the restriction. This is the default for vectors.

  Norm (magnitude)

- Select the item related to a component (**first component of field**, **second component of field**, etc.) if you want to extract a component of a vector or a tensor for the restriction.

  first component of field

Click the **Upper level menu** item after you have selected the restriction function.

iii. Click the **Select a function** item to select or modify the function to compute the check value.

Select a function

> **Important**
>
> Note that this item is only available if the restriction function is not **length/surface/volume of domain**.

- Select the **minimum value** item if you want to compute the minimum of restricted values r_n.

  minimum value

- Select the **maximum value** item if you want to compute the maximum of restricted values r_n.

  maximum value

- Select the **average value** item if you want to compute the average of restricted values r_n.

  average value

- Select the **value at position** item if you want to extract the value of restricted values r_n, at the node closest to a particular position.

  value at position

  When prompted, enter the coordinates of the position.

- Select the **weighted average value** item if you want to compute the weighted average of restricted values r_n.

  weighted average value

- Select the **integral value** item if you want to compute the integral of restricted values r_n.

  integral value

Click the **Upper level menu** item after you have selected the function to compute the check value.

iv. Click the **Select a domain** item to select or modify the domain where the criteria will be evaluated. By default, the selected domain is the domain associated with the field.

Select a domain

- Select the **Define domain of the sub-criterion** item in order to select or modify the domain where the criterion will be evaluated. By default, the domain of the criterion is the domain associated with the field.

  Define domain of the sub-criterion

The top list displays the current domain. If you want to change this, select a domain in the bottom list and click the **Add** menu item. The top list will be updated and will contain the selected domain. Only one topo-object (domain or boundary) can be selected.

After you have selected the domain where the criteria will be evaluated, click the **Upper level menu** menu item.

- If the desired domain or boundary is not displayed, you can create a new one by clicking the **Create a new topo-object** item.

Create a new topo-object

The top list will display all of the existing topo-objects.

Select an initial topo-object in the top list, and click the **Select object 1** menu item to confirm your choice.

Select a second topo-object in the top list, and click the **Select object 2** menu item to confirm your choice.

Select the type of operation from the following choices: **intersection** to create a new region from the intersection of the selected topology objects; or **union** to create a new region that is a combination of the selected topology objects.

Finally, specify a name and click the **create** menu item to create the new topo-object.

After you have completed these steps, the new region will be accessible in the top list and may be used to create yet another new region. This new region will be also accessible from the top list of the **Define domain of sub-criterion** panel, and it may be selected as a domain where the criterion will be evaluated.

Note that you can only perform topological operations between topo-objects of the same dimension.

- The name of a topo-object may be modified by selecting the **Modify the name of a topo-object** item.

Modify the name of a topo-object Item

In the top list, select the topo-object whose name you want to modify.

Click the **Select object** menu item to confirm your choice.

Enter the new name and click the **modify** menu item to change the name.

Finally, click the **Upper level menu** menu item.

- You can delete a topo-object by selecting the **Delete a topo-object** item.

Delete a topo-object Item

In the top list, select the topo-object for deletion.

Click the **Select object** menu item to confirm your choice.

Click the **delete** menu item and click the **Upper level menu** menu item.

After you have specified the domain where the criteria will be evaluated, click the **Upper level menu** item.

v. Click the **Select the type of criterion** item to select or modify the inequality test.

Select the type of criterion

- Click the **greater than** item if you want apply the test: $F\ (r_n)\ > L$.

  greater than

- Click the **greater or equal to** item if you want apply the test: $F\ (r_n)\ \geq L$.

  greater or equal to

- Click the **lower than** item if you want apply the test: $F\ (r_n)\ < L$.

  lower than

- Click the **lower or equal to** item if you want apply the test: $F\ (r_n)\ \leq L$.

  lower or equal to

After you have specified the type of inequality test, click the **Upper level menu** item.

vi. Click the **Modify the limit value** item to set the limit value of the criteria.

Modify the limit value

When prompted, specify the new limit value.

c. To modify an existing criterion of the group, click the **Modify sub-criterion # [index]** item. For example, to modify the first criterion of the group, select the following:

Modify sub-criterion # 1

A menu will appear which has the same items as the menu described previously for **Add a sub-criterion**. You can modify the field, the restriction function, the function to compute the check value, the domain where the criteria will be evaluated, the inequality test, and the limit value.

d. If you want to remove a criterion from the group, click the **Remove a sub-criterion** item.

Remove a sub-criterion

Select the criterion for removal from the group, and click the **Upper level menu** item.

e. After all of the criteria of the group have been defined, click the **Upper level menu** item.

2. To modify an existing group of criteria, click the **Modify criterion [criterion name]** item. For example, to modify the first criterion when it has the default name, select the following:

Modify criterion Stop # 1

A menu will appear which has the same items as the menu described previously for **Add a criterion**. You may change the name of the group of criteria, add a new criterion to the group, or modify or remove an existing criterion of the group.

3. If you want to remove an existing group of criteria, click **Remove a criterion**.

   Remove a criterion

   Select the criterion for removal, and click the **Upper level menu** item.

4. After all of the groups of criteria have been defined, click the **Upper level menu** item.

   Upper level menu

26.5.9. Output for Interruption Criteria

The list of criteria will be printed at the beginning of the ANSYS POLYFLOW simulation, and hence, at the beginning of the listing file. The list will appear before the beginning of evolution, as shown in the following example:

```
***************************************
*                                     *
*  CRITERIA TO INTERRUPT EVOLUTION    *
*                                     *
***************************************

The evolution will be interrupted if :

          min(TEMPERATURE) on (S1*B1).
                  <= 70.000E+00


 ===> Starting Time marching
         at t-ini = 0.0000000E+00
 =================================
*************************************
*** Starting Step  1              ***
*************************************
...
```

At the end of the step, you can find the information about the current status of the criteria. If none of the groups of criteria are satisfied, the following will be displayed:

```
*** time step information : ***
===============================
t      = 0.1000000E-01, step # 1
dtnew  = 0.1000000E-01, okincr : T
tests  = 0.2640593E-01

*********************************
*                               *
*  INTERRUPTION OF EVOLUTION    *
*                               *
*********************************

The evolution continues :

All criteria are not satisfied:
       min(TEMPERATURE) on (S1*B1).
       116.832E+00 < 70.000E+00 , criterion is : F
```

```
*************************************
*** Starting Step  2              ***
*************************************
```

Otherwise, if a group of criteria is satisfied, the following is displayed:

```
*** time step information : ***
===============================
t      = 0.6960801E+00, step #  13
dtnew  = 0.1536490E+00, okincr : T
tests  = 0.1534719E-01

********************************
*                               *
*  INTERRUPTION OF EVOLUTION    *
*                               *
*********************************

The evolution is interrupted :

All criteria are satisfied:

    min(TEMPERATURE) on (S1*B1).
    69.978E+00 < 70.000E+00 , criterion is : T

Sfin has not been reach.  S = 0.6960801E+00
                       Sfin = 0.2000000E+02
===========================================
```

Finally, at the end of the listing, you will find the following message if the evolution or the transient scheme has been interrupted by criteria:

```
********************************
*  Summary of the simulation   *
********************************

 The computation succeeded.
 The evolution scheme has been interrupted because the criteria are
  satisfied.
```

The criteria status is also accessible in ANSYS POLYDIAG, as described in *Interrupting Evolution* (p. 589).

Chapter 27: Time-Dependent Flows

Information on solving time-dependent flow problems in ANSYS POLYFLOW is presented in the following sections:

27.1. Introduction

Time-dependent flow problems are characterized by the presence of one (or more) time derivative in the basic equations, and are distinct from steady-state and evolution problems. Time-dependent flow problems can have time-dependent inlet flow rates, boundary conditions, material parameters, etc. In other respects, the problems are similar to steady-state problems, so only the time-dependent aspects will be discussed in this chapter.

Time-dependent flows may or may not reach a steady-state solution, depending on flow parameters and boundary conditions. Because of the intrinsic nonlinearity of most flow problems, it is not possible, in general, to predict whether a transient flow will or will not lead to a steady-state flow regime.

In ANSYS POLYFLOW the time dependence of material data and boundary conditions is defined in terms of a time parameter *t*, in much the same way as evolution problems (see *Evolution* (p. 517)) are defined in terms of *S*. The numerical parameters for controlling the time-marching scheme are similar to those used in evolution. The fundamental difference between time-dependent and evolution problems is that the evolution scheme is controlled by convergence of the iterative scheme, while the time-marching scheme is controlled by the convergence and the accuracy of the time-integration technique.

27.2. Theory

27.2.1. Equations

Time-dependent flow problems are governed by the following set of ordinary differential equations:

$$\mathbf{M}(\mathbf{X})\,\dot{\mathbf{X}} + \mathbf{K}(\mathbf{X})\,\mathbf{X} + \mathbf{F}(\mathbf{X}) = \mathbf{0} \qquad \textbf{(27–1)}$$

which is subject to initial conditions of the type

$$\mathbf{X}(t_0) = \mathbf{X}_0 a \qquad \textbf{(27–2)}$$

$\mathbf{X}$ is the vector of nodal unknowns such as velocity, pressure, temperature, viscoelastic extra stresses, and free surface location. The symbol $\dot{\mathbf{X}}$ denotes the time-derivative of $\mathbf{X}$. The matrices $\mathbf{M}$ and $\mathbf{K}$ are the mass and stiffness matrices, which may depend on the unknown vector $\mathbf{X}$. The vector $\mathbf{F}$ corresponds to the volumetric forcing function and the natural boundary conditions.

The above equations are solved in ANSYS POLYFLOW by means of a parabolic time-stepping procedure. Instead of trying to satisfy *Equation 27–1* (p. 539) at an arbitrary time t, ANSYS POLYFLOW will calculate a solution of *Equation 27–1* (p. 539) at a discrete set of times t_n, defined by

$$\mathbf{X}_n = \mathbf{X}(t_n) \qquad \textbf{(27–3)}$$

$$t_n = t_{n-1} + \Delta t_n \qquad \textbf{(27–4)}$$

where the subscript n refers to the time step.

From *Equation 27–1* (p. 539), the time derivative $\dot{\mathbf{X}}$ can be obtained. Consider the function $f(\mathbf{X})$:

$$f(\mathbf{X}) = -\mathbf{M}^{-1}(\mathbf{KX}+\mathbf{F}) = \dot{\mathbf{X}} \qquad \textbf{(27–5)}$$

An approximation of $\dot{\mathbf{X}}$ is created using the formula

$$\dot{\mathbf{X}} \approx \theta f(\mathbf{X}_{n+1}) + (1-\theta) f(\mathbf{X}_n), \ (0 \le \theta \le 1) \qquad \textbf{(27–6)}$$

Also, using the first-order discretization of the first derivative

$$\dot{\mathbf{X}} = \frac{\mathbf{X}_{n+1} - \mathbf{X}_n}{\Delta t_n} \qquad \textbf{(27–7)}$$

results in

$$\mathbf{X}_{n+1} = \mathbf{X}_n + \Delta t_n \dot{\mathbf{X}} = \mathbf{X}_n + \Delta t_n (\theta f(\mathbf{X}_{n+1}) + (1-\theta) f(\mathbf{X}_n)) \qquad \textbf{(27–8)}$$

Different values of θ result in different integration methods with different accuracy and stability attributes.

27.2.2. Integration Methods

The integration methods available in ANSYS POLYFLOW are summarized in *Table 27.1: Integration Methods in ANSYS POLYFLOW* (p. 540).

Table 27.1 Integration Methods in ANSYS POLYFLOW

Integration method	θ	precision
Explicit Euler	0	$O(\Delta t)$
Crank-Nicolson	$\frac{1}{2}$	$O(\Delta t^2)$

Galerkin	$\frac{2}{3}$	$O(\Delta t)$
Implicit Euler	1	$O(\Delta t)$

With the explicit Euler method ($\theta = 0$), $\mathbf{X}_{n+1}$ can be evaluated directly from the known value of $\mathbf{X}$ at the previous time t_n (and, hence, the method is explicit). The other methods, with nonzero θ, are known as implicit, since $\mathbf{X}_{n+1}$ depends not only on $\mathbf{X}_n$ at the previous time, but also on the time derivative of $\mathbf{X}$ at t_{n+1}, which is itself a function of the unknown $\mathbf{X}_{n+1}$. Since $\mathbf{X}_{n+1}$ appears on both sides of *Equation 27–8* (p. 540), using an implicit method requires solving for $\mathbf{X}_{n+1}$.

From this point of view, the implicit method requires more operations. However, all explicit techniques are only conditionally stable: the time step size Δt must not exceed a certain value. The maximum admissible Δt of an explicit scheme is related to the size of elements and the speed of the trajectories. With an explicit technique, the time step should not exceed the time required for a trajectory to pass through an element (for all elements in the mesh).

ANSYS POLYFLOW offers only implicit time-marching schemes of the predictor-corrector family. The predictor is an explicit time-marching scheme that gives an estimate of $\mathbf{X}_{n+1}$, denoted hereafter by $\mathbf{X}^p_{n+1}$. This value is an initial guess for the corrector. The corrector itself might be nonlinear, in which case several iterations may be needed for convergence. However, in the limit of small time steps, the mass matrix terms dominate the system (*Equation 27–1* (p. 539)) and all problems become linear. Predictor-corrector methods enjoy the stability properties of implicit techniques. Another advantage is the automatic control of time step. This is based on the difference between the predicted value $\mathbf{X}^p_{n+1}$ and the corrected value $\mathbf{X}_{n+1}$. After the solution at $t = t_{n+1}$, an estimate of the next time step size Δt_{n+2} can be calculated using the formula

$$\Delta t_{n+2} = \Delta t_{n+1}\left(\frac{a\varepsilon}{d_{n+1}}\right)^b \qquad \textbf{(27–9)}$$

where the coefficients a and b depend on the order of the implicit method, and d_{n+1} is the maximum relative difference between the predicted and corrected value of $\mathbf{X}_{n+1}$:

$$d_{n+1} = \max_i \frac{|\mathbf{X}_{i\,(n+1)} - \mathbf{X}^p_{i\,(n+1)}|}{|\mathbf{X}_{i\,(n+1)}|} \qquad \textbf{(27–10)}$$

where $\mathbf{X}_{i\,(n+1)}$ is the ith component in the vector $\mathbf{X}_{n+1}$. The only user-defined parameter in *Equation 27–9* (p. 541) is ε, which represents a relative tolerance of the local truncation error in $\mathbf{X}_{n+1}$ by comparison with the exact solution $\mathbf{X}(t_{n+1})$. The choice of ε has a significant effect on computational cost and accuracy.

27.2.3. Internal Solution Strategy

The following is a description of the solution strategy as implemented in ANSYS POLYFLOW.

1. Calculate the predicted value $\mathbf{X}^{p}_{n+1}$ using an explicit scheme:

$$\mathbf{X}^{p}_{n+1} = \mathbf{X}_n + \Delta t_n \dot{\mathbf{X}}_n \qquad \textbf{(27–11)}$$

2. Calculate the corrected value $\mathbf{X}_{n+1}$ using an implicit scheme:

    a. Initialize $\mathbf{X}^{(0)}_{n+1} = \mathbf{X}^{p}_{n+1}$.

    b. Solve for $\mathbf{X}_{n+1}$ with the iteration

    $$\left(\frac{\mathbf{M}}{\theta \Delta t_n} + \mathbf{K}\right) \mathbf{X}^{(s+1)}_{n+1} = \mathbf{M}\left(\frac{\mathbf{X}_n}{\theta \Delta t_n} + \frac{1-\theta}{\theta}\dot{\mathbf{X}}_n\right) - F^{(s)}_{n+1} \qquad \textbf{(27–12)}$$

    This equation is identical in form to *Equation 27–8* (p. 540).

    c. If this converges, continue to step 3. Otherwise, divide Δt_n by 2, and return to step 1, unless Δt_n is less than the minimum step size, in which case stop the calculation.

3. Accept or reject the corrected value on the basis of the precision:

$$\max_i \left| \mathbf{X}_{i\,(n+1)} - \mathbf{X}^{p}_{i\,(n+1)} \right| < \frac{\varepsilon}{10} \qquad \textbf{(27–13)}$$

    Otherwise, reject $\mathbf{X}_{n+1}$ and return to step 1.

4. Estimate the next time step using *Equation 27–9* (p. 541). Compare the new value of t_{n+2} with $1.5\Delta t_{n+1}$ and keep the smaller value.

5. Return to step 1 for the solution at time $t = t_{n+2}$.

These steps are repeated until the upper time limit is reached, Δt_n falls below the minimum step size (Step 2.c.), or the maximum number of successful steps is reached.

27.2.4. Time-Marching Schemes

In ANSYS POLYFLOW three time-marching schemes are available.

- implicit Euler method

    You can select the implicit Euler method as the corrector in the time-integration procedure. ANSYS POLYFLOW automatically uses the explicit forward Euler formula (*Equation 27–11* (p. 542)) as the predictor.

    Substituting $\theta = 1$ into *Equation 27–12* (p. 542), the following set of nonlinear algebraic equations corresponding to the implicit Euler method is obtained:

$$\frac{1}{\Delta t_n}\mathbf{M}\left(\mathbf{X}_{n+1}-\mathbf{X}_n\right)+\mathbf{K}\mathbf{X}_{n+1}+\mathbf{F}_{n+1}=\mathbf{0} \tag{27–14}$$

The implicit Euler method is a first-order scheme, i.e., the time-integration accuracy is O (Δt). It does not cause oscillatory behavior, no matter how large the time-step size is.

The implicit Euler method is the default method.

- Galerkin method

  With this method, the corrector is obtained by setting $\theta = 2/3$ in *Equation 27–12* (p. 542), which yields

$$\frac{3\mathbf{M}}{2\Delta t_n}\left(\mathbf{X}_{n+1}-\mathbf{X}_n\right)+\mathbf{K}\mathbf{X}_{n+1}+\mathbf{F}_{n+1}-\frac{\mathbf{M}}{2}\dot{\mathbf{X}}_n=\mathbf{0} \tag{27–15}$$

  The explicit forward Euler formula (*Equation 27–11* (p. 542)) is used, as with the implicit Euler method.

  The Galerkin method is more accurate than the implicit Euler method. However, it can generate oscillatory errors if the time step is large, though these are not as troublesome as with the Crank-Nicolson method. The Galerkin method is a compromise between the implicit Euler and Crank-Nicolson methods.

- Crank-Nicolson method

  This method is also called the modified Euler method or the trapezoid rule. It is a second-order method with an accuracy of O (Δt^2). It is advisable to use a predictor of the same accuracy, so the following Adams-Bashforth explicit formula is used in ANSYS POLYFLOW:

$$\mathbf{X}^p_{n+1}=\mathbf{X}_n+\frac{\Delta t_n}{2}\left(\left(2+\frac{\Delta t_n}{\Delta t_{n-1}}\right)\dot{\mathbf{X}}_n-\frac{\Delta t_n}{\Delta t_{n-1}}\dot{\mathbf{X}}_{n-1}\right) \tag{27–16}$$

  The corrector corresponding to the Crank-Nicolson method has been obtained by setting $\theta = 1/2$ in *Equation 27–12* (p. 542), which yields

$$\frac{2}{\Delta t_n}\mathbf{M}\left(\mathbf{X}_{n+1}-\mathbf{X}_n\right)+\mathbf{K}\mathbf{X}_{n+1}+\mathbf{F}_{n+1}-\mathbf{M}\dot{\mathbf{X}}_n=\mathbf{0} \tag{27–17}$$

  The Adams-Bashforth explicit formula (*Equation 27–16* (p. 543)) requires the knowledge of the time-derivative of $\mathbf{X}$ at two previous time steps: $\dot{\mathbf{X}}_n$ and $\dot{\mathbf{X}}_{n-1}$. It cannot be used until the completion of the first three time steps. Therefore, the time step adjustment as a function of the local truncation errors can only begin at the 4th step. For this reason, the initial time step should not be too large, since there is no error control during the first three time steps.

  The Crank-Nicolson method is the most accurate of the three methods available in ANSYS POLYFLOW. A drawback of this method is the generation of oscillatory errors when the time step is large. To avoid this, you should set the tolerance (i.e., ε) quite small.

27.3. User Inputs for Time-Dependent Problems

Time-dependent problems are set up in much the same way as steady-state problems. The primary difference is the designation of certain parameters (material parameters, boundary conditions, etc.) as time-dependent. The setting of time dependence is very similar to the steps used to define an evolution problem (see *Evolution* (p. 517)).

27.3.1. Setting Up a Time-Dependent Problem

Below is an outline of the steps for solving a time-dependent problem that differ from the usual problem setup steps.

1. Determine which parameters in your problem are time-dependent. These can be material parameters (such as density) or boundary conditions.
2. In the **Create a new task** menu, specify a time-dependent problem, and define the geometry type, etc., in the usual way.

   Time-dependent problem(s)
3. To set a time-dependent material parameter or a time-dependent boundary condition, the steps are slightly different:

   - For a time-dependent material parameter, follow these steps to define the parameter's value as $xf(t)$:

     a. Go to the menu appropriate for setting the value of this parameter. For instance, if the parameter is density, select the **Density** menu item in the **Material data** menu.

        Density

     b. Click the **EVOL** button at the top of the ANSYS POLYDATA menu to enable the time-dependence inputs.

        The **EVOL** button will change to **EVOL [on]** to indicate that the time-dependence inputs are enabled.

     c. Select the menu item appropriate for the time-dependent parameter. For density, this would be **Modification of density**.

        Modification of density

     d. Enter x when ANSYS POLYDATA prompts for the **New value**.
     e. Click **OK** to accept this new value.
     f. A menu will appear from which you will select the function f. The available functions are the same as those used for evolution, and are described in *Available Evolution Functions* (p. 518).

        If several parameters are time-dependent, a different function f can be chosen for each one.

     g. Select **Upper level menu** to complete the specification.
     h. Click the **EVOL** button again to disable the inputs for time dependence.

The **EVOL** button will change to **EVOL [off]** to indicate that the time-dependence inputs are disabled.

- For a time-dependent boundary condition, follow these steps to define the parameter’s value as $xf\ (t)$:

  a. Select **Flow boundary conditions** or **Thermal boundary conditions**, as appropriate, from the sub-task menu.

  b. Select the boundary condition you want to set, and click **Modify**.

  c. Click the **EVOL** button at the top of the ANSYS POLYDATA menu to enable the time-dependence inputs.

     The **EVOL** button will change to **EVOL [on]** to indicate that the time-dependence inputs are enabled.

  d. Select the type of boundary condition you want to impose. For instance, for a flow boundary condition, you might choose **Normal and tangential velocities imposed (vn, vs)**.

  e. After you enter the value (x) of each numerical parameter (such as v_n) associated with this boundary condition, clicking **Upper level menu** will cause a menu to appear, from which you can select the function f. The available functions are the same as those used for evolution, and are described in *Available Evolution Functions* (p. 518).

     If several parameters are time-dependent, a different function f can be chosen for each one.

     Note that boundary conditions can be position-dependent and time-dependent. That is, x may be a constant, or a function of coordinates.

  f. Select **Upper level menu** to complete the specification.

  g. Click the **EVOL** button again to disable the inputs for time dependence.

     The **EVOL** button will change to **EVOL [off]** to indicate that the time-dependence inputs are disabled.

Repeat this step for each time-dependent parameter.

4. Continue setting up the problem, following the usual procedures.
5. After all sub-tasks have been defined, select **Numerical parameters** from the **F.E.M. Task Menu**.

Numerical parameters

In the **Numerical parameters** menu, select **Modify the transient iterative parameters**. This will bring up the **Transient iterative parameters** menu, which contains the following items:

- **Modify the initial time value**

  This allows you to modify the time at which the time-marching procedure starts. The initial time corresponds to t_0 in *Equation 27–2* (p. 539). The vector $\mathbf{X}$ is initialized to a default initial solution calculated by ANSYS POLYFLOW (time-dependent parameters will then be evaluated at t_0), unless you choose to read an initial solution from a file (i.e., start from a results file, as described in *Starting an ANSYS POLYFLOW Calculation from an Existing Results File* (p. 109)).

  The default value of the initial time is 0.

- **Modify the upper time limit**

  This item allows you to specify the time at which the solution procedure stops.

- **Modify the initial value of the time-step**

  Use this item to set the initial time step Δt_1. The first solution is calculated at $t_0 + \Delta t_1$. The initial value of the time step will be used during the first three time steps; i.e., Δt_2 and Δt_3 are equal to Δt_1 unless the simulation does not converge, in which case the time step is usually divided by 2. Thereafter, the time step size will be controlled automatically by ANSYS POLYFLOW according to *Equation 27–9* (p. 541). Since accuracy cannot be checked for the first three time steps, it is recommended that a small value of the initial time step be used.

  The default value for the initial time step is 0.01, and is, in general, a good choice, although this parameter depends upon the time scale of the process; it may become inappropriate for very fast or very slow processes.

- **Modify the min value of the time-step**

  ANSYS POLYFLOW will stop if the new value of Δt is less than this allowed minimum value.

- **Modify the max value of the time-step**

  This is the maximum time-step size that ANSYS POLYFLOW will accept. The default value is 0.25. You will want to modify this value in certain cases:

  - when you want to solve a problem with a constant time step. It is sufficient to assign a unique value for both the minimum and the maximum time step, and to select a large value for the tolerance.
  - when you want result output files at more time intervals than you would get with the default time step. Set the maximum value for the time step to a smaller value than the default to achieve this.
  - for very slow processes when you do not need to get the results every quarter second.

- **Modify the tolerance**

  The tolerance, ε, represents the maximum admissible relative truncation error in the numerical solution $\mathbf{X}_{n+1}$ with reference to the exact solution $\mathbf{X}(t_{n+1})$.

  The choice of ε depends on the time-integration method and can have a significant effect on computation cost and accuracy. The automatic control of time step Δt is directly related to ε. Too small an ε can result in overly expensive computation, although a high time-accuracy can be reached. On the other hand, setting ε too large can cause one or more of the following problems:

  - significant departure from the assumptions used in the model derived from the theory of *Theory* (p. 539)
  - oscillatory behavior (in the case of the Crank-Nicolson and Galerkin methods)
  - divergence of nonlinear problems (due to the fact that *Equation 27–8* (p. 540) will no longer be dominated by the mass matrix terms)

  The default value of ε is 0.001, which is usually the optimal value for the Crank-Nicolson method. For the implicit Euler method, you can use a larger value; the recommended value is 0.01. For the Galerkin method, a value between 0.001 and 0.01 is suggested.

- **Modify the max number of successful steps**

  This item allows you to set the maximum number of successful (i.e., converged) steps. The computation will be stopped when this number is reached even if the upper time limit has not yet been reached. The default value is 200. Note that you can always restart the scheme from the results file in order to continue the time-marching scheme without losing the benefit of the previous solutions. See *Starting an ANSYS POLYFLOW Calculation from an Existing Results File* (p. 109) for more information on these file types.

6. In the same menu, you can also specify the time-marching scheme using one of the following items:

   - **Use of the implicit Euler method** (the default scheme)
   - **Use of the Galerkin method**
   - **Use of the Crank-Nicolson method**

   See *Time-Marching Schemes* (p. 542) for details of these methods.

7. In the same menu, you can also enable or disable the prediction of the velocity field:

   - **Enable/Disable prediction of velocity field**

     In many processes, inertia terms are small compared to viscous forces; often they are even completely ignored. If this is the case, control of the accuracy for the time integration of the momentum equation will require very small time steps, which is usually not justified. It is therefore recommended to disable the accuracy check for the time integration of the momentum equation by selecting the **Enable/Disable prediction of velocity field** option. This is especially recommended for blow molding applications involving contact detection with extremely large variations of the velocity field at extremely short time scales. See *Time Dependence and Contact Handling* (p. 389) for details.

27.3.2. Initial Conditions

The default initial condition for time-dependent problems has all variables set to zero. In many situations, you will want to calculate a steady-state solution first, and then use it as an initial condition for the time-dependent problem. This steady-state solution will be saved in a results file (called `res`, by default). See *Starting an ANSYS POLYFLOW Calculation from an Existing Results File* (p. 109) for information about starting from an existing results file.

If you start from a time-dependent results file, ANSYS POLYFLOW can also read the time derivative of the solution from the corresponding restart file (which is automatically saved during a time-dependent calculation and called `rst` by default). See *Starting an Evolution or Time-Dependent Calculation from Existing Results and Restart Files* (p. 109) for details.

If you start without the `rst` file, ANSYS POLYFLOW will not be able to perform a first-order or second-order continuation, and will not find the old value of t.

If you want to continue the calculation starting from the `res` file, you have to first modify the initial values of t and dt in the **Numerical Parameters** menu. Specifying the values that correspond to the `res` file is almost equivalent to using an `res` file and an `rst` file. The only difference is that the t-derivative of the solution is not available and the convergence of the continuation may be different.

If an `rst` file is available, the extrapolation scheme can be restarted exactly at the level where the program was interrupted before. Typically, when the second-order Crank-Nicolson/Adams-Bashforth scheme is used, the time derivatives at two previous time steps are used to extrapolate the solution. When the first-order explicit Euler scheme is used, the derivative at the previous time step is used for the extrapolation.

You should never modify program data such as model parameters or boundary conditions when starting from an `rst` file. Any deviation will result in an ill-posed problem.

27.3.3. Output of Results for Time-Dependent Problems

For time-dependent calculations, you can ask ANSYS POLYFLOW to save the output files at specified intervals during the calculation. This will allow you to examine the progress of the solution, and study the time history of solution variables. See *Output for Time-Dependent and Evolution Calculations* (p. 126) for more information.

27.4. Outputs to a Listing File

While performing the time-marching procedure, ANSYS POLYFLOW writes the time step information to a listing file (if one has been specified) or your screen, whenever a new time step has successfully been calculated. The information displayed is

- **t**: The estimated time of the solution.
- **dtnew**: The time step used in the next step.
- **step #**: The number of the recently completed step.
- **okincr**: If the solution is acceptable, this will be T; otherwise, it will be F. If the solution is not acceptable, then the time step will be rejected.
- **tests**: The relative maximum difference between the predicted and corrected values. This gives an estimate of local truncation errors that are less than or equal to 0.1 times the prescribed relative tolerance.

You can use this information to determine whether the time step is being limited by accuracy or by the convergence of the iterations. If convergence occurs but the time step remains small, the small time steps are caused by a strict tolerance criteria. If this is the case, you may want to disable the velocity field prediction, as discussed at the end of *Setting Up a Time-Dependent Problem* (p. 544).

27.5. Interrupting Time-Dependent Computations

As for evolution schemes, a transient scheme may be interrupted when a given variable reaches a limit value. The description of how to interrupt evolution in *Interrupting Evolution* (p. 527) is also applicable to time-dependent computations. See this section for further details.

27.6. Volume Conservation

Some transient flow calculations are performed for a given material volume, for which the conservation must be guaranteed. Often, the incompressibility equation suffices for this, however, that equation has only a local effect, and does not prevent the numerical loss of creation of material that may originate, e.g., from round-off errors or from adaptive meshing. Hence, you can impose a global constraint on the entire fluid domain. For this, at the level of the definition of a sub-task, you should select the item

Volume conservation

When entering the menu, ANSYS POLYDATA informs you about the current known volume of the discretized fluid domain. The discretized attribute is important, and the given value may differ from the expected value. For example, a discretized sphere may have a volume that is slightly lower than the volume of the actual sphere. The current status is also displayed. By default, this volume conservation is disabled. You can activate it by clicking on the option

Enable volume conservation

Subsequently, you can modify the prescribed value of the volume by selecting the option

Modify prescribed volume =

and by introducing a new value. This last option should be used for compensating the possible departure from the expected volume. For obvious reasons, it should not be used for assigning a totally different value to the volume that may eventually be incompatible with the other attributes of the simulation.

Important

The volume conservation can be invoked for cases where the volume is known to be constant. In other words, it should not be invoked for cases where boundary conditions involve entry/exit of material, neither for cases where volume is created via a changing density.

Chapter 28: Computing Derived Quantities

This chapter describes the derived quantities that can be calculated during your ANSYS POLYFLOW simulation and displayed when you postprocess your results.

The following topics are discussed

28.1. Overview of Derived Quantities

Derived quantities in ANSYS POLYFLOW are also known as postprocessors. Postprocessors are sub-tasks that require the results of a previous calculation as their input. For example, the calculation of viscosity based on the known velocity and temperature fields can be performed in a postprocessor.

Important

Do not confuse ANSYS POLYFLOW postprocessors, which compute certain quantities, with graphics programs like FLUENT/Post that are used to visualize results.

Postprocessors can be defined in ANSYS POLYDATA when at least one other sub-task has already been defined. You must specify the type of postprocessor you want, as well as any related material properties and boundary conditions. The derived quantities that are available in ANSYS POLYDATA, and the inputs required to define them, are described in the remaining sections of this chapter.

28.2. Stream Function

For 2D planar or axisymmetric flow problems, ANSYS POLYFLOW will automatically calculate a stream function based on the velocity field.

28.2.1. Calculation of Stream Function

For incompressible fluids, the stream function Ψ is given by

$$u = \frac{\delta\Psi}{\delta y}, v = -\frac{\delta\Psi}{\delta x} \tag{28–1}$$

These equations are slightly different for the 2D axisymmetric case. *Equation 28–1* (p. 552) does not require any material parameters to be specified, and boundary conditions are always of the Neumann type.

On parts of boundaries where a zero normal velocity has been imposed, ANSYS POLYFLOW requires the stream function to be constant, to ensure that an isoline of Ψ will not cross the domain boundary due to approximation errors.

Equation 28–1 (p. 552) guarantees the uniqueness of the stream function only if the value of Ψ has been imposed at a minimum of one node. By default, ANSYS POLYDATA sets Ψ equal to zero at the node nearest to (x, y) = (0, 0). Isolines of Ψ are always parallel to the velocity vector and correspond to particle trajectories only in steady-state flows.

28.2.2. User Inputs for Stream Function

For 2D planar or axisymmetric flow problems, ANSYS POLYFLOW will automatically calculate a stream function based on the velocity field. To change the point where the stream function vanishes, follow the steps below:

1. In the task menu, select **Assign the stream function**.

   Assign the stream function

2. Specify the condition of the stream function.

   Condition on the stream function for field 1

   a. Click **No** when ANSYS POLYDATA asks you if you want the stream function to vanish at the node closest to the coordinates (0, 0).

   b. Enter a **New value** of **X** and click **OK**.

   c. Enter a **New value** of **Y** and click **OK**.

28.3. Local Shear Rate

ANSYS POLYFLOW can calculate the local shear rate ($\dot{\gamma}$) on the part of the domain where the velocity field has been calculated. The shear rate is given by

$$\dot{\gamma} = \sqrt{2\ \mathrm{tr}\ (\mathbf{D}^2)} \tag{28–2}$$

The procedure for specifying the calculation of the local shear rate is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Local shear-rate**.

   Local shear-rate

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

   No additional parameters or boundary conditions are required.

28.4. Viscosity

For generalized Newtonian fluids, ANSYS POLYFLOW can calculate the local value of the viscosity, which can depend on the shear rate and/or the temperature. The viscosity will appear as a scalar field in your graphical postprocessing program.

The procedure for specifying the calculation of the local viscosity is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Viscosity**.

Viscosity

ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

No additional parameters or boundary conditions are required.

28.5. Rate-of-Deformation Tensor

ANSYS POLYFLOW can calculate the rate-of-deformation tensor $\mathbf{D}$ on the part of the domain where the velocity field has been computed. Three components of the rate-of-deformation tensor will be calculated for 2D planar cases, four components for 2D axisymmetric cases, and six components for all other cases.

This postprocessor will produce a tensor with three, four, or six components, depending on the problem geometry. Furthermore, the components are based on the selected reference frame (Cartesian for 2D planar and 3D cases, cylindrical for axisymmetric cases). In many cases, the calculation of the local shear rate (see *Local Shear Rate* (p. 553)) is preferred if only a (generalized) scalar kinematics quantity is needed.

The procedure for specifying the calculation of the rate-of-deformation tensor is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Rate of deformation tensor**.

   Rate of deformation tensor

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

   No additional parameters or boundary conditions are required.

28.6. Inelastic Stress Tensor

For generalized Newtonian flow, ANSYS POLYFLOW can calculate the extra-stress tensor T on the part of the domain where the velocity field has been calculated. Three components of the inelastic stress tensor will be calculated for 2D planar cases, four components for 2D axisymmetric cases, and six components for all other cases.

The procedure for specifying the calculation of the inelastic stress tensor is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

Postprocessor

b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Inelastic stress tensor**.

   Inelastic stress tensor

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

3. Specify any additional outputs that you want for the inelastic stress tensor. ANSYS POLYFLOW calculates all of the tensor components by default. To specify the calculation of the tensor eigenvalues and/or the tensor component along velocity, select the appropriate **Enable** menu item(s).

   To disable any of the currently-enabled components, select the corresponding **Disable** menu item(s). See *Stress Eigenvalues and Components* (p. 568) for details about the evaluation of the eigenvalues and the extra-stress component along the direction of the velocity.

   No additional parameters or boundary conditions are required, since the viscosity law is the same as that for the flow problem (which must be of the generalized Newtonian type).

28.7. Viscous Heating and Dissipated Power

For generalized Newtonian flows, ANSYS POLYFLOW can calculate the heat generation per unit time and per unit volume (or per unit area for a 2D planar case) that is generated under the given flow conditions. In addition to the viscous heating, ANSYS POLYFLOW evaluates the dissipated power, which is the integral of the viscous heating over the flow domain.

It is not necessary to solve a nonisothermal problem in order to compute the viscous heating. ANSYS POLYFLOW can calculate the viscous dissipation that occurs without actually solving the energy equation. Also, ANSYS POLYFLOW can calculate the viscous heating even if it has been neglected in the main nonisothermal flow problem.

The procedure for specifying the calculation of viscous heating is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Viscous heating**.

   Viscous heating

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

No additional parameters or boundary conditions are required, since the viscosity law is the same as that for the flow problem (which must be of the generalized Newtonian type).

28.8. Total Extra-Stress Tensor

For differential or integral viscoelastic flow problems, ANSYS POLYFLOW can calculate the total extra-stress tensor in the domain. The total extra-stress tensor is the sum of the viscous (or the (generalized) Newtonian) components and the viscoelastic components.

If more than one relaxation time has been defined, the viscoelastic component is the sum of the viscoelastic components for all modes.

The procedure for specifying the calculation of the total extra-stress tensor is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Total extra-stress tensor**.

   Total extra-stress tensor

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

3. Specify any additional outputs for the total extra-stress tensor. ANSYS POLYFLOW calculates all of the tensor components by default.

   - To specify the calculation of the tensor eigen values and/or the tensor component along velocity, select the appropriate **Enable** menu item(s).
   - To disable any of the currently-enabled components, select the corresponding **Disable** menu item(s).

   No additional parameters or boundary conditions are required, since the postprocessor takes its data from the viscoelastic flow problem itself.

   See *Stress Eigenvalues and Components* (p. 568). for details about the evaluation of the eigenvalues and the extra-stress component along the direction of the velocity.

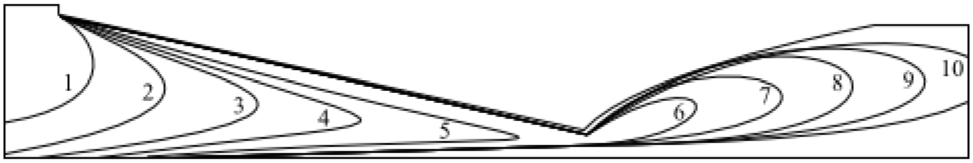
28.9. Residence Time

ANSYS POLYFLOW can calculate the residence time distribution in the part of the domain where a velocity field has been calculated. The residence time distribution can be calculated in 2D and 3D calculations in steady-state and time-dependent modes. This postprocessor differs from individual time integration along a trajectory, where the result is obtained for a single material point only. With this postprocessor, a residence time distribution value is obtained over the whole domain.

28.9.1. Calculation of Residence Time

Consider a simple steady-state free surface flow, where the flow enters from the upper left part of the domain. Assuming that the residence time distribution is zero in the entry section, ANSYS POLYFLOW will calculate the residence time distribution in the domain. An example of contour lines of the residence time distribution is shown in *Figure 28.1* (p. 557).

Figure 28.1 Contour: Residence Time Distribution for a Steady-State Free Surface Problem



ANSYS POLYFLOW computes the residence time distribution using the equation

$$\frac{DT}{Dt} = 1 \quad \textbf{(28–3)}$$

where $T = 0$ at time $t = 0$, and the boundary conditions along the entry sections are taken into account. T is the residence time, and $\frac{DT}{Dt}$ is the material derivative of the residence time.

Equation 28–3 (p. 557) states that the residence time increases as the absolute time t for particles along their path increases. It is a hyperbolic equation, and it requires boundary conditions (as well as initial conditions for time-dependent problems). The value of the residence time must be set along all entry sections.

Usually you will set a value of zero. On all sections on which an inflow boundary condition has been applied for the momentum equation, the residence time is set to zero by default. You can add boundary conditions on entry sections where a different type of condition has been selected.

For time-dependent calculations, ANSYS POLYFLOW imposes $T = 0$ as an initial condition. In this case, the time-dependent problem is well-posed provided that T has been imposed on all entry sections. For steady-state problems, *Equation 28–3* (p. 557) does not define a well-posed problem in recirculating zones (vortices) and along walls where $\mathbf{v} = 0$.

This is not usually important, since the residence time distribution is meaningless in these regions, and you can easily exclude these corresponding contour lines from the display in your graphical postprocessing program. In *Figure 28.1* (p. 557), contour lines have been selected such as to eliminate large numbers of contours along walls.

Another possibility for calculating the residence time distribution in the presence of steady-state vortices is to define a time-dependent problem that uses a steady-state velocity field that has already been calculated. In this cases, the velocity field will be obtained from the `res` file.

The residence time distribution postprocessor uses the 4×4 subelement technique in 2D and the 2×2×2 subelement technique in 3D, together with a consistent streamline-upwinding/Petrov-Galerkin formulation.

28.9.2. User Inputs for the Residence Time

The procedure for specifying the calculation of the residence time is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Residence time**.

   Residence time

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

3. Specify the residence time along all entry sections. By default, sections that have an inflow boundary condition have a zero value for residence time.

   > **Important**
   >
   > You must specify at least one residence time boundary condition.

   a. To specify the residence time on an entry section, select the appropriate **No condition** menu item and click **Modify**.

   b. Select the **Value imposed** option.

   c. Enter values for the constants a, b, c, and d to satisfy the equation

   $$\text{residence time} = a + bx + cy + dz \qquad \textbf{(28–4)}$$

      i. To modify the value of a constant, select the appropriate **Modify** menu item.

      ii. Enter a **New value** for the relevant constant and click **OK**.

   d. Repeat to specify the residence time on other boundaries (if required).

28.10. Tracking of Material Points

In time-dependent calculations, it is sometimes necessary to calculate the trajectory of material points. ANSYS POLYFLOW is not a Lagrangian code (i.e., it does not use a description and formulation following the particles), and nodes do not correspond, in general, to material particles. Therefore, a postprocessor is provided to track these material points. Typically this is useful in blow molding simulations, when remeshing is not of the Lagrangian type, although it can be very useful in many other situations also.

28.10.1. Calculation for Tracking Material Points

A material particle's initial position is tracked according to

$$\frac{D\mathbf{X}}{Dt} = 0 \tag{28–5}$$

where $\mathbf{X}$ is the initial position of the material particle (i.e., the original position before deformation occurs). When the material points postprocessor is used, ANSYS POLYFLOW will keep track of the initial position of each material point, as well as the final position of that same material point. This allows you, for example, to determine the source on the parison of a flaw that occurs at a particular location on the final product.

Figure 28.2 (p. 560) shows the results for a time-dependent blow molding problem. No entry sections have been defined, so no additional boundary conditions are necessary. In FLUENT/Post , you can select contour lines of x or y component. Since the y component roughly corresponds to the tangential direction, *Figure 28.2* (p. 560) shows contour lines of initial y positions. These lines correspond to material lines.

Since *Equation 28–5* (p. 559) is a hyperbolic equation, the material points postprocessor uses the 4 × 4 subelement technique in 2D and the 2 × 2 × 2 subelement technique in 3D. Subrefinement of the variable governed by a hyperbolic equation with respect to the interpolation of the velocity field is a powerful and accurate technique for dealing with purely advective problems.

28.10.2. User Inputs for Tracking Material Points

The procedure for specifying the tracking of material points is as follows:

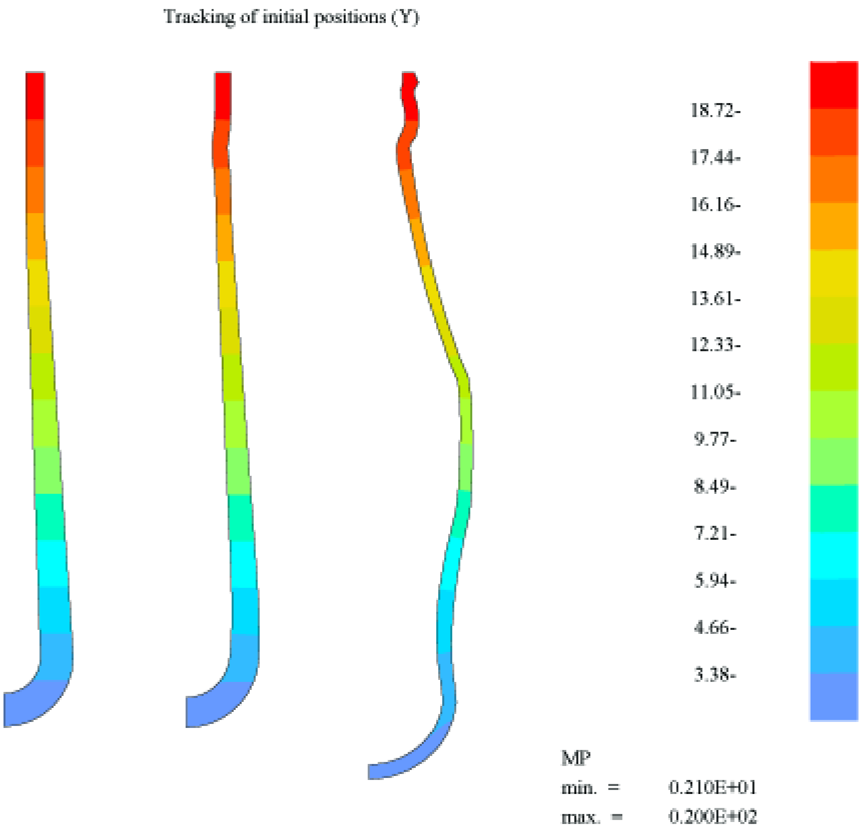
1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

Figure 28.2 Contour Lines of the Initial y Position



b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Tracking of material points**.

Tracking of material points

ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

28.11. Tracking of a Material Property

For flow problems, you can track particles by assigning a value of a material property (a conventional scalar field named c for "color") in the flow domain and integrating the pure advection of this property in the flow domain:

$$\frac{Dc}{Dt} = 0 \qquad \textbf{(28–6)}$$

Equation 28–6 (p. 560) is subject to inlet boundary conditions that can be defined separately on each inlet section. This postprocessor is typically used to calculate the transport of the property in the flow. Each separate entry has a different color represented by a separate value of the c property along the inlet. The flow transports the property and you can locate the color at each position in the flow domain.

The continuous interpolation used in the numerical scheme will produce some interpolation error, so intermediate values will also be displayed in the flow domain. However, provided you select distinct values of the property along inlets (such as 1 and 10, for example), you can easily identify regions of the flow in your graphical postprocessing program.

The procedure for specifying the calculation of a material property is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Tracking of a material property**.

   Tracking of a material property

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** in the message panel to continue.

3. Specify the material property along all entry sections.

   Important

   You must specify at least one material property boundary condition.

   a. To specify the material property on a boundary, select the appropriate **No condition** menu item and click **Modify**.

   b. Select the **Value imposed** option.

   c. Enter values for the constants a, b, c, and d to satisfy the equation

$$\text{material property} = a + bx + cy + dz \tag{28–7}$$

      i. To modify the value of a constant, select the appropriate **Modify** menu item.

      ii. Enter a **New value** for the relevant constant and click **OK**.

   d. Repeat to specify the material properties on other boundaries (if required).

28.12. Forces on Slices

For fiber spinning problems, ANSYS POLYFLOW can compute the force acting on each slice of the mesh. The procedure for specifying the calculation of forces on slices is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

Postprocessor

b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Forces on slices**.

   Forces on slices

3. Specify the domain where you want the calculation to be performed.

   a. Select any domains where you do not want to perform the calculation and click **Remove**. ANSYS POLYDATA will move these subdomains to the list at the bottom of the panel. The domains where you want the calculation to be performed will remain in the list at the top of the panel.

   b. Click **Upper level menu**.

4. Define the remeshing inlet and outlet.

   a. Define the remeshing inlet. Select the inlet of the slicing system and click **Confirm**.

   b. Define the remeshing outlet. Select the outlet of the slicing system and click **Confirm**.

28.13. Heat Fluxes

ANSYS POLYFLOW can integrate in the flow domain the so-called Fourier heat flux in regions where a temperature field is defined. The Fourier heat flux $\mathbf{q}$ is defined as

$$\mathbf{q} = -\left(k\left(T \right) \nabla T \right) \tag{28–8}$$

where $k\left(T \right)$ is the heat conductivity.

The heat flux will appear as a vector in your graphical postprocessing program. The procedure for specifying the calculation of heat fluxes is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Heat Fluxes**.

   Heat Fluxes

   ANSYS POLYDATA will tell you the subdomains where ANSYS POLYFLOW can perform the calculation. Select the subdomains where you want ANSYS POLYFLOW to calculate the heat fluxes. No additional parameters or boundary conditions are required.

28.14. Flow Rate

ANSYS POLYFLOW automatically integrates the total volumetric flow rate along all boundary sets, so you do not need to define a postprocessor sub-task for **Flow Rate**. Results are displayed as numerical values in the ANSYS POLYFLOW listing file (search for the word `Flow rates` to find them), or as a mass balance in ANSYS POLYDIAG (**Mass balance** in the **Evolution information** panel).

This mass balance is a measure of the quality of the solution, and is defined as the ratio of the total flow rate along all boundary sets to the maximum inlet flow rate.

28.15. Joule Effect

For problems involving the calculation of an electrical field (see *Electrical Heating* (p. 485)), ANSYS POLYFLOW can integrate in the domain the Joule effect defined as

$$q_t = -\frac{\sigma_\varepsilon \nabla V \cdot \nabla V}{2} \tag{28–9}$$

where q_t stands for the heat source per unit volume in the energy equation, σ_ε is the electrical conductivity of the material, and V is the electrical potential field.

The Joule effect will appear as a scalar in your graphical postprocessing program. The procedure for specifying the calculation of the Joule effect is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Joule Effect**.

   Joule Effect

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

   No additional parameters or boundary conditions are required.

28.16. Mixing Index

The mixing index λ is defined as [*33*] (p. 716)

$$\lambda = \frac{\dot{\gamma}}{\dot{\gamma} + \omega} \qquad \textbf{(28–10)}$$

where $\dot{\gamma}$ is the equivalent shear rate defined as

$$\dot{\gamma} = \sqrt{2 \ \mathrm{tr} \left(\mathbf{D}^2 \right)} \qquad \textbf{(28–11)}$$

and ω is the magnitude of the vorticity vector. For a shear flow, the mixing index is equal to 0.5, whereas its value is 1 in a purely elongational flow.

In a purely rotational flow, the mixing index is zero. The mixing index is not defined everywhere in the flow problem; singular values arise in regions where the denominator is zero (e.g., along the axis of symmetry of a Poiseuille flow).

The mixing index will appear as a scalar in your graphical postprocessing program. The procedure for specifying the calculation of the mixing index is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Mixing Index**.

   Mixing Index

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

   No additional parameters or boundary conditions are required.

28.17. Vorticity

ANSYS POLYFLOW can calculate the vorticity vector for the flow. Vorticity is a measure of the rotation of a fluid element as it moves in the flow field, and is defined as the curl of the velocity vector:

$$\xi = \nabla \times \mathbf{v} \qquad \textbf{(28–12)}$$

For 2D flows, the magnitude of the vorticity is displayed, while it will appear as a vector in your graphical postprocessing program for 3D flows. The procedure for specifying the calculation of vorticity is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Vorticity**.

   Vorticity

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue. No additional parameters or boundary conditions are required.

28.18. Convected Heat

The convected heat postprocessor computes the integral along selected boundaries of $\rho c_p \mathbf{v} \cdot \mathbf{n} T$, where T is the temperature. The computed value for each boundary will appear in the listing file.

The procedure for specifying the calculation of convected heat is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Convected Heat**.

   Convected Heat

   ANSYS POLYDATA will tell you the subdomain(s) on whose boundaries the calculation can be performed. You can click **OK** to continue.

3. For each boundary for which you want to compute the convected heat, follow these steps:

   a. Select the boundary name in the list of boundaries.

   b. Click **Modify**.

   c. Select **Convected heat calculated**.

   If you decide to disable the convected heat calculation, follow the same steps, but select **No convected heat calculated**.

28.19. Parison Thickness

For 2D axisymmetric extrusion simulations, you can compute the parison thickness. The parison being extruded has at least two surfaces. At each point in the domain, ANSYS POLYFLOW computes the distance between the point and both the inner and the outer surfaces. A symmetric calculation is performed, and at a given location or height along the surface, a unique quantity is given. This method can be used to compute the thickness of a parison being extruded, or the thickness of a blown product.

Important

The calculation method is valid for relatively thin objects, with little variation in thickness; it is not appropriate for thick objects.

The procedure for setting up the calculation of the parison thickness is as follows:

1. Create a new sub-task.

   Create a sub-task

   - Select **Postprocessor** as the sub-task type. When prompted, enter a name.

     Postprocessor

2. In the **F.E.M. Task Postprocessor** menu, select **Parison thickness**.

   Parison thickness

   By default, **parison #01** is already created. To create a new one, select **Creation of a new parison**. To delete a parison, select **Deletion of a parison**.

3. Select the parison for which you want to compute the thickness (e.g., **parison #01**).

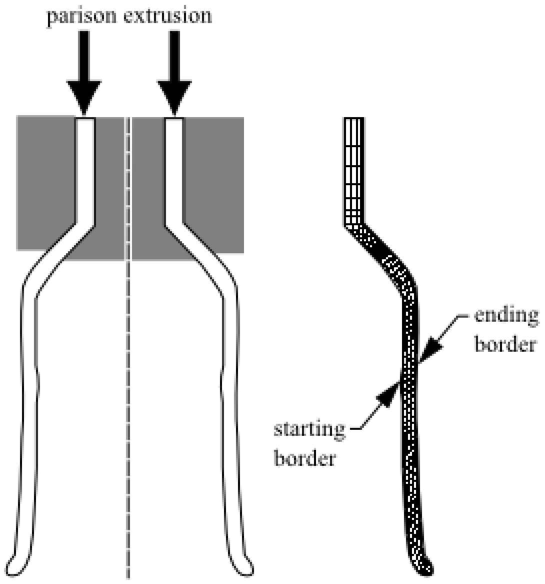
   parison #01

4. Specify the region where the postprocessor sub-task applies.

   Domain of the sub-task

5. Specify the boundary sets representing the starting and ending borders to be used in the thickness calculation (see *Figure 28.3* (p. 567)).

Figure 28.3 Starting and Ending Borders for Parison Thickness Calculation



ANSYS POLYFLOW will evaluate the distance between these borders at a point within them to determine the thickness at that location.

Borders for thickness calculation

a. In the resulting **Borders for thickness calculation** panel, select the boundary set representing the first border (or a portion of it) and click **Modify**.

b. Choose **Starting border**.

c. Repeat the previous two steps if other boundary sets are needed to completely define the starting border.

d. Select the boundary set representing the second border (or a portion of it) and click **Modify**.

e. Choose **Ending border**.

f. Repeat the previous two steps if other boundary sets are needed to completely define the ending border.

28.20. Extension Evaluation

Evaluation of the extension components is available for 3D blow molding and thermoforming simulations. See *Calculation of the Extensions* (p. 385) for information on the calculation of the extension components, and *Inputs for Computing the Extension Components* (p. 403) for details on defining a postprocessor for extension evaluation.

28.21. Mass of Blown Product

Calculation of the mass of the blown product is available for 3D blow molding and thermoforming simulations. See *Calculation of the Mass of the Blown Product* (p. 388) for information on the calculation of the mass of the blown product, and *Inputs for Computing the Mass of the Blown Product* (p. 404) for details on defining a postprocessor for the mass of the blown product.

28.22. Volume of Blown Product

Calculation of the volume of the blown product is available for 3D blow molding and thermoforming simulations. See *Calculation of the Volume of the Blown Product* (p. 388) for information on the calculation of the volume of the blown product, and *Inputs for Computing the Volume of the Blown Product* (p. 405) for details on defining a postprocessor for the volume of the blown product.

28.23. Permeability of Blown Product

Calculation of the permeability of the blown product is available for 3D blow molding and thermoforming simulations. See *Calculation of the Permeability of the Blown Product* (p. 388) for information on the calculation of the permeability of the blown product, and *Inputs for Computing the Permeability of the Blown Product* (p. 407) for details on defining a postprocessor for the permeability of the blown product.

28.24. Stress Eigenvalues and Components

The only proper way to represent a stress field is by means of tensors. Although it is easy to represent a velocity by means of a vector, the situation is somewhat more complex for a tensor. The most appropriate way to represent a tensor is by displaying ellipsoids at each point. In addition, the amount of information contained in a tensor is far greater than in a vector. For a 3D calculation, a stress tensor would involve six independent components.

28.24.1. Calculation of Eigenvalues

From a stress tensor, it is possible to determine the stress that is actually applied on a small surface element of matter. Consider a fluid domain and a small surface with normal unit direction given by $\hat{\mathbf{n}}$. The force density $\mathbf{f}$ applied on this surface can be obtained from

$$\mathbf{f} = \mathbf{T} \cdot \hat{\mathbf{n}} \tag{28–13}$$

where $\mathbf{T}$ is the stress tensor. In general, $\mathbf{f}$ will not be oriented along $\hat{\mathbf{n}}$, but it is possible to select a surface with an orientation such that $\mathbf{f}$ and $\hat{\mathbf{n}}$ have the same direction. In this case,

$$\mathbf{f} = \mathbf{T} \cdot \hat{\mathbf{n}} = \sigma \hat{\mathbf{n}} \tag{28–14}$$

where σ is the proportionality factor. This equation can be rewritten as

$$(\mathbf{T} - \sigma \mathbf{I}) \cdot \hat{\mathbf{n}} = 0 \tag{28–15}$$

where $\mathbf{I}$ is the unit tensor. This is the characteristic equation for $\mathbf{T}$, and the solutions for σ are said to be the eigenvalues.

ANSYS POLYFLOW solves *Equation 28–15* (p. 568) by setting the determinant of the first term on the left-hand side to zero:

$$\det(\mathbf{T} - \sigma \mathbf{I}) = 0 \tag{28–16}$$

In general, this leads to a third-order polynomial expression, the roots of which are the eigenvalues. There are three eigenvalues, but for most 2D planar flows, the third one is equal to zero. These quantities are independent with respect to the selected reference frame, and may provide a more convenient description of the stress state in the fluid than the display of individual components of the stress tensor.

28.24.2. Calculation of the Stress Component Along the Velocity Direction

It is also useful to evaluate the stress component along the velocity direction. This is accomplished by evaluating the force density, which is applied on a small surface that is perpendicular to the velocity. If $\hat{\mathbf{v}}$ is the unit direction along the velocity, the stress component σ_v along the velocity direction can be evaluated formally as

$$\sigma_v = \hat{\mathbf{v}} \cdot \mathbf{T} \cdot \hat{\mathbf{v}} \qquad \textbf{(28–17)}$$

ANSYS POLYFLOW solves a more simple (but equivalent) form of this equation:

$$\sigma_v = \frac{\mathbf{v} \cdot \mathbf{T} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} \qquad \textbf{(28–18)}$$

where $\mathbf{v}$ is the velocity vector. Dividing by the square of the velocity magnitude is necessary, since the quantity σ_v depends only on the direction of the velocity field, and not explicitly upon its magnitude.

Clearly this quantity σ_v can be evaluated if the velocity field is nonzero. It is important to note that this quantity is not objective; indeed, a superimposed rigid motion can affect it. However, if the velocity boundary conditions are carefully selected, for a steady-state flow, this quantity provides extra-stress information along path lines.

If the total stress component $\sigma_{tot,v}$ along the velocity direction is needed, you can add the pressure contribution as follows:

$$\sigma_{tot,v} = -p + \sigma_v \qquad \textbf{(28–19)}$$

You can obtain this value using a custom field function in CFD-Post. See the CFD-Post documentation for details.

28.24.3. User Inputs for Stress Eigenvalues

You will specify the inputs for stress eigenvalues when you set up the calculation of the inelastic stress tensor or the total extra-stress tensor, as described in *Inelastic Stress Tensor* (p. 554) and *Total Extra-Stress Tensor* (p. 556), respectively.

28.25. Quantification of Die Balancing

Die balancing is a major challenge for the die designer. To help address this challenge, ANSYS POLYFLOW can compute a measure of the die balancing based on the deviation of the normal velocity with respect to the average velocity along the die exit:

$$b_d = \int_{\partial \Omega} \left(\mathbf{v} \cdot \mathbf{n} - \frac{Q}{S} \right)^2 ds \qquad \textbf{(28–20)}$$

where $\mathbf{v}$ is the velocity, Q is the flow rate across the boundary set, S is the surface of the boundary set, and $\mathbf{n}$ is the outward normal along the boundary set.

A perfectly balanced die with a uniform velocity at the exit leads to a measure of the die balancing equal to zero. This measure is not absolute; you have to compare the die balancing measure for different die geometries to determine which is the best one (i.e., the one with the smallest value for the die balancing measure). Results are displayed as numerical values in the ANSYS POLYFLOW listing file (search for the words `Flow balance` to find them), or in ANSYS POLYDIAG (in the **Mass balance** panel).

The procedure for specifying the calculation of the die balancing measure is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Flow Rate and Balance**.

   Flow Rate and Balance

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

3. Specify the boundary sets for which you want to compute the die balancing measure. By default, ANSYS POLYFLOW will compute only the flow rate (described in *Flow Rate* (p. 563)) on all boundary sets.

   a. Select a boundary set where you want to perform the quantification of die balancing and click **Modify**.

   b. Select **Flow rate and balance calculated**.

   c. Repeat for any other boundary sets (if required) and click **Upper level menu**.

28.26. Energy Balance

For a nonisothermal simulation, ANSYS POLYFLOW automatically integrates all quantities related to energy:

- Total viscous heating in the flow domain
- Total heat source
- Heat flux (Fourier heat flux) when the temperature is imposed
- Total Joule effect
- Convected heat along a boundary

along all boundary sets, so you do not need to define postprocessor sub-tasks for them. Results are displayed as numerical values in the ANSYS POLYFLOW listing file or in ANSYS POLYDIAG (in the **Energy balance** panel).

28.27. Parison Programming

Optimization of the thickness distribution of a blown product is available for 3D blow molding and thermoforming simulations in shell models. See *Inputs for Parison Programming* (p. 408) for details on how to define such a postprocessor.

28.28. Volume of Liquid

With this postprocessor, you can evaluate the volume of the flow domain. If the flow domain contains overlapping internal moving parts (see *Flows with Internal Moving Parts* (p. 457)), the calculation of the volume will remove the part of the flow domain overlapped by the moving parts, in order to provide the actual volume occupied by the fluid.

The procedure for specifying the calculation of the flow domain volume is as follows:

1. Create a new sub-task.

   Create a sub-task

   a. Select **Postprocessor** as the sub-task type.

      Postprocessor

   b. When prompted, enter a name for the postprocessor sub-task.

2. In the **F.E.M. Task Postprocessor** menu, select **Volume of liquid**.

   Volume of liquid

   ANSYS POLYDATA will tell you where the calculation will be performed. You can click **OK** to continue.

   No additional parameters or boundary conditions are required.

28.29. Temperature Programming

Optimization of the initial temperature distribution of a thermoformed product is available for shell models. See *Inputs for Temperature Programming* (p. 412) for details on how to define such a postprocessor.

28.30. Thickness Evaluation

In 2D as well as in 3D simulations of blow molding and pressing, you can compute the thickness of the final object (as well as during shaping). In industrial processing, the thickness has to be understood as the distance between two arbitrary surfaces; it is therefore subjected to an appropriate definition. Also, from the point of view of modeling, it would require the identification of preferably distinct topological entities; however this cannot always be achieved. A technique has been implemented, which matches a reasonable industrial definition as much as possible: at a given point, it basically consists of evaluating the distance with respect to the closest opposite surface. A topological object has to be constructed for the thickness evaluation, and a few parameters for geometric tolerances have to be given. In 3D, the thickness is evaluated on the selected boundary; planes of symmetry can be discarded from the domain of evaluation. Note that in 2D, a similar scenario applies; however, since the display of a quantity along a line is not easy, the evaluation is expanded onto the domain via a Laplace equation.

Typically, the thickness at the end of the process is of interest. Hence, it is not necessary to perform a thickness evaluation at each time step. Also, since the thickness at a point is evaluated as a distance measured along a line perpendicular to the surface at this point, it is preferable to bound its value.

The procedure for setting up the evaluation of thickness is as follows.

1. Specify the domain of the evaluation.

   Domain of evaluation

   Specify the domain where the thickness needs to be evaluated; it coincides with the fluid domain in blow molding and pressing simulations, and with the solid domain when evaluating residual stresses and deformations.

2. Specify the boundary of the evaluation.

   Boundary of the evaluation

   Specify the border which is used as the basis for the evaluation of the thickness; it may consist of one or several entities, but typically symmetry planes and lines are not considered.

3. Update the current coordinates with displacements.

   Take displacement into account

   When the simulation involves the prediction of residual stresses and deformations (along with the Narayanaswamy model), it is possible to specify that the thickness has to be evaluated on the basis of coordinates updated with residual displacements. Note that the actual coordinates update is only performed internally.

4. Specify the activation time.

   Modify the activation time

   Typically, the thickness at the end of the process is of interest. Hence, it is not necessary to perform a thickness evaluation at each time step, and you may select the activation time for thickness evaluation. The thickness evaluation will be performed only when the simulation time is greater than or equal to the activation time.

5. Specify the thickness limit.

   Modify the thickness limit

   The thickness at a point is evaluated as a distance between the point and the opposite surface, measured along a line perpendicular to the surface at this point. It is preferable to bound its value in order to prevent geometrically irrelevant values.

6. Specify the geometric tolerance.

   Modify the geometric tolerance

The thickness at a point is evaluated as a distance between the point and the opposite surface, measured along a line perpendicular to the surface at this point. The algorithm involves a search of geometric intersection (between the perpendicular line and boundary elements). A geometric tolerance is needed in order to compensate for round-off errors.

Chapter 29: Using the Solver

This chapter describes some details of the solver implementation and solver parameters that are under your control, as well as solver parameters that ANSYS POLYDATA and ANSYS POLYFLOW set for you automatically.

29.1. Controlling the Calculations

29.1.1. Recalculating with a Different Interpolation

Iterating can begin from the default solution (with all values set to zero) or from an old results file. ANSYS POLYFLOW can interpolate from a saved results file to a modified results file obtained with a different element type. For example, you can start with a temperature field obtained with quadratic elements and compute a new solution with 4×4 subelements. To do this, you will specify in ANSYS POLYDATA the solution from which to start (as described in *Starting an ANSYS POLYFLOW Calculation from an Existing Results File* (p. 109)), change the interpolation scheme, save a new data file, and begin the ANSYS POLYFLOW calculation from the new data file.

It is also possible to start a calculation from results obtained on a different mesh. See *Results Interpolation Onto Another Mesh* (p. 150) for details.

29.1.2. Specifying the Number of Iterations

You can select the maximum number of iterations you want to perform. The default number is 10, which is sufficient when the Newton-Raphson method is used. If you are using a Picard scheme or a decoupled scheme, a higher value may be needed, depending on the precision required. For the Picard scheme, the maximum number of iterations is automatically increased from 10 to 30. ANSYS POLYFLOW's expert system that helps for convergence may in some cases increase the number of iterations beyond the specified maximum number on the basis of past convergence history. Typically this will be done if the system of equations is "almost converging," but a few more iterations are needed.

For linear problems, such as Stokes flow or heat conduction, ANSYS POLYFLOW performs two iterations by default. In fact, a second iteration is needed only to verify that the result of the first iteration remains unchanged. Thus, if you know in advance that your problem is linear, you are advised to require only one iteration.

In the **Numerical parameters** menu, select **Modify the numerical parameters for iterations**. Then select **Modify the max number of iterations**.

29.1.3. Convergence and Divergence

You can modify the convergence test value, which is set to 0.001 by default. The convergence test is based on relative error. For each type of field, the modification at every node between two successive iterations is compared to the maximum value of the field at the current iteration. The global test is based on the highest relative variation for most fields (the exceptions including the pressure, the geometrical variables, and all quantities appearing inside parentheses in convergence messages). The default value of the convergence test is appropriate for most problems, except for integral viscoelastic simulations.

To modify the convergence test value, select **Modify the numerical parameters for iterations** in the **Numerical parameters** menu, and then select **Modify the convergence test**.

Not all problems converge. When divergence occurs, the error rapidly increases with each iteration. Since there is usually no hope of convergence when the relative error has become too great, ANSYS POLYFLOW stops the iteration (but not the evolution or time-marching scheme) when the error reaches a preset maximum value (the default is 1000).

To modify the divergence test value, select **Modify the numerical parameters for iterations** in the **Numerical parameters** menu, and then select **Modify the divergence test**.

Most flow problems result in a system of nonlinear algebraic equations. The Newton-Raphson method is available for solving these systems. The quadratic convergence of the Newton-Raphson method means that only a small number of iterations is required, provided the starting solution is not too far from the converged solution. If a solution cannot be achieved, it is recommended that an evolution procedure be tried. See *Evolution* (p. 517) for information on evolution procedures.

During the simulation, if the calculation is converging, but cannot meet the convergence criterion within the specified maximum number of iterations, ANSYS POLYFLOW will automatically perform additional iterations (up to 30% of the specified maximum number) to try to reach the convergence criterion.

29.1.4. Automatic Detection of Distorted Elements

During a calculation that involves remeshing, ANSYS POLYFLOW will automatically detect any distorted mesh elements. If a distorted element is detected, the calculation will continue, but ANSYS POLYFLOW will save results and restart files for the solution just before the distortion is detected. You can then restart the simulation from these files, using a better mesh.

29.1.5. Time-Marching

For time-dependent problems, time-marching scheme parameters apply to all sub-tasks. Note that some numerical parameters specific to a particular sub-task are defined in the **Interpolation** menu of the sub-task. For example, in the case of power law fluids (or, more generally, generalized Newtonian fluids with a low power law index), it has been found that, in many circumstances, the Newton-Raphson method does not converge. It is then necessary to shift from the Newton-Raphson method to Picard iterations, in which the viscosity is updated in terms of the last velocity field. The general recommended procedure is to require a few Picard iterations, followed by the Newton-Raphson method. For details, see *Viscosity-Related Iterations* (p. 221).

You can modify the numerical parameters relevant to the time-marching scheme by selecting **Modify the transient iterative parameters** in the **Numerical parameters** menu.

For more information on time-marching parameters, see *Time-Dependent Flows* (p. 539).

29.1.6. Decoupling Calculations

ANSYS POLYFLOW provides options for decoupling the calculation of various fields, depending on the setup of the simulation.

29.1.6.1. Internal Radiation

A simulation that includes an **Internal radiation** sub-task is computationally expensive. For example, if a 3D domain is discretized into six radiative directions, the simulation will basically require as many variables as a single-mode viscoelastic model. Most of the time, at least 12 directions will be used to reach a reasonable accuracy. Internal radiation calculations are thus good candidates for decoupled schemes, with respect to the demands on both the CPU and memory.

The **Numerical parameters** menu offers the following three options for internal radiation problems. Note that the currently selected option is listed at the top of the menu.

- a coupled calculation

  For a coupled calculation, the velocities, irradiance, and temperature are all solved in the same pass. This is the default approach, and the most expensive from a computational and memory standpoint. If the setup has been changed from the default, you can enable a coupled calculation by selecting **Coupled computation {velocities,irradiance,temperature}** (note that this option is not available until you have already clicked **Decoupled computation velocities/irradiance/temperature**).

  Coupled computation {velocities,irradiance,temperature}

- a partially decoupled calculation

  For a partially decoupled calculation, the momentum equation is solved in a first pass, and then the irradiance and temperature are solved in a second pass. It is often a good choice, because irradiance and temperature are coupled. To enable a decoupled, two-pass calculation, click **Decoupled computation velocities/{irradiance,temperature}** (note that this option is not available until you have already clicked **Coupled computation {velocities,irradiance,temperature}**).

  Decoupled computation velocities/{irradiance,temperature}

- a fully decoupled calculation

  For a fully decoupled calculation, the momentum equation is solved in a first pass, then the irradiance is solved in a second pass, and finally temperature is solved in a third pass. Such a calculation yields iterations that are the most inexpensive from a computational and memory standpoint, but it might require more iterations. It is a good choice when your computer memory is limited. To enable a decoupled, three-pass calculation, click **Decoupled computation velocities/irradiance/temperature** (note that this option is not available until you have already clicked **Decoupled computation velocities/{irradiance,temperature}**).

  Decoupled computation velocities/irradiance/temperature

29.1.6.2. Free Surfaces and Moving Surfaces

When solving moving boundary problems, you can decouple the calculation of the flow field from the node position update. This scheme will exhibit increased numerical stability in some cases. To use this scheme, select **Decoupled computation of moving surfaces** in the **Numerical parameters** menu.

Although it requires more total CPU time than the coupled approach in most cases, the decoupled approach is of interest in a number of applications. In blow molding applications, since the time steps are usually limited by the contact detection algorithm, decoupling can actually result in a significant speedup in the total CPU time.

When the position update is decoupled from the velocity calculation, the computations are made first with an update of the positions, followed by iterations to solve the velocity-pressure-temperature-etc. problem, then back to compute the next position update. The main application of decoupled free surfaces is related to integral viscoelastic models and blow molding applications (see *Additional Strategies for Convergence* (p. 263)).

29.1.6.3. Transport of Species

When a simulation with a **Transport of species** sub-task is defined, ANSYS POLYFLOW needs to compute a species concentration field in addition to the velocity, pressure, and possibly other fields such as temperature, position, and stresses. By default, ANSYS POLYFLOW solves for all of the variables at the same time. In an attempt to reduce the demands on the CPU and memory, you can decouple the computation of the transported variable from the other fields. To enable this scheme, click **Decoupled computation velocities/species** in the **Numerical parameters** menu.

Decoupled computation velocities/species

When you enable this decoupling, the equations are solved in two stages. First, the mass and momentum (and possibly position) equations are solved, considering the species concentration as given. Either the concentration calculated at the previous iteration or the initial concentration is used. Then, the transport equation is solved alone, considering other equations as given. These two stages are repeated until all of the variables have converged. The mass and momentum equations are solved first because the velocities contribute to the transport equation.

The advantage of this technique is the reduction of both memory requirements and CPU time per iteration. The disadvantage is the possibly lower rate of convergence with respect to a fully coupled solver when a strong coupling exists between the momentum and transport equations. In rare cases, the decoupled technique can result in divergence, even though the coupled solver converges. For 2D problems, it is recommended that you always use the coupled technique, because the reduction of CPU time with decoupling will not be significant.

29.1.6.4. Nonisothermal Flows

When a nonisothermal flow problem is defined, ANSYS POLYFLOW needs to compute a temperature field in addition to velocity, pressure, and possibly other fields such as position and stresses. Because the Prandtl number of most polymers is large, the Péclet number of the flow can be high and a refined interpolation might be required for the temperature field, such as the combination of the mini-element and the subdivided 2×2×2 temperature element in 3D.

By default, ANSYS POLYFLOW solves for all variables at the same time. In an attempt to reduce the memory and CPU time for a subset of nonisothermal problems, you can decouple the computation of the velocity-pressure variables from the temperature calculation. To use this scheme, select **Decoupled computation velocities/temperature** in the **Numerical parameters** menu.

When you select this decoupling, the equations are solved in two stages. First, the mass and momentum (and possibly position) equations are solved considering the temperature as given. Either the temperature calculated at the previous iteration or the initial temperature is used. Then, the energy equation is solved alone, considering other equations as given. These two stages are repeated until all variables have converged. The mass and momentum equations are solved first because the velocities contribute to the convective and viscous dissipation terms in the energy equation. If the simulation includes contact detection, like blow molding or thermoforming, the convection and the viscous heating effects are not dominant in the equations. Thus, the sequence of decoupling is reversed: the energy equation is computed first, and then the velocity and pressure are computed.

The advantage of this technique is the reduction of both memory requirements and CPU time per iteration. The disadvantage is the possibly lower rate of convergence with respect to a fully-coupled solver when a strong coupling exists between the momentum and energy equations. The typical case where the technique is useful is when the viscosity of the fluid is independent of the temperature or only slightly dependent, and when natural convection (Boussinesq terms) is neglected. In this case, momentum and continuity equations together, and the energy equation on its own, can be solved one after the other, and the convergence rate is the same as for the coupled technique.

However, when natural convection dominates in the momentum equation, when the temperature strongly influences the viscosity (which is the case for many polymers), neglecting cross momentum-energy derivatives may reduce the rate of convergence. This can result in an increase in the CPU time because the number of iterations may increase more quickly than the CPU time per iteration decreases. In rare cases, the decoupled technique can result in divergence even though the coupled solver converges.

For example, for an Arrhenius law, use the decoupling technique with care if the energy of activation is higher than 3000 and when you expect temperature differences on the order of 30°C or more in the flow. For 2D problems, always use the coupled technique because the reduction of CPU time with decoupling will not be significant.

Important

A dependence of the temperature upon the velocity, through the advection or viscous dissipation terms, for example, will not affect the rate of convergence. Even in the case when the Péclet number is high, the advection term itself will not degrade convergence (but it might affect the solution accuracy if the mesh is not sufficiently fine).

29.1.6.5. Combining Decoupled Calculations

If you decouple the calculations for temperature, moving parts, and species, all other fields will be computed with the velocity field while the temperature, the positions, and the species will be computed together in a second sequence.

In the case of a setup with internal radiation, irradiance is always computed in the second pass. The temperature, positions, and/or species are always computed in the last pass of the calculation, which may be the second pass or the third pass, depending on the selected strategy for radiation. In the former case, irradiance is computed in the second pass with temperature, positions, and/or species; in the latter case, irradiance is computed alone in the second pass while temperature, positions, and/or species are computed in the third pass.

29.2. Solver Details

29.2.1. Introduction

The basic problem that ANSYS POLYFLOW must solve at a given time step or iteration is a linear algebraic system of the following form:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{29–1}$$

The matrix $\mathbf{A}$ is typically large, sparse, and asymmetric. ANSYS POLYFLOW version 3.12 includes a series of new direct and iterative solvers that are alternatives to the traditional, mesh-based multifrontal solver (now referred to as the Classic Direct Solver, see *Classic Direct Solver* (p. 580)). These new solvers are available to you in the **Numerical parameters** menu of the **F.E.M. Task**. In the **Numerical parameters** menu, select the **Modify numerical parameters for iterations** item to access the **Numerical parameters for iterations** menu. Here, you can select the **Modify the solver agressivity** item and choose one of the available solvers that are described in this chapter.

29.2.2. Classic Direct Solver

ANSYS POLYFLOW uses a frontal method to solve *Equation 29–1* (p. 579) (with potentially multiple fronts created by domain decomposition). That is, it uses a direct method based on Gaussian elimination.

The basic principle of the frontal method is that the assembly of each element’s contribution to the matrix $\mathbf{A}$ and the vector $\mathbf{b}$ can be immediately followed by a Gaussian elimination process on some of the element variables. In fact, the full matrix $\mathbf{A}$ is never formed. The frontal procedure instead goes through the following steps for each element:

1. Assemble the element contribution to $\mathbf{A}$ and $\mathbf{b}$.
2. Eliminate by Gaussian elimination the element degrees of freedom that will no longer contribute to $\mathbf{A}$ and $\mathbf{b}$.

The frontal method requires only the currently active (incompletely assembled) equations to be stored in central memory. Following the complete transformation of $\mathbf{A}$ into an upper-triangular form, the solution vector $\mathbf{x}$ is obtained by back-substitution.

29.2.2.1. ANSYS POLYFLOW’s Implementation

By default, ANSYS POLYFLOW incorporates a very efficient “cache-optimized" direct solver, which improves the computer efficiency by a factor of 2 to 5 by improving the locality of the data during the frontal elimination. Instead of performing a sequential elimination of the variables, unknowns ready for elimination are clustered in blocks, a (small) matrix is inverted, and the complete block is eliminated from subsequent equations. This “block" elimination of the variables dramatically improves the performance on most hardware platforms, which are extremely cache-sensitive. Another very important factor is that the performance remains mostly unaffected by the problem size.

This block solver makes use of a BLAS3 (Basic Linear Algebra Subroutine, see

`www.netlib.org/blas`, especially `blas3-paper.ps`, for details) implementation of the Gaussian elimination, and specifically of the DGEMM subroutine. If necessary, the size of the blocks can be adjusted by modifying the `BLOCS` entry of the `.p3rc` file. The default corresponds to

```
BLOCS 20
BLAS LEVEL3
```

and this is the recommended value for most superscalar and even vector computers.

Although you can set the BLAS level to other values, it is unlikely to improve performance.

29.2.2.2. Mesh Decomposition and Optimization

Optimization of the element numbering is essential in order to obtain the best performance from the ANSYS POLYFLOW solver. ANSYS POLYFLOW performs the optimization automatically by default, as described in *Mesh Decomposition and Optimization* (p. 146).

For large 3D meshes, decomposition of the mesh into subparts is also recommended to improve the solver performance and to take advantage of the multiple fronts. For very large meshes, the speedup and memory savings can be extremely large, typically 2 to 10, possibly up to 20; for this reason, ANSYS POLYFLOW performs mesh decomposition automatically, as described in *Mesh Decomposition and Optimization* (p. 146).

29.2.2.3. Solver Robustness

The block solver is not quite as robust as the sequential solver (which can be used by setting `BLOCS 1` and `BLAS  LEVEL1` in the `.p3rc` file) for solving ill-conditioned systems of equations. It is very unlikely, however, that you will have any need to change from the default block solver.

29.2.3. AMF Direct Solver

The default solver is now an algebraic multifrontal (AMF) direct solver (rather than mesh-based) where the critical variable ordering is based on the assembled equations, having the advantage of taking into account more sparsity in many systems of equations (e.g., in multimode viscoelastic flows where stress modes are uncoupled).

Factors are, by default, written to disk at a location dictated by the `TMP` (Linux) or `TEMP` (Windows) environment variables. It is also possible to force storage of those factors in the core by entering `IPIN-CORE` in the `.p3rc` file.

29.2.4. AMF Direct Solver + Secant

In many nonlinear problems, it is possible to take advantage of the system regularity between iterations in order to factorize the system matrix only from time to time. This is the so-called “secant" or “modified Newton" iterative process. This technique is known to work in most problems with potentially significant savings (as the CPU-dominant system solution step is performed less frequently and system assembly can be done at a reduced cost), however, in very rare cases, the secant iteration may direct iterations along a wrong path, therefore, this option is not the default. Note that this technique requires slightly more disk space for storing the factors than is the case for the pure Newton-Raphson scheme, so do not expect memory savings from this technique.

29.2.5. AMF Direct Solver + ILU

While slightly increasing the “risk" level of the solver, it is possible to switch to an iterative strategy where a combination of solvers is used via the AMF direct solver + Incomplete L-U factorization (ILU) method. In this case, terms are being dropped in the system matrix (but not in the Right Hand Side so that the solution remains identical) and the AMF direct solver is used as a preconditioner. The terms that are dropped are decided based on block descriptions and problem formulation. Of course, the resulting system is cheaper to solve, but dropping those terms prevents the solution from being computed in one step, and it is then necessary to embed the process in a global iterative strategy represented by a Generalized Minimal Residual Method (GmRes) [*23*] (p. 716). This strategy is most efficient in systems involving many blocks, such as moving coordinates or multimode viscoelastic problems, but is unfortunately inefficient in simple cases such as fixed domain, velocity/pressure systems, because opportunities to discover weak couplings in such systems are limited. In most cases, the ILU solver requires less memory than a pure direct solver.

29.2.6. AMF Direct Solver + Secant + ILU

It is possible to combine the previously described secant strategy with the ILU solver and to take advantage of both approaches. This approach is know to work in more than 90% of the ANSYS POLYFLOW problems, but saving on the CPU/memory cannot be guaranteed. In order to consider this option, both secant and ILU strategies should be of interest individually.

29.2.7. Multigrid Iterative Solver

Finally in systems of mostly linear nature that do not include several blocks of equations (such as large velocity/pressure simulations on a fixed mesh, in MST cases for example), none of the above strategies might prove efficient. An independent, pure iterative solver has therefore been included that is based on an Algebraic MultiGrid approach (AMG) combined with an outer iterative strategy such as BiCGStab (Bi-Conjugate Gradient Stabilized). This method will only work on a subset of systems that do not involve moving coordinates, viscoelasticity, or other forms of nonlinearity, but this technique might prove useful on large 3D meshes, both in memory and CPU. Note that the AMG/BiCGStab solver may be counterproductive unless the mesh becomes very large. See [*27*] (p. 716) and [*30*] (p. 716) for details on the AMG approach and the BiCGStab strategy respectively.

29.2.8. Delaying Policy for Stabilized Pressure

The **delaying policy** option in the **Numerical parameters for iterations** menu becomes available if the pressure is stabilized (option available for generalized Newtonian fluids in the **Interpolation** menu of the flow sub-task) and if one of the AMF solvers (described earlier) has been selected. When the pressure is stabilized, one adds nonzero (but small) diagonal terms to the pressure equations. With such terms, it is no more strictly necessary to delay the elimination of a pressure variable after the elimination of its neighboring velocities; zero pivots are avoided, and the solver is much more efficient (in memory and in CPU). There is, however, a small risk that the propagation of numerical error during elimination is higher leading to the divergence of the solver. This seems more frequent with the ILU method. That is why the default value is **pressure not delayed** for the AMF direct solver (with or without secant) and the default value is **pressure delayed** for the ILU solver. Of course, for non stabilized pressure, delaying is always needed and activated.

29.3. Analyzing Solver Performance

To check the status of the solver during or after the calculation, and to analyze the solver performance, you can use ANSYS POLYDIAG, the diagnostic tool for ANSYS POLYFLOW.

29.3.1. Starting ANSYS POLYDIAG

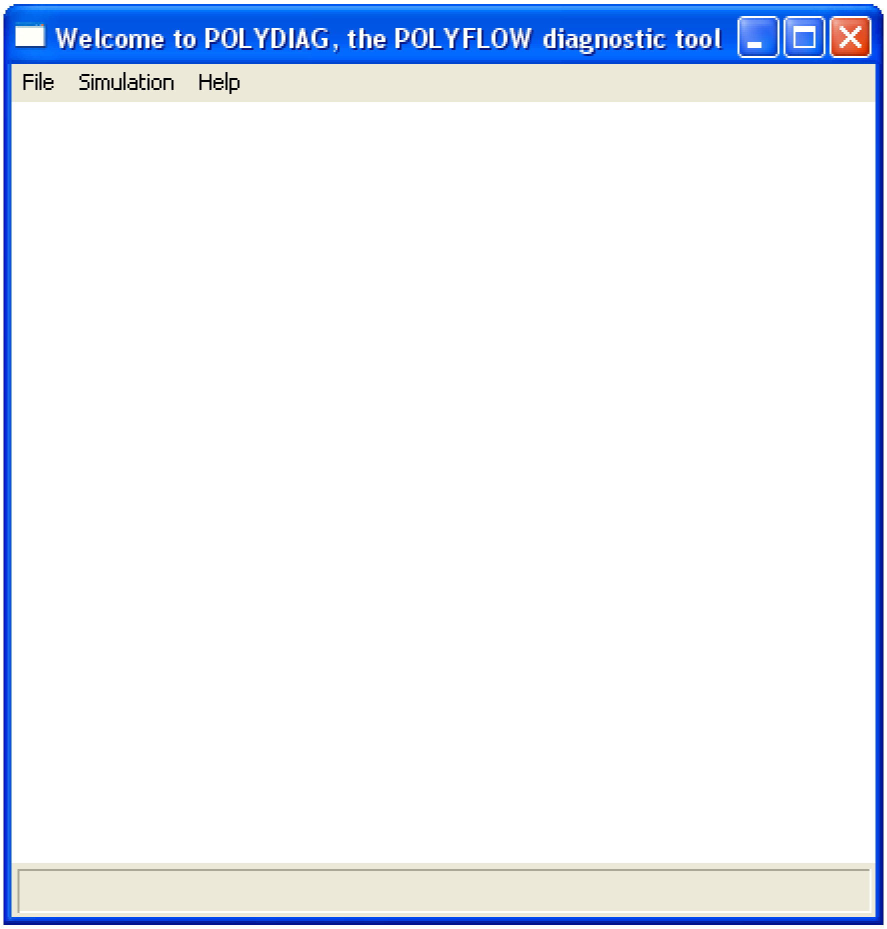
To start ANSYS POLYDIAG, type

```
polydiag
```

at the command prompt.

The main ANSYS POLYDIAG window will appear as shown in *Figure 29.1* (p. 583).

Figure 29.1 The Main ANSYS POLYDIAG Window



You can also start ANSYS POLYDIAG from ANSYS POLYMAN, as described in *Starting the Programs* (p. 40).

There are three menus: **File**, **Simulation**, and **Help**. The items in the **File** menu are used for loading and updating simulation information, and for exiting from ANSYS POLYDIAG. See *Loading, Viewing, and Updating the Simulation Status* (p. 583) and *Exiting ANSYS POLYDIAG* (p. 596) for details. The **Simulation** menu contains a single item for viewing the simulation status. See *Loading, Viewing, and Updating the Simulation Status* (p. 583) for details. Finally, the **Help** menu contains a single option (**How to use Polydiag**) that, when clicked, provides an overview of how to use ANSYS POLYDIAG.

29.3.2. Loading, Viewing, and Updating the Simulation Status

To use ANSYS POLYDIAG to check the status of a currently-running ANSYS POLYFLOW simulation or to analyze a completed calculation, follow these steps:

1. Load the simulation status file, called `instances.clp`.

   File → **Load Instance file**

2. View the current status of the simulation.

   Simulation → **Current status**

   See *Evolution Information* (p. 584) – *Mass Balance Report* (p. 588) for a description of the information you can view. You can also save this information to a file, as described in *Writing Status Information to a File* (p. 590).

3. Periodically update the status of the simulation.

   File → **Update**

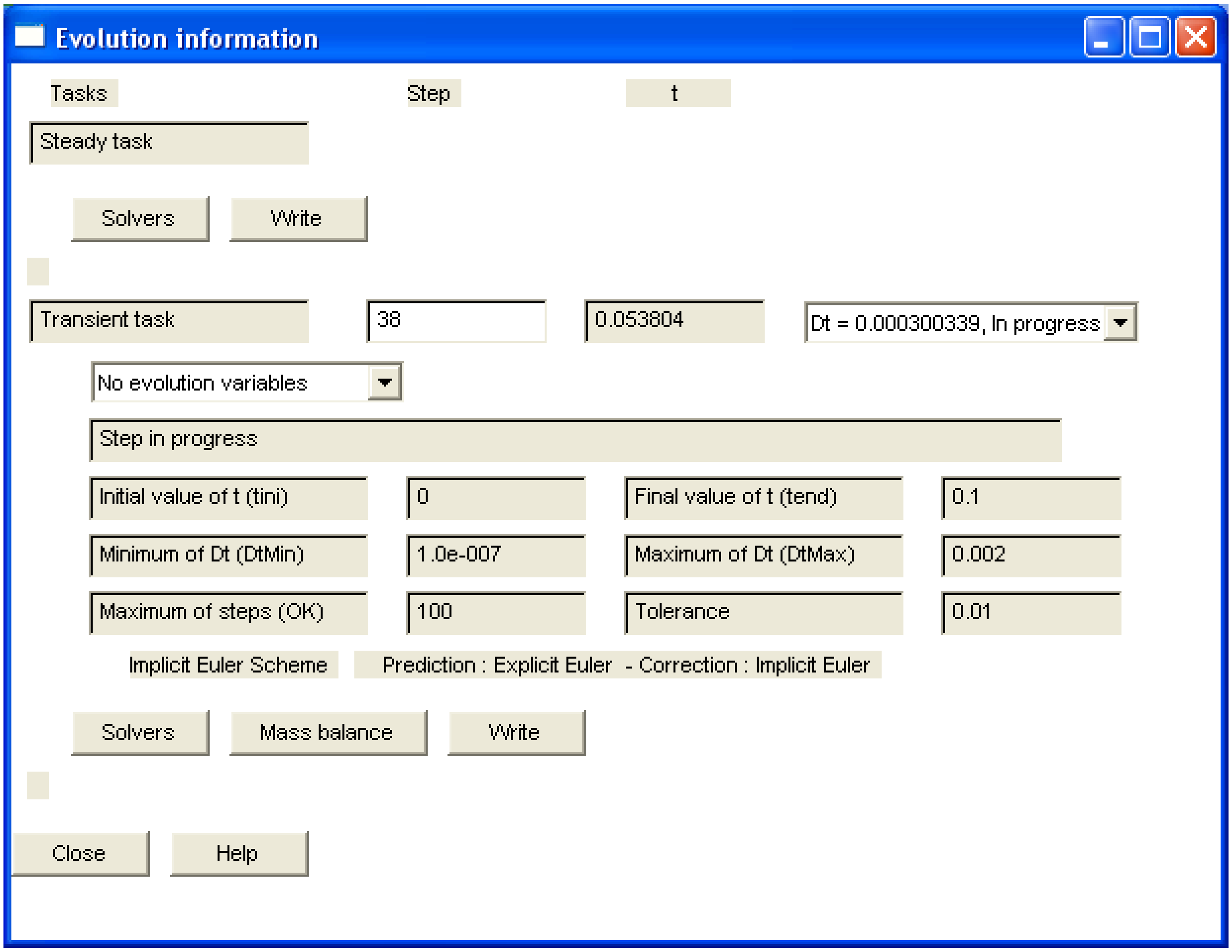
   (If you are viewing information for a simulation that has already finished, this step will not be necessary.)

General information about the simulation (e.g., indication that the computation succeeded or some expert advice) will be printed in the main ANSYS POLYDIAG window, and more specific information about different aspects of the simulation can be viewed in the **Evolution information** panel, and in other panels that are opened from within it. See *Evolution Information* (p. 584) – *Mass Balance Report* (p. 588) for details.

29.3.3. Evolution Information

When you select the **Simulation/Current status** menu item, ANSYS POLYDIAG will open the **Evolution information** panel (*Figure 29.2* (p. 584)). The **Evolution information** panel presents the following information for each task (only the first and last lines for a steady task):

Figure 29.2 The Evolution information Panel



Evolution information

Tasks	Step	t	
Steady task			
Transient task	38	0.053804	Dt = 0.000300339, In progress

No evolution variables

Step in progress

Initial value of t (tini)	0	Final value of t (tend)	0.1
Minimum of Dt (DtMin)	1.0e-007	Maximum of Dt (DtMax)	0.002
Maximum of steps (OK)	100	Tolerance	0.01

Implicit Euler Scheme   Prediction : Explicit Euler - Correction : Implicit Euler

- First line: name of the task, step index, value of the evolution parameter (*S*) or time (*t*), and the increment of *S* or *t*. For a steady task, only the name will appear on this line.

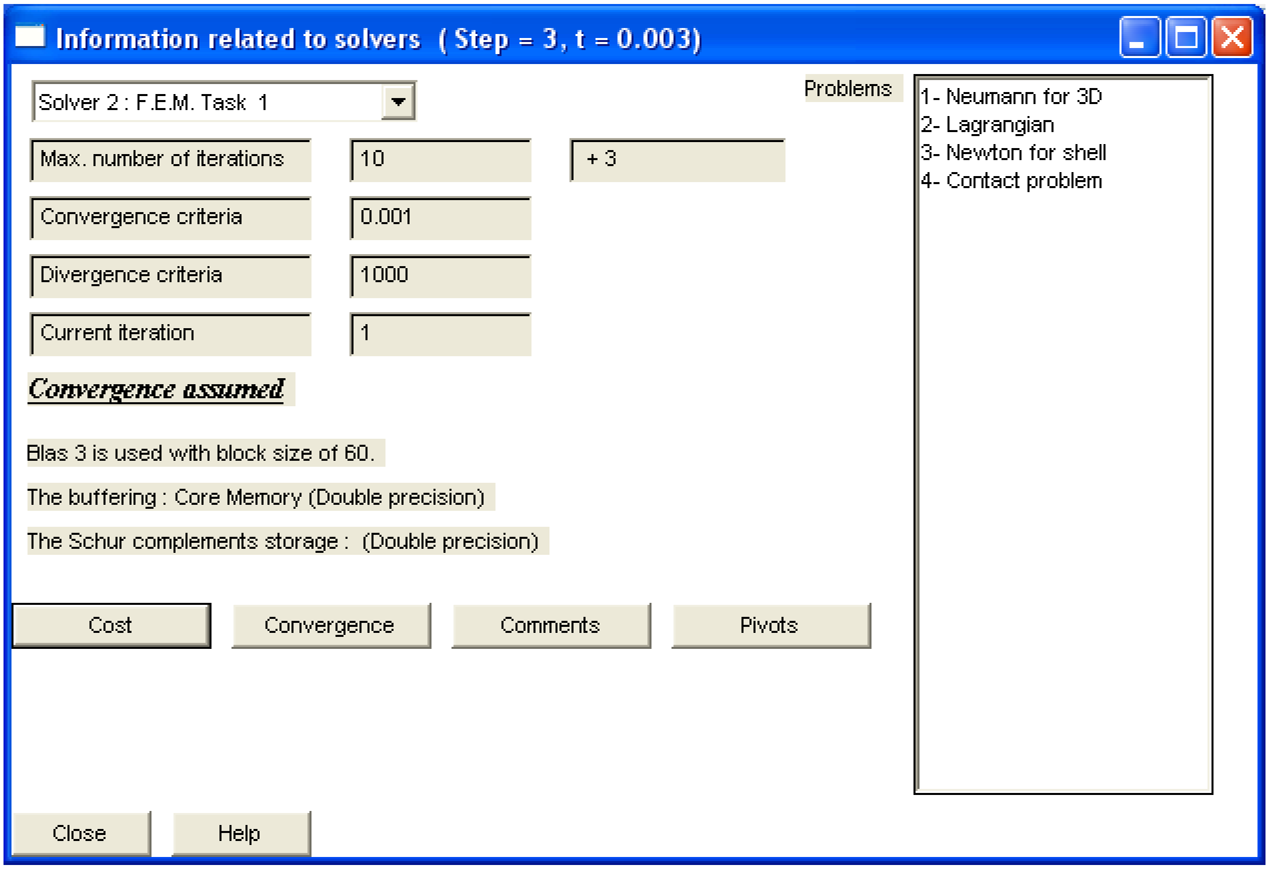
Note that the field for the increment of *S* or *t* will include a status of **OK**, **KO**, or **In progress**. **OK** indicates that the step was successful, **KO** indicates that it was not, and **In progress** indicates that it is still being calculated.

- Second line: list of evolution variables and their values (or an indication of **No evolution variables**, as in *Figure 29.2* (p. 584), which is for a time-dependent task).
- Third line: message about the success or failure of the current step.
- Fourth line: Initial and final values of *S* or *t*.
- Fifth line: minimum and maximum values of *dS* or *dt*.
- Sixth line: maximum number of successful steps.
- Seventh line: type of evolution (or transient) scheme.
- Last line: **Solvers**, **Mass balance**, **Interruption of evolution**, and **Write** buttons, which are described in *Solver Information* (p. 585), *Mass Balance Report* (p. 588), *Interrupting Evolution* (p. 589), and *Writing Status Information to a File* (p. 590).

29.3.4. Solver Information

To open the **Information related to solvers** panel (*Figure 29.3* (p. 585)), click the **Solvers** button for a particular task in the **Evolution information** panel.

Figure 29.3 The Information related to solvers Panel



Information related to solvers (Step = 3, t = 0.003)

Solver 2 : F.E.M. Task 1

Max. number of iterations	10	+ 3
Convergence criteria	0.001	
Divergence criteria	1000	
Current iteration	1	

Convergence assumed

Blas 3 is used with block size of 60.

The buffering : Core Memory (Double precision)

The Schur complements storage : (Double precision)

Problems

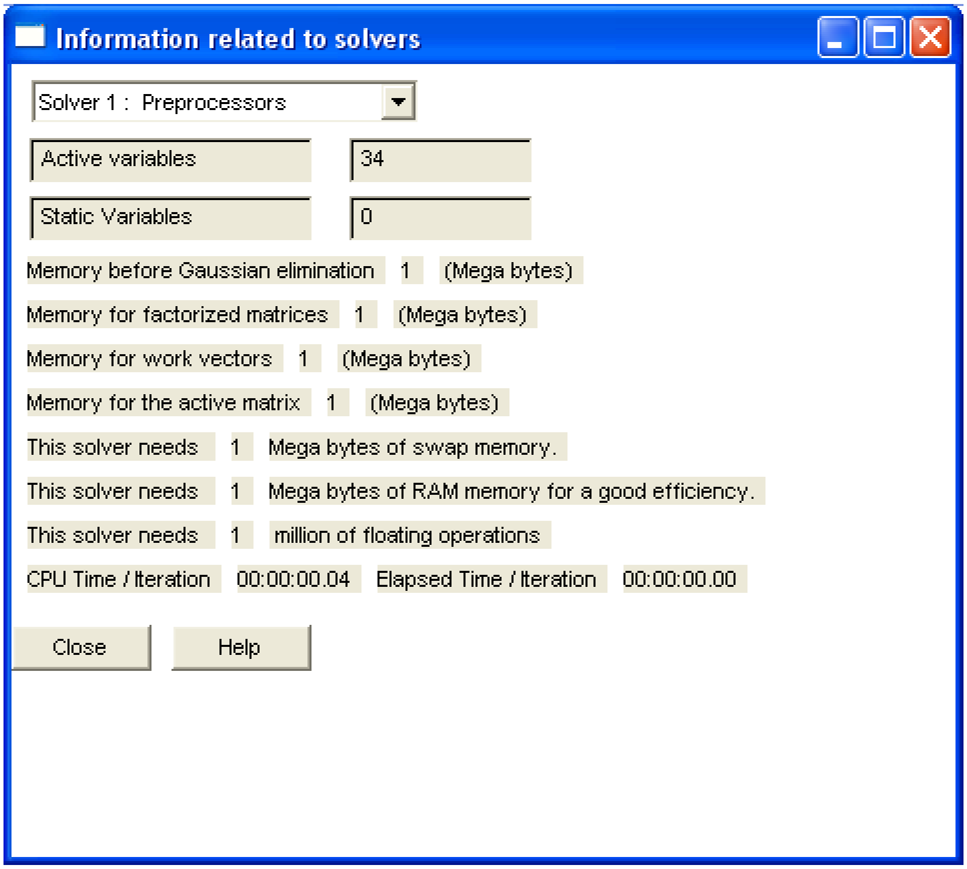
1- Neumann for 3D
2- Lagrangian
3- Newton for shell
4- Contact problem

The **Information related to solvers** panel presents the following information for the selected task:

- First line: the name of the selected task and (on the right side of the panel) the list of problems it is solving.
- Second line: the specified maximum number of iterations and the maximum number of iterations added by the expert system. The number of iterations added depends on the past convergence profile.
- Third line: convergence criterion.
- Fourth line: divergence criterion.
- Fifth line: current iteration.
- Sixth line: indication of convergence or divergence.
- Seventh line: information about BLAS.
- Eighth line: information about buffering.
- Ninth line: information about storage.
- Last line: **Cost**, **Convergence**, **Comments**, and **Pivots** buttons, which are described below.

To get the same information about another step, change the step number in the **Evolution information** panel. If you click the **Cost** button in the **Information related to solvers** panel, the panel shown in *Figure 29.4* (p. 586) will open.

Figure 29.4 Cost Information



Information related to solvers

Solver 1 : Preprocessors

Active variables	34
Static Variables	0

Memory before Gaussian elimination 1 (Mega bytes)

Memory for factorized matrices 1 (Mega bytes)

Memory for work vectors 1 (Mega bytes)

Memory for the active matrix 1 (Mega bytes)

This solver needs 1 Mega bytes of swap memory.

This solver needs 1 Mega bytes of RAM memory for a good efficiency.

This solver needs 1 million of floating operations

CPU Time / Iteration 00:00:00.04 Elapsed Time / Iteration 00:00:00.00

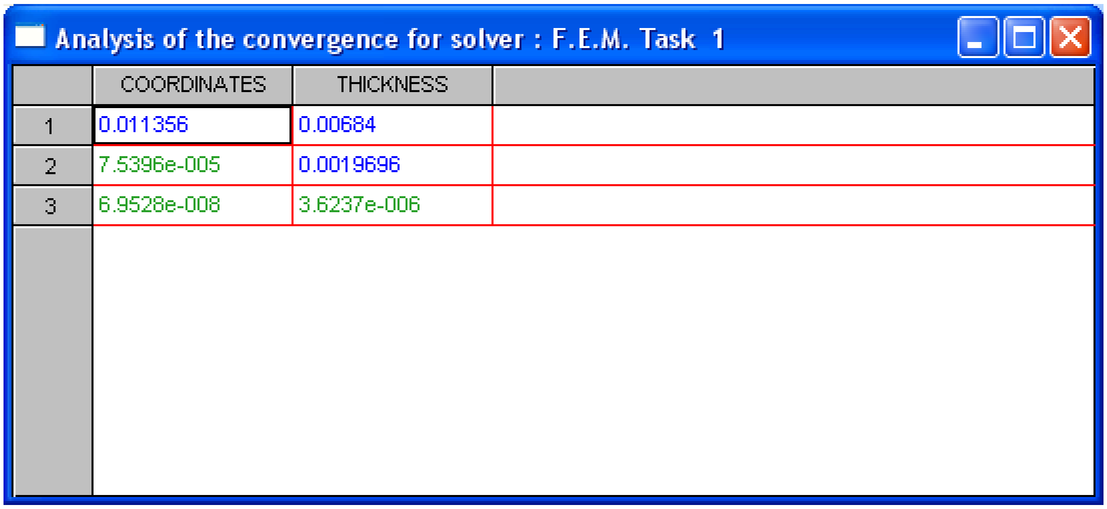
The following information is presented:

- First line: the name of the selected task.
- Second line: number of active variables.

- Third line: number of static variables.
- Fourth through ninth lines: memory requirements.
- Tenth line: number of floating point operations needed to solve the linear system of equations.
- Last line: average CPU time and elapsed time per iteration.

If you click the **Convergence** button in the **Information related to solvers** panel, the window shown in *Figure 29.5* (p. 587) will open. This window shows the relative variations of each computed field at each iteration. Note that converged iterations are shown in green, iterations that diverged are shown in red, and iterations in progress are shown in blue. The yellow cells correspond to the secant iterations, whereas the white cells correspond to the Newton-Raphson iterations.

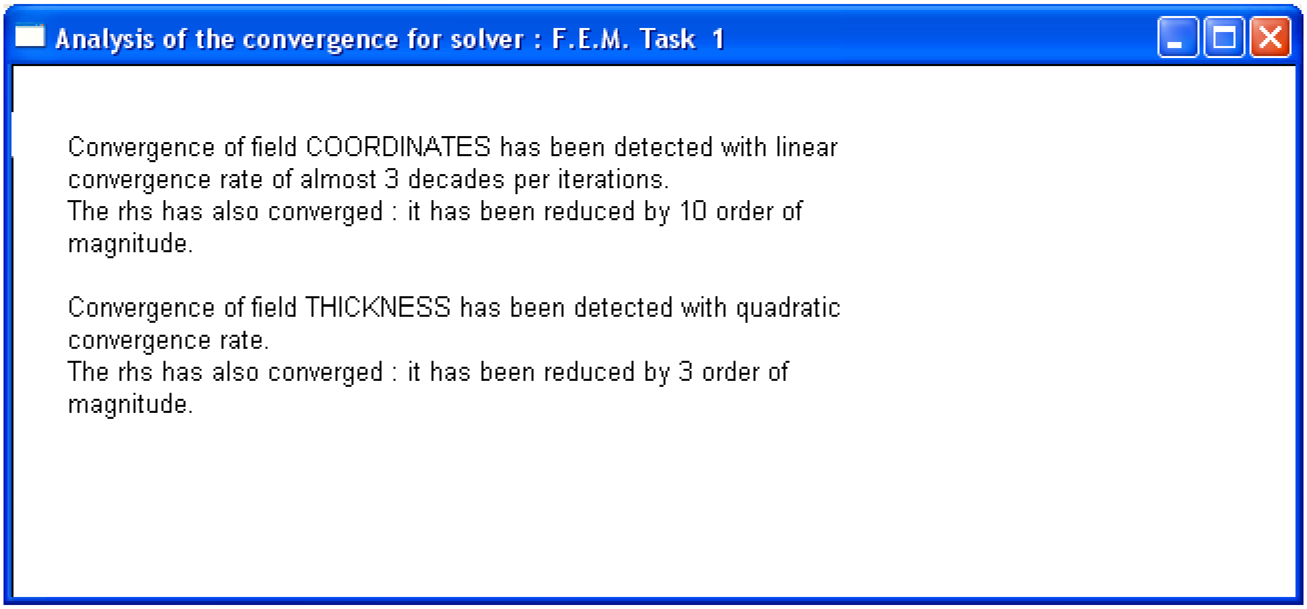
Figure 29.5 Convergence Information



	COORDINATES	THICKNESS
1	0.011356	0.00684
2	7.5396e-005	0.0019696
3	6.9528e-008	3.6237e-006

If you click the **Comments** button in the **Information related to solvers** panel, the window shown in *Figure 29.6* (p. 587) will open. This window lists any comments related to the convergence of the solver.

Figure 29.6 Convergence Comments

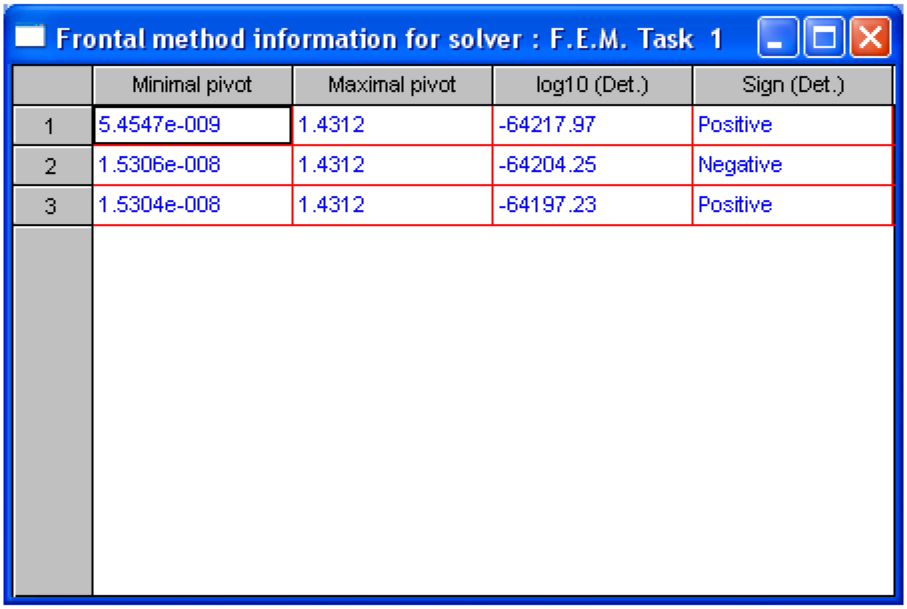


Convergence of field COORDINATES has been detected with linear convergence rate of almost 3 decades per iterations.
The rhs has also converged : it has been reduced by 10 order of magnitude.

Convergence of field THICKNESS has been detected with quadratic convergence rate.
The rhs has also converged : it has been reduced by 3 order of magnitude.

If you click the **Pivots** button in the **Information related to solvers** panel, the window shown in *Figure 29.7* (p. 588) will open. This window lists pivot information for the frontal method solver at each iteration.

Figure 29.7 Pivot Information



Frontal method information for solver : F.E.M. Task 1

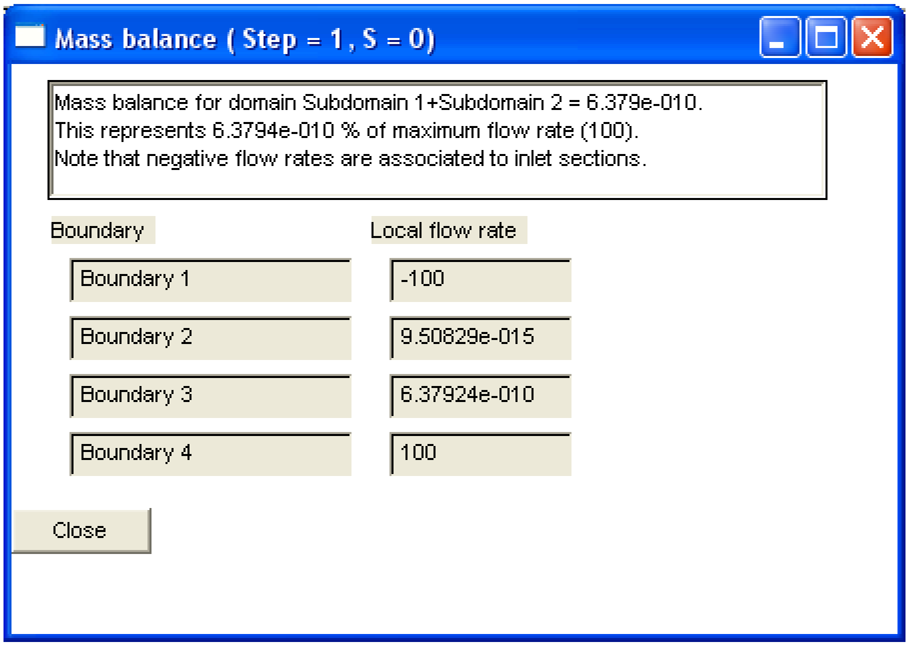
	Minimal pivot	Maximal pivot	log10 (Det.)	Sign (Det.)
1	5.4547e-009	1.4312	-64217.97	Positive
2	1.5306e-008	1.4312	-64204.25	Negative
3	1.5304e-008	1.4312	-64197.23	Positive

Decreasing pivots during the course of the iterations and a change of sign in the Jacobian matrix are indications of a singular matrix, for which a solution will not be possible.

29.3.5. Mass Balance Report

To open the **Mass balance** panel (*Figure 29.8* (p. 588)), click the **Mass balance** button for a particular task in the **Evolution information** panel.

Figure 29.8 The Mass balance Panel



Mass balance (Step = 1 , S = 0)

Mass balance for domain Subdomain 1+Subdomain 2 = 6.379e-010.
This represents 6.3794e-010 % of maximum flow rate (100).
Note that negative flow rates are associated to inlet sections.

Boundary	Local flow rate
Boundary 1	-100
Boundary 2	9.50829e-015
Boundary 3	6.37924e-010
Boundary 4	100

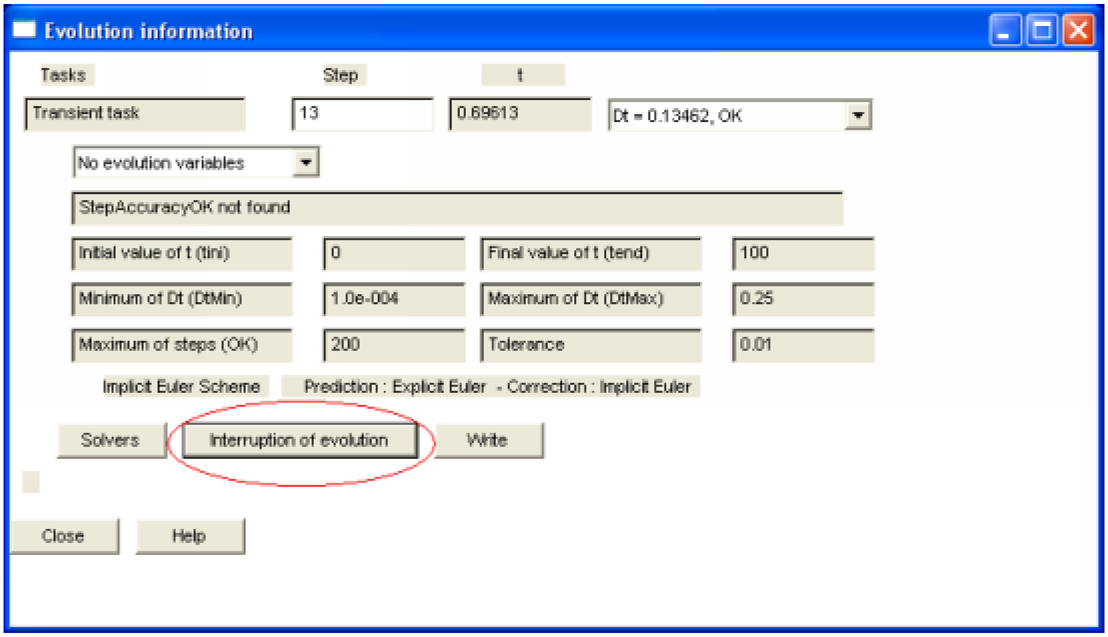
The **Mass balance** panel presents the following information for the selected task:

- First line: summary of the global mass balance for each subdomain where a flow problem has been solved.
- Remaining lines: flow rate through each boundary.

29.3.6. Interrupting Evolution

If you have defined some criteria to interrupt the evolution, the **Interruption of evolution** button is accessible in the lower part of the **Evolution information** panel as shown in *Figure 29.9* (p. 589).

Figure 29.9 The Evolution information Panel with the Interruption of evolution Button



Evolution information

Tasks	Step	t	
Transient task	13	0.69613	Dt = 0.13462, OK

No evolution variables

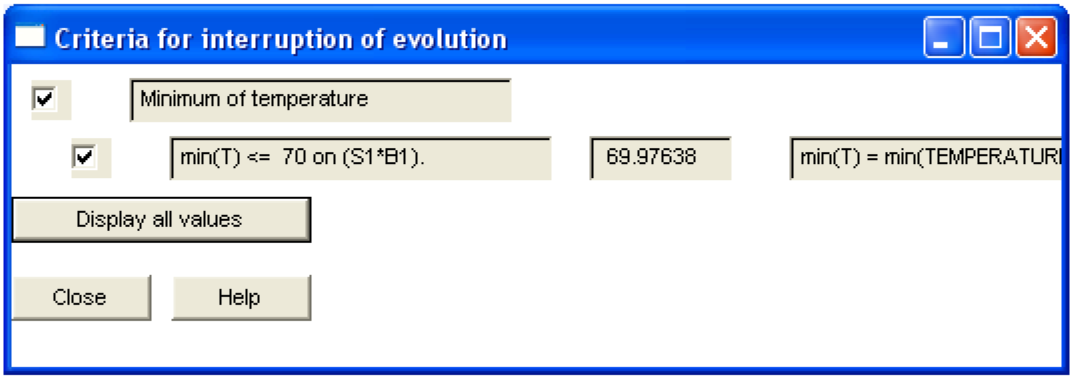
StepAccuracyOK not found

Initial value of t (tini)	0	Final value of t (tend)	100
Minimum of Dt (DtMin)	1.0e-004	Maximum of Dt (DtMax)	0.25
Maximum of steps (OK)	200	Tolerance	0.01

Implicit Euler Scheme   Prediction : Explicit Euler - Correction : Implicit Euler

Click the **Interruption of evolution** button to display the current status of all criteria in the **Criteria for interruption of evolution** panel (*Figure 29.10* (p. 589)).

Figure 29.10 The Criteria for interruption of evolution Panel



Criteria for interruption of evolution

☑ Minimum of temperature

☑ min(T) <= 70 on (S1*B1).   69.97638   min(T) = min(TEMPERATURI

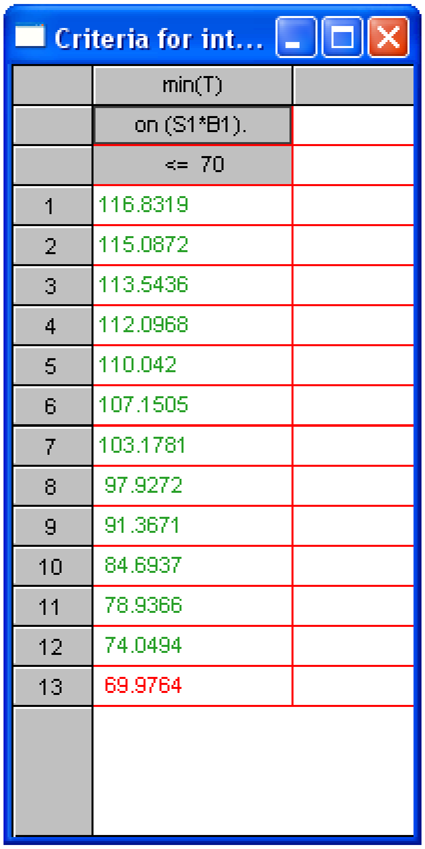
In this panel, the criteria are gathered by group, for which there are the following:

- A line for the group information.
  - A check box flagged if all criteria of the group are satisfied.

  - The name of the group of criteria.
- An indented line for each criteria.
  - A check box flagged if the criteria is satisfied.
  - A short description of the criteria.
  - The value of the criteria.
  - A long description of the criteria.
- After all criteria information, there is the **Display all values** button. This button allows you to display all criteria values for all evolution steps.

The **Display all values** button opens another version of the **Criteria for interruption of evolution** panel (*Figure 29.11* (p. 590)). This panel displays the criteria values in a grid where the rows correspond to the successful steps and a column corresponds to a criteria. The group of criteria are separated by a thick red line. If the criteria is satisfied, its value is displayed in red. Likewise, if the criteria is not satisfied, the value is displayed in green. If you click the **Display all values** button while the associated panel is already displayed, the latter version will be closed.

Figure 29.11 The Criteria for interruption of evolution Panel

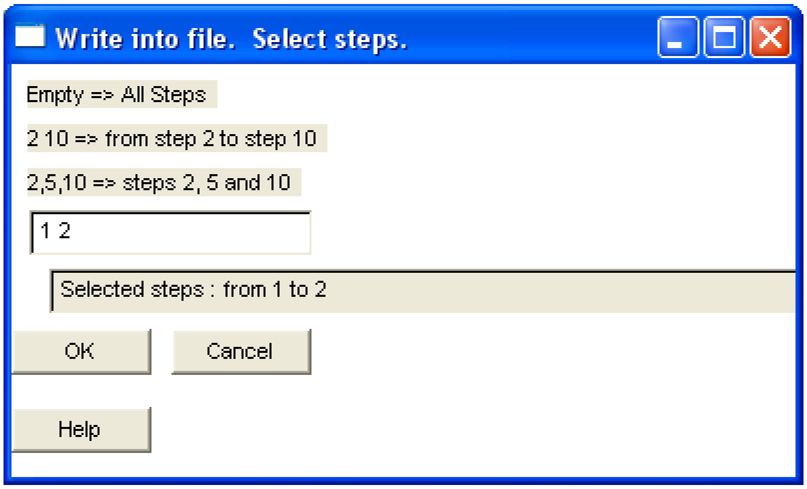


	min(T)	
	on (S1*B1).	
	<= 70	
1	116.8319	
2	115.0872	
3	113.5436	
4	112.0968	
5	110.042	
6	107.1505	
7	103.1781	
8	97.9272	
9	91.3671	
10	84.6937	
11	78.9366	
12	74.0494	
13	69.9764	

29.3.7. Writing Status Information to a File

To write the simulation status information for a particular task to a file, click the **Write** button for that task in the **Evolution information** panel. ANSYS POLYDIAG will open the **Write into file** panel (*Figure 29.12* (p. 591)).

Figure 29.12 The Write into file Panel



Write into file. Select steps.

Empty => All Steps

2 10 => from step 2 to step 10

2,5,10 => steps 2, 5 and 10

1 2

Selected steps : from 1 to 2

Follow these steps to save the information to a file:

1. Specify the steps for which you want to save the information:

   - To save all steps, leave the field on the fourth line empty (as in *Figure 29.12* (p. 591)).
   - To save a range of steps from step a to step b, enter a b in the field.
   - To save one or more nonsequential steps (e.g., steps 5, 10, and 15), enter them separated by commas (e.g., `5,10,15`).

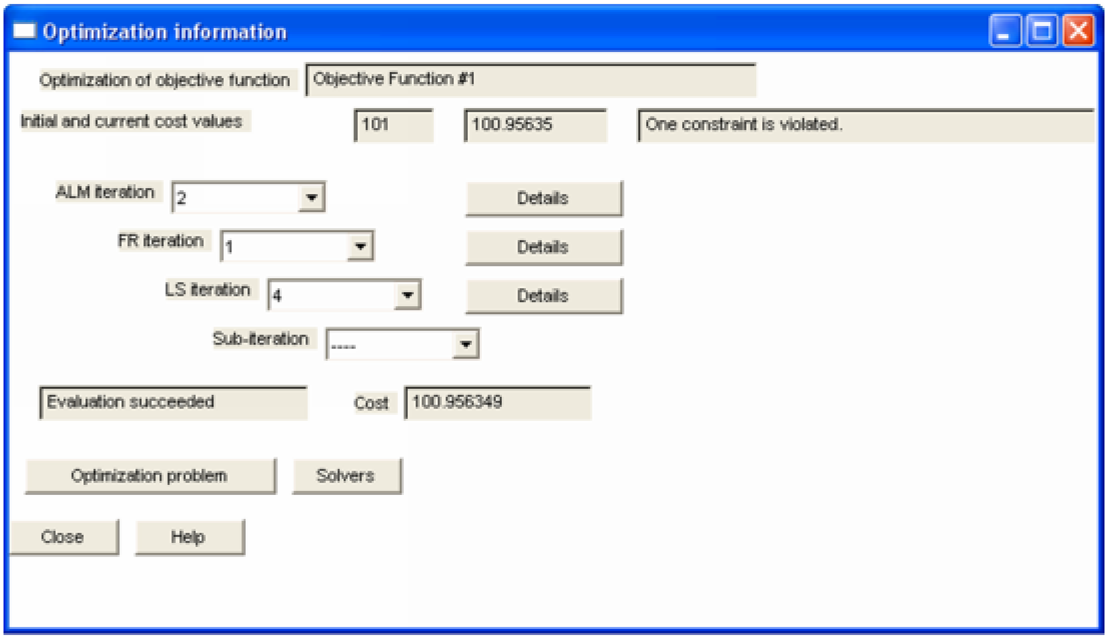
2. Click **OK**, and specify the file name when prompted.

29.3.8. Optimization Information

When you select the **Simulation/Current status** menu item in ANSYS POLYDIAG after ANSYS POLYFLOW has started the optimization, the **Optimization information** panel

Simulation → **Current status**

Figure 29.13 The Optimization information Panel



Optimization of objective function: Objective Function #1

Initial and current cost values: 101 | 100.95635 | One constraint is violated.

ALM iteration: 2

FR iteration: 1

LS iteration: 4

Sub-iteration: -----

Evaluation succeeded

Cost: 100.956349

The **Optimization information** panel provides the following text boxes and buttons so that you can view information about the simulation:

- the **Optimization of objective function** text box

  This displays the name of the objective function being optimized. When there are multiple objective functions defined using the multipriority scheme, you can select the objective function for which you want to obtain information.

- the **Initial and current cost values** text boxes

  These text boxes provide information about the cost and status of the optimization. The text box on the left displays the initial cost, which is calculated during the first evaluation of the solution at the beginning of the optimization. The remaining text boxes display the current cost and constraint status, which correspond to the most recent evaluation of the solution.

- the **ALM iteration** drop-down list and **Details** button

  You can display the details of all of the completed ALM iterations by clicking the **Details** button next to the **ALM iteration** drop-down list. Panels will open that display the values of the design variables, the objective/cost functions, and the constraints for the ALM iterations. See *Design Variable Information* (p. 594) – *Constraint Information* (p. 596) for details about these panels.

  If you want information about the FR or LS iterations of a particular ALM iteration, select the index number from the **ALM iteration** drop-down list (the most recent ALM iteration is selected by default). Then use the **FR iteration** and **LS iteration** drop-down lists and **Details** button, as explained in the descriptions that follow.

- the **FR iteration** drop-down list and **Details** button

  You can display the details of all of the FR iterations of the ALM iteration selected in the **ALM iteration** drop-down list, simply by clicking the **Details** button next to the **FR iteration** drop-down list. Panels will open that display the values of the design variables, the objective/cost functions, and the constraints for the FR iterations. See *Design Variable Information* (p. 594) – *Constraint Information* (p. 596) for details about these panels.

If you want information about the LS iterations of a particular FR iteration, select the index number from the **FR iteration** drop-down list (the final FR iteration of the previously selected ALM iteration is selected by default). Then click the **Details** button next to the **LS iteration** drop-down list, as explained in the description that follows.

- the **LS iteration** and **Sub-iteration** drop-down lists and **Details** button

  You can display the details of all of the LS iterations and sub-iterations of the ALM and FR iteration selected in the **ALM iteration** and **FR iteration** drop-down lists, respectively. Simply click the **Details** button next to the **LS iteration** drop-down list. Panels will open that display the value of α (see *The Line Search (LS) Method* (p. 659)), the design variables, the objective / cost functions, and the constraint as a function of the LS iterations and sub-iterations. See *Design Variable Information* (p. 594) – *Constraint Information* (p. 596) for details about these panels.

  If you want information about the cost or evaluation status of a particular LS iteration or sub-iteration, select the index number from the **LS iteration** drop-down list (and the **Sub-iteration** drop-down list, if necessary). Then view the **Cost** text box and the text box on its left, as explained in the description that follows. By default, the final LS iteration and sub-iteration numbers of the previously selected ALM and FR iterations are selected. Note that when there are no sub-iterations associated with a LS iteration, "**- - - -**" is displayed in the **Sub-iterations** drop-down list.

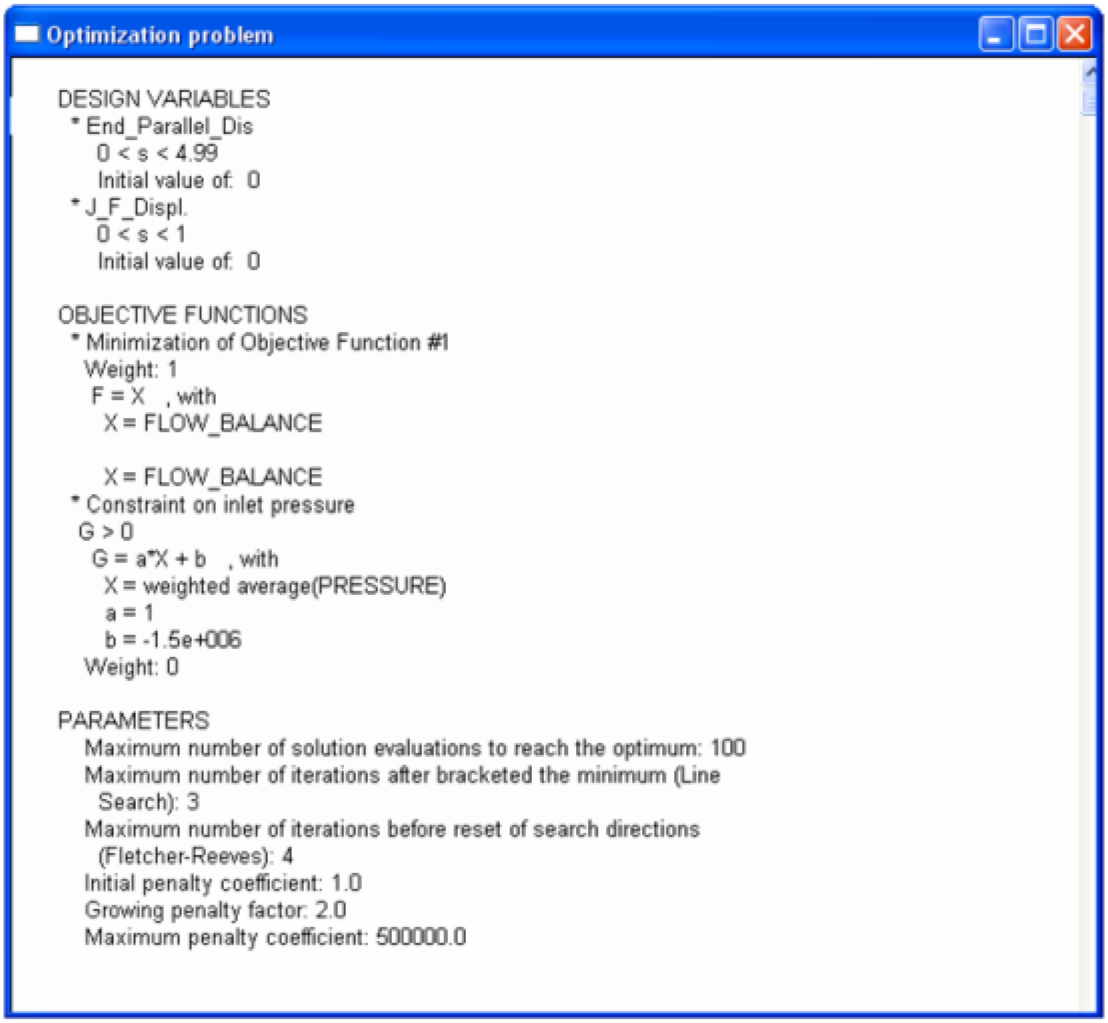
- the **Cost** text boxes

  The **Cost** text box and the text box on its left display the cost and status of the evaluation, respectively, corresponding to the ALM, FR, and LS iterations and sub-iterations selected previously.

- the **Optimization problem** button

  Click the **Optimization problem** button to open the **Optimization problem** panel, which displays information about the design variables, objective functions, and constraints. An example of this panel is shown in *Figure 29.14* (p. 594).

Figure 29.14 The Optimization problem Panel



```
DESIGN VARIABLES
 * End_Parallel_Dis
    0 < s < 4.99
    Initial value of: 0
 * J_F_Displ.
    0 < s < 1
    Initial value of: 0

OBJECTIVE FUNCTIONS
 * Minimization of Objective Function #1
   Weight: 1
    F = X   , with
     X = FLOW_BALANCE

     X = FLOW_BALANCE
 * Constraint on inlet pressure
   G > 0
    G = a*X + b   , with
     X = weighted average(PRESSURE)
     a = 1
     b = -1.5e+006
   Weight: 0

PARAMETERS
   Maximum number of solution evaluations to reach the optimum: 100
   Maximum number of iterations after bracketed the minimum (Line
     Search): 3
   Maximum number of iterations before reset of search directions
     (Fletcher-Reeves): 4
   Initial penalty coefficient: 1.0
   Growing penalty factor: 2.0
   Maximum penalty coefficient: 500000.0
```

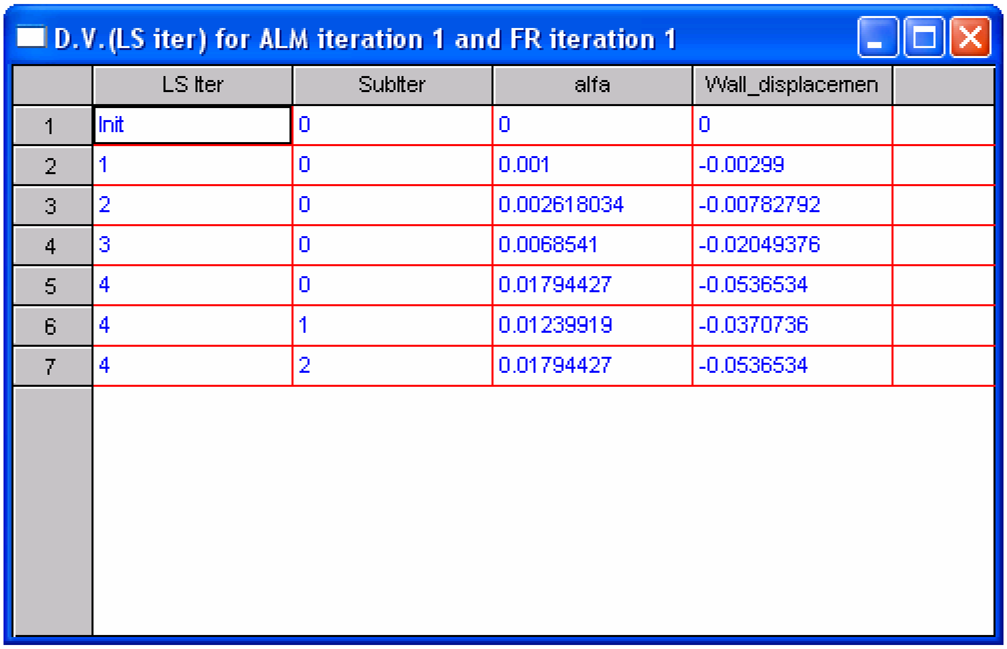
- the **Solvers** button

  Click the **Solvers** button to open the **Information related to solvers** panel, which displays the solver information related to the ALM, FR, and LS iterations and subiterations selected previously. See *Solver Information* (p. 585) for details about the Information related to solvers panel.

29.3.9. Design Variable Information

When you click a **Details** button in the **Optimization information** panel, one of the panels that opens displays a table of information related to the design variables (*Figure 29.15* (p. 595)). The last columns display the values of the design variables. Note that a design variable value that is equal to one of the bounds (lower or upper) is shown in red, whereas intermediate values are shown in blue. The first column displays iteration index numbers of the ALM method, the FR method, or the LS method, depending on the drop-down list with which the **Details** button is associated (e.g., if you click the **Details** button next to the **ALM iteration** drop-down list, the first column displays the ALM iteration index numbers). When the LS iteration numbers are displayed in the first column, the second column displays the LS sub-iteration numbers if relevant, and the third column displays the value of α for the line search method.

Figure 29.15 The Panel Displaying Design Variable Information



D.V.(LS iter) for ALM iteration 1 and FR iteration 1

	LS Iter	SubIter	alfa	Wall_displacemen	
1	Init	0	0	0	
2	1	0	0.001	-0.00299	
3	2	0	0.002618034	-0.00782792	
4	3	0	0.0068541	-0.02049376	
5	4	0	0.01794427	-0.0536534	
6	4	1	0.01239919	-0.0370736	
7	4	2	0.01794427	-0.0536534	

29.3.10. Objective and Cost Function Information

When you click a **Details** button in the **Optimization information** panel, one of the panels that opens displays a table of information related to the objective and cost functions (*Figure 29.16* (p. 596)). The last columns display the following values:

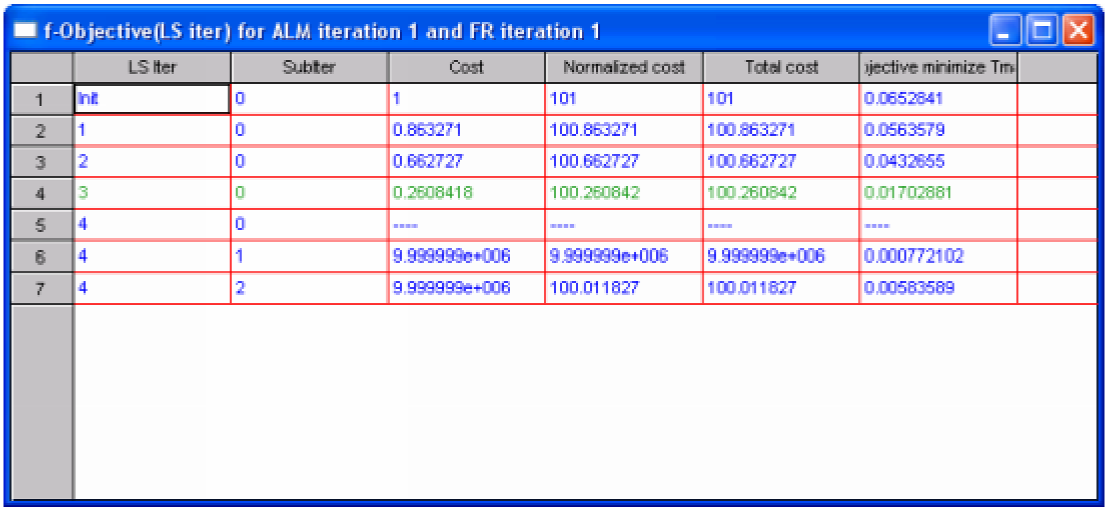
- the cost of the objective function
- the normalized cost
- the total cost, including the constraints
- the values of the objective functions

Note that the values in a row corresponding to the best value of the total cost are shown in green, whereas the other values are shown in blue.

The first column displays iteration index numbers of the ALM method, the FR method, or the LS method, depending on the drop-down list with which the **Details** button is associated (e.g., if you click the **Details** button next to the **ALM iteration** drop-down list, the first column displays the ALM iteration index numbers). When the LS iteration numbers are displayed in the first column, the second column displays the LS sub-iteration numbers if relevant.

Figure 29.16 (p. 596) shows the objective and cost function information for the LS iterations and sub-iterations. The cells displaying " **- - - -**" correspond to failed evaluations.

Figure 29.16 The Panel Displaying Objective and Cost Function Information



f-Objective(LS iter) for ALM iteration 1 and FR iteration 1

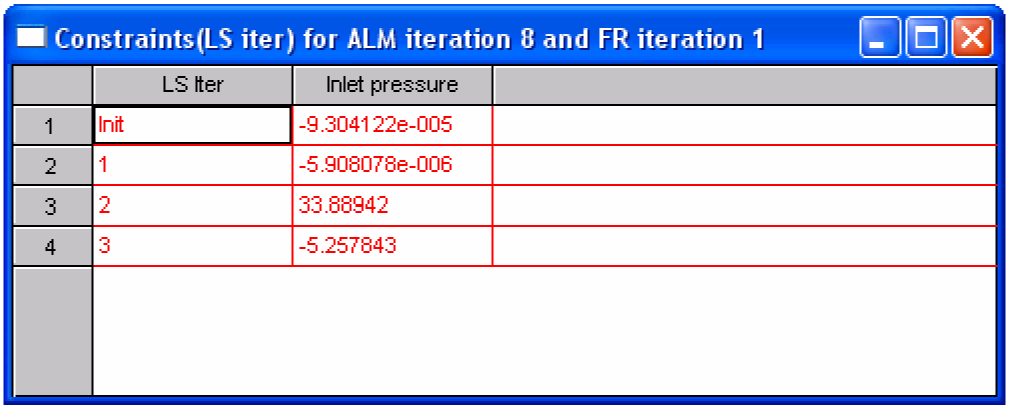
	LS Iter	Subiter	Cost	Normalized cost	Total cost	jective minimize Tm
1	Init	0	1	101	101	0.0652841
2	1	0	0.863271	100.863271	100.863271	0.0563579
3	2	0	0.662727	100.662727	100.662727	0.0432655
4	3	0	0.2608418	100.260842	100.260842	0.01702881
5	4	0	----	----	----	----
6	4	1	9.999999e+006	9.999999e+006	9.999999e+006	0.000772102
7	4	2	9.999999e+006	100.011827	100.011827	0.00583589

29.3.11. Constraint Information

When you click a **Details** button in the **Optimization information** panel, one of the panels that opens displays a table of information related to the constraints (*Figure 29.17* (p. 596)). The last columns display the constraint values. Note that the value of a violated constraint is shown in red, whereas a satisfied constraint is shown in blue. The first column displays iteration index numbers of the ALM method, the FR method, or the LS method, depending on the drop-down list with which the **Details** button is associated (e.g., if you click the **Details** button next to the **ALM iteration** drop-down list, the first column displays the ALM iteration index numbers). When the LS iteration numbers are displayed in the first column, the second column displays the LS sub-iteration numbers if relevant.

Figure 29.17 (p. 596) shows an example of constraint information for LS iterations. If any evaluation had failed in this example, "- - - -" would be displayed in the corresponding cell.

Figure 29.17 The Panel Displaying Constraint Information



Constraints(LS iter) for ALM iteration 8 and FR iteration 1

	LS Iter	Inlet pressure
1	Init	-9.304122e-005
2	1	-5.908078e-006
3	2	33.88942
4	3	-5.257843

29.3.12. Exiting ANSYS POLYDIAG

To exit from ANSYS POLYDIAG, select the **File/Quit** menu item.

File → **Quit**

Chapter 30: User-Defined Functions (UDFs)

If ANSYS POLYFLOW's functions do not meet your needs, you might want to program your own. You can do this with user-defined functions (UDFs).

Information on writing and using user-defined functions in ANSYS POLYFLOW is presented in the following sections:

30.1. Introduction

UDFs can be used to define the value parameters (material parameters, constants appearing in boundary conditions, etc.), or boundary conditions for velocity, temperature, and pressure as functions of other quantities.

Parameters can be functions of the following quantities:

- Fields: quantities that vary from point to point in the domain, such as the velocity, the temperature, or the pressure (PMAT-based functions)
- Additional quantities: local compression rate, local shear rate, local stretch rate, local vorticity, Giesekus function, component of the rate-of-deformation tensor along the velocity (PMAT-based functions)
- Evolution parameter *S* (EVOL-based functions)

Boundary conditions can be functions of the following quantities:

- Fields: quantities that vary from point to point in the domain, such as the velocity, the temperature, or the pressure (PMAT-based functions)
- Evolution parameter *S* (EVOL-based functions)

Note that UDFs for boundary conditions are called only once per evolution step. If you define a velocity UDF that depends, e.g., on temperature, this UDF will be evaluated with the velocity values of the previous step (or 0 for the first step); there is no update of the boundary condition with this UDF during iterations.

30.1.1. Solver Performance with UDFs

UDFs are interpreted by ANSYS POLYFLOW's interpreter, which is part of the CLIPS package, originally developed at NASA. Each time the UDF is called, CLIPS will perform a number of tasks, such as build argument lists, etc. This will generally result in a reduction of ANSYS POLYFLOW's speed.

Also, ANSYS POLYFLOW's building of stiffness matrices requires the computation of numerical derivatives of the equations that are based on UDFs. These tasks can be CPU-intensive, and for this reason ANSYS POLYFLOW has been optimized so that the number of UDF calls per element is minimized. Acceleration tables and approximations of derivatives are used to reduce the number of needed calls.

It is not possible to give an exact figure for the performance penalty of using UDFs, since the performance strongly depends upon the number of arguments, the complexity of the function and the context in which it is used. For EVOL-based UDFs, you can expect only a small performance penalty.

For PMAT-based UDFs, on a 3D mesh with 5000 elements, the overhead of using a CLIPS-based function is on the order of 20% for a UDF with a small number of variables, mostly because of the need to compute numerical derivatives. This percentage can be larger for 2D problems (especially viscoelastic problems where the CPU time required to build finite-element matrices is large) or when a small number of elements is used. Because UDFs for boundary conditions are called once per evolution step, the performance penalty is negligible.

30.2. Writing User-Defined Functions

This section describes how to write UDFs, including information about the type of functions that can be included in the definition of a UDF.

Important

All functions used in a UDF must operate on floating point numbers. All numbers must include a decimal point.

UDFs can be passed up to 10 variables. These passed variables should be named in the first line of your UDF. You will define the correspondence between these variables and ANSYS POLYFLOW variables inside ANSYS POLYDATA, as described in *Using User-Defined Functions* (p. 608).

30.2.1. Naming Your UDF

If your UDF is going to work only with the evolution parameter *S*, then its name should be of the form UDFid, where id is an integer between 1 and 1000. If the UDF is going to work with fields, you can name it anything you want.

30.2.2. Summary of CLIPS Syntax

ANSYS POLYFLOW UDFs are written in the language CLIPS. Examples of UDFs can be found in the Examples Manual or in the file

```
$POLYFLOW/bin/udf.clp
```

where `$POLYFLOW` is the environment variable that indicates where the ANSYS POLYFLOW program has been installed on your computer system. See the installation instructions for more information.

The CLIPS syntax encapsulates all constructions with parentheses.

An external function is defined by the “deffunction" construct. Besides the `deffunction` keyword, a deffunction construct is comprised of three elements:

- a name
- a list of parameters (zero or more)
- a sequence of actions, or expressions, that will be executed in order when the deffunction is called

The return value of a deffunction is the evaluation of the last action.

A deffunction must have a unique name, different from any CLIPS-reserved keyword (otherwise, a CLIPS error will be generated). CLIPS variables and positional parameters start with a ? sign. The syntax is case-sensitive.

A deffunction accepts exactly the required number of arguments as defined in ANSYS POLYDATA. For PMAT UDFs, this will be the number of fields defined in ANSYS POLYDATA. For evolution UDFs, there will be one argument.

30.2.2.1. Example

Here's an example of the CLIPS syntax. Suppose you want to define a UDF that utilizes the polynomial function

$$f(x) = 4 + 3x + 2x^2 + x^3 \tag{30–1}$$

The CLIPS equivalent function, here named UDFP1, with a descriptive comment, is

```
;
; This is a polynomial function with 1 argument
;
(deffunction UDFP1 (?X1)
  (+ 4. (* 3. ?X1) (* 2. ?X1 ?X1) (* 1. ?X1 ?X1 ?X1) ))
```

All expressions are evaluated from left to right, with an order redefined by parentheses. The value returned (the last action) is the sum of four terms, three of them being products of the coefficients and the argument named ?X1.

Lines beginning with ; are comments, and are ignored by the interpreter.

30.2.2.2. Using Variables

To assign a quantity to a variable, CLIPS has the bind command. For instance, to assign the variable X1 the value of 3., use the command

```
(bind ?X1 3.)
```

The bind command can also be used to create new variables, local to your UDF. For instance, (bind ?MyX (+ 8. 9.)) defines a variable MyX local to the UDF, and gives it the value 17.

When a value returned by the function is stored in a bound variable, simply specify the name of that variable as the last executed entry of the function. For example, the earlier polynomial example could be rewritten as

```
;
; This is a polynomial function with 1 argument
;
(deffunction UDFP1 (?X1)
  (bind ?MyX (+ 4. (* 3. ?X1) (* 2. ?X1 ?X1)
  (* 1. ?X1 ?X1 ?X1) ) )
  ?MyX
)
```

Important

The returned value should not be enclosed by parentheses.

30.2.3. Testing Your UDF

A small executable named `qa` is provided for testing UDFs. This program is located in the `$POLYFLOW/bin` directory. When executed, this program will display a `CLIPS>` prompt. Consider that the file containing your CLIPS code is named `myudf.clp`, and that the function name is `UDFP1` as in the previous example. At the CLIPS prompt, enter

```
(batch myudf.clp)
```

to instruct `qa` to read in and assess your UDF. If no errors are found, `qa` will report `TRUE` and print out the UDF. If any errors are found with the UDF, `qa` will print them here.

Assuming no errors were found, you can now test your UDF to see if it returns the values you expect. At the CLIPS prompt, enter

```
(UDFP1 1.1)
```

to evaluate the UDF with `?X1` taking the value of `1.1`. `qa` will print `11.051`, the value returned by `UDFP1`. Note that this is the correct value, since $4 + 3x + 2x^2 + x^3 = 11.051$ when $x = 1.1$.

Here is the transcript of this `qa` session:

```
CLIPS> (batch myudf.clp)
TRUE
CLIPS>
;
; This is a polynomial function with 1 argument
;
(deffunction UDFP1 (?X1)
  (bind ?MyX (+ 4. (* 3. ?X1) (* 2. ?X1 ?X1) (* 1. ?X1 ?X1 ?X1) ))
  ?MyX
)
CLIPS> (UDFP1 1.1)
11.051
CLIPS>
```

30.2.4. Mathematical Functions

CLIPS provides a number of functions for mathematical computation. They are split into two packages: a set of standard functions, and a set of extended functions.

30.2.4.1. Standard Mathematical Functions

The following are the standard mathematical functions. These functions should be used only on numerical arguments. You will see an error message if a string argument is passed to a math function.

- Addition

  The + function returns the sum of its arguments (possibly more than two). Each of its arguments should be a numeric expression.

  The syntax for this function is

  `(+ `*`numeric_expression numeric_expression … numeric_expression`*`)`

  Example:

  ```
  CLIPS> (+ 2. 3.)
  5.
  ```

```
CLIPS> (+ 2. 3. 4.)
9.
```

- Subtraction

  The - function returns the value of the first argument minus the sum of the all subsequent arguments. Each of its arguments should be a numeric expression.

  The syntax for this function is

  (- *numeric_expression numeric_expression … numeric_expression*)

  Example:

  ```
  CLIPS> (- 2. 3.)
  -1.
  CLIPS> (- 12. 3. 4.)
  5.
  ```

- Multiplication

  The * function returns the product of its arguments. Each of its arguments should be a numeric expression.

  The syntax for this function is

  (* *numeric_expression numeric_expression … numeric_expression*)

  Example:

  ```
  CLIPS> (* 2. 3.)
  6.
  CLIPS> (* 2. 3. 4.)
  24.
  ```

- Division

  The / function returns the value of the first argument divided by the product of the subsequent arguments. Each of its arguments should be a numeric expression.

  The syntax for this function is

  (/ *numeric_expression numeric_expression … numeric_expression*)

  Example:

  ```
  CLIPS> (/ 6. 3.)
  2.
  CLIPS> (/ 12. 3. 4.)
  1.
  ```

- Maximum numeric value

  The max function returns the value of its largest numeric argument. Each of its arguments should be a numeric expression.

  The syntax for this function is

  (max *numeric_expression numeric_expression … numeric_expression*)

Example:

```
CLIPS> (max 1. 3. 2. 1.9)
3.
```

- Minimum numeric value

  The `min` function returns the value of its smallest numeric argument.

  The syntax for this function is

  `(min` *`numeric_expression numeric_expression … numeric_expression`*`)`

  Example:

```
CLIPS> (min 2.2 3.5 4.8 2.01)
2.01
```

- Absolute value

  The `abs` function returns the absolute value of its only argument. The argument should be a numeric expression.

  The syntax for this function is

  `(abs` *`numeric_expression`*`)`

  Example:

```
CLIPS> (abs 2.3)
2.3
CLIPS> (abs -4.5)
4.5
```

- Convert to float

  The `float` function converts its only argument to type float and returns this value. The argument should be a numeric expression.

  The syntax for this function is

  `(float` *`numeric_expression`*`)`

  Example:

```
CLIPS> (float 4)
4.0
CLIPS> (float -2.)
-2.0
```

30.2.4.2. Extended Mathematical Functions

In addition to standard mathematical functions, CLIPS also provides a large number of scientific and trigonometric functions for more extensive computations.

- Convert from degrees to grads

The `deg-grad` function converts its only argument from units of degrees to units of grads. That is, it multiplies the argument by a factor of $10/9$. The return value of this function is a float. The argument should be a numeric expression.

The syntax for this function is

`(deg-grad` *`numeric_expression`*`)`

Example:

```
CLIPS> (deg-grad 90)
100.0
```

- Trigonometric functions

  The following functions take a single numeric argument and return a floating-point number. The argument is expected in radians.

Function	Returns
acos	arccosine
acosh	hyperbolic arccosine
acot	arccotangent
acoth	hyperbolic arccotangent
acsc	arccosecant
acsch	hyperbolic arccosecant
asec	arcsecant
asech	hyperbolic arcsecant
asin	arcsine
asinh	hyperbolic arcsine
atan	arctangent
atanh	hyperbolic arctangent
cos	cosine
cosh	hyperbolic cosine
cot	cotangent
coth	hyperbolic cotangent
csc	cosecant
csch	hyperbolic cosecant
sec	secant
sech	hyperbolic secant
sin	sine
sinh	hyperbolic sine
tan	tangent
tanh	hyperbolic tangent

Example:

```
CLIPS> (cos 0)
1.0
CLIPS> (acos 1.)
0.0
```

- Convert from degrees to radians

  The `deg-rad` function converts its only argument from units of degrees to units of radians. That is, it multiplies the argument by a factor of $\pi/180$. The return value of this function is a float. The argument should be a numeric expression.

  The syntax for this function is

  (`deg-rad` *numeric_expression*)

  Example:

  ```
  CLIPS> (deg-rad 180)
  3.141592653589793
  ```

- Convert from grads to degrees

  The `grad-deg` function converts its only argument from units of grads to units of degrees. That is, it multiplies the argument by a factor of $9/10$. The return value of this function is a float. The argument should be a numeric expression.

  The syntax for this function is

  (`grad-deg` *numeric_expression*)

  Example:

  ```
  CLIPS> (grad-deg 100)
  90.0
  ```

- Convert from radians to degrees

  The `rad-deg` function converts its only argument from units of radians to units of degrees. That is, it multiplies the argument by a factor of $180/\pi$. The return value of this function is a float. The argument should be a numeric expression.

  The syntax for this function is

  (`rad-deg` *numeric_expression*)

  Example:

  ```
  CLIPS> (rad-deg 3.141592653589793)
  180.0
  ```

- Return the value of π

  The `pi` function returns the value of π as a float.

  The syntax for this function is

  `(pi)`

Example:

```
CLIPS> (pi)
3.141592653589793
```

- Square root

  The `sqrt` function returns the square root of its only argument as a float. The argument should be a numeric expression.

  The syntax for this function is

  `(sqrt` *`numeric_expression`*`)`

  Example:

  ```
  CLIPS> (sqrt 9)
  3.0
  ```

- Power

  The power function (`**`) returns the value of its first argument raised to the power of its second argument. The arguments should be numeric expressions. The return value is a float.

  The syntax for this function is

  `(**` *`numeric_expression numeric_expression`*`)`

  Example:

  ```
  CLIPS> (** 2 3)
  8.0
  ```

- Exponential function

  The `exp` function evaluates the exponential function e^x (where e is the base of the natural logarithm, $e \approx 2.7182818$) at the value of the function's only argument, and returns the result as a float. The argument should be a numeric expression.

  The syntax for this function is

  `(exp` *`numeric_expression`*`)`

  Example:

  ```
  CLIPS> (exp 2.)
  7.3890560989306502
  ```

- Natural logarithm

  The `log` function evaluates the natural logarithm at the value of the function's only argument and returns the result as a float. The argument should be a numeric expression.

  The syntax for this function is

  `(log` *`numeric_expression`*`)`

  Example:

```
CLIPS> (log 2.718281828459045)
0.9999999999999999
```

- Base 10 logarithm

  The `log10` function evaluates the base 10 logarithm at the value of the function’s only argument and returns the result as a float. The argument should be a numeric expression.

  The syntax for this function is

  (`log10` *`numeric_expression`*)

  Example:

  ```
  CLIPS> (log10 100)
  2.0
  ```

30.2.5. Procedural Functions

The following are functions that provide procedural programming capabilities as found in languages such as FORTRAN and C.

Important

In the descriptions below, items enclosed in square brackets `[ ]` are optional. If you choose to include the optional information, do not include the square brackets, only the items within them.

- If...then...else function

  CLIPS provides an `if...then...else` structure to allow for conditional execution of a set of actions.

  The syntax is

  ```
  (if <expression>
    then
       <action>
    [else
       <action>]
  )
  ```

  Any number of allowable actions can be used inside of the `then` and `else` portions, including other `if...then...else` structures. The `else` portion is optional. If `<expression>` evaluates to anything other than the symbol FALSE, then the actions immediately following the `then` are executed.

  Otherwise, the actions following the `else` are executed. Although the `if` function returns a value, it is not useful in ANSYS POLYFLOW.

  Example:

  This function sets the local variable `?Val` to 3. if the variable `?v` is equal to 6, and sets `?Val` to 1. otherwise. In addition to =, you can use the comparison operators <, >, <=, and >=.

  ```
  (if (= ?v 6)
    then
  ```

```
      (bind ?Val 3.)
   else
      (bind ?Val 1.)
  )
```

- While

  The `while` function is provided to allow simple looping. Its use is similar to that of the `if` function.

  The syntax is

  ```
  (while <expression>[do]
    <action>)
  ```

  Any number of functions can be included in the `<action>`. The symbol `do` can appear after the `<expression>` but it has no effect on the function.

  If `<expression>` is evaluated to FALSE, then the function stops and does nothing. If `<expression>` is not FALSE, then `<action>` is executed. Then `<expression>` is reevaluated: if FALSE, the function stops; if not FALSE, `<action>` is executed again, and the process continues until `<expression>` evaluates to FALSE.

  Example:

  ```
  (while (< ?v 10)
     (bind ?v (- (* ?v ?v) 10))
  )
  ```

  This function take the value of `v` and, if it less than 10, repeatedly replaces it with the value of v^2`-10` until `v` is greater than 10.

- Loop-for-count

  The `loop-for-count` function is provided to allow simple iterative looping.

  The syntax for this function is

  ```
   (loop-for-count <range-spec>[do]<action>)
  ```

  where `<range-spec>` has the syntax

  ```
  <end-index> | (<loop-variable>[<start-index><end-index>])
  ```

  where `<start-index>` and `<end-index>` are integer expressions.

  Any number of functions and expressions can be included in the `<action>`.

  Example 1:

  ```
   (loop-for-count 5 (bind ?X1 (sin ?X1)))
  ```

  This example replaces `?X1` by the value of $\sin(\sin(\sin(\sin(\sin(?\mathrm{X1})))))$.

  Example 2:

  ```
  (deffunction PowerSeries4 (?x)
    (bind ?y 0)
    (loop-for-count (?Count 0 4) do
      (bind ?y (+ ?y (** ?x ?Count)))
    )
    ?y
  )
  ```

This example takes a number x and returns $1 + x + x^2 + x^3 + x^4$

30.3. Using User-Defined Functions

The steps for implementing a UDF in ANSYS POLYFLOW are provided here. Note that you should also perform all other steps needed to correctly define your model before running ANSYS POLYFLOW.

1. Write your UDF. See *Writing User-Defined Functions* (p. 598) for information on writing UDFs.
2. Declare the dependency in ANSYS POLYDATA.

    a. In ANSYS POLYDATA, select the parameter or boundary condition you want to have described by your UDF. For instance, if you want density to be related to temperature in a way described by your UDF, select **Density** in the **Material data** menu. If you want to impose the normal velocity as a function of coordinates, select **Normal and tangential velocities imposed** or **Normal velocity and tangential force imposed** in the **Flow boundary conditions** menu.

    b. For parameters, if your UDF is a function of various fields, click the **PMAT** button at the top of the menu.

       For boundary conditions, if your UDF is a function of various fields, select the **User Defined Function** option in the menu where **Constant** and **Linear function of coordinates** are located.

       For both parameters and boundary conditions, if your UDF is a function of the evolution parameter S, click **EVOL**.

    c. Define this parameter to be a constant times the value given by your UDF. Select the appropriate menu item to modify the parameter value (e.g., **Modification of density**) and enter the value of this constant.

    d. If you are using the **PMAT** option:

       i. In the **Functional Dependence of...** menu, select **User Defined Function**

          User Defined Function

          and then **Create a new function**.

          Create a new function

          You will see a function **f1(...)** listed in the menu.

       ii. Select **f1(...)** and enter the appropriate information in the **User Defined Function** menu.

           A. Enter the name of your UDF.

              Modify function name

           B. Enter the number of fields that your UDF uses.

              Modify nb. of fields

              Initially, all fields (listed **X1**, **X2**,...) are defined to be the **X-coordinate** value.

           C. Redefine each field by selecting it and then choosing the appropriate new field from the list of choices. Information about the choices is provided at the end of this section.

If you are using a UDF for a boundary condition:

   i. In the menu for the appropriate component of the field (e.g., normal velocity), select **User Defined Function**.

      User Defined Function

   ii. In the resulting **User Defined Function** menu, specify the name of your UDF.

      Modify function name

   iii. Next, specify the number of fields that your UDF uses.

      Modify nb. of fields

      The menu will expand to show an item for each field. Initially, all fields (**X1**, **X2**, etc.) are defined to be the x-coordinate value.

   iv. Specify each field by selecting it and choosing the appropriate new field from the list of choices.

If you are using the **EVOL** option:

   i. In the **S-Dependence of ...** menu, select **f(S) = User Defined Function**.

      f(S) = User Defined Function

   ii. When prompted, enter the **UDFid**. This is an integer between 1 and 1000 that should correspond to the name of your UDF. That is, your UDF's name should be `UDF`*`id`* where *`id`* is this integer.

Important

Note that at this stage, ANSYS POLYDATA is unable to verify that this function has indeed been defined and that the number of arguments is correct; those errors, if they occur, will be reported during the execution of ANSYS POLYFLOW.

3. Execute ANSYS POLYFLOW with the `-f` option:

   ```
   POLYFLOW -f udf-file < data-file
   ```

   where *`udf-file`* is the file containing your UDF and *`data-file`* is the data file containing the problem definition.

   If a CLIPS error is detected in your function definition, CLIPS will report this at the start of the ANSYS POLYFLOW execution. If the function name used in ANSYS POLYDATA cannot be found in your UDF file, or if the number of arguments is incorrect, an error will be reported in the `SOLVER` section of the listing file (described in *Writing an ANSYS POLYFLOW Listing File* (p. 110)).

30.4. Dependence with Respect to Quantities Derived from the Kinematics

30.4.1. Some Quantities Derived from the Kinematics

When you select a field to be redefined, you will see several fields listed in the resulting menu, including the coordinate components, velocity components, pressure, and temperature. Six additional quantities, local compression rate, local shear rate, local stretch rate, local vorticity, Giesekus function, and D_{vv} (component of the rate-of-deformation tensor along the velocity) are also available. The first three are functions of invariants of the rate of deformation tensor $\mathbf{D}$, while the last three are actually no longer invariant, but can be useful in some circumstances.

The three invariants are defined as follows:

$$\begin{aligned} \mathrm{I}_{\mathbf{D}} &= \mathrm{tr}\,(\mathbf{D}) \\ \mathrm{II}_{\mathbf{D}} &= \mathrm{tr}\left(\mathbf{D}^2\right) \\ \mathrm{III}_{\mathbf{D}} &= \det\,(\mathbf{D}) \end{aligned} \qquad \textbf{(30–2)}$$

The compression rate is defined as the trace of the rate-of-deformation tensor, $\mathrm{I}_{\mathbf{D}}$, given by *Equation 30–2* (p. 610). For an incompressible fluid, this first invariant vanishes.

The local shear rate $\dot{\gamma}$ is based on the second invariant of the rate-of-deformation tensor:

$$\dot{\gamma} = \sqrt{2\mathrm{II}_{\mathbf{D}}} \qquad \textbf{(30–3)}$$

For a pure shear flow, this quantity provides the actual shear rate.

The local stretch rate $\dot{\varepsilon}$ is based on the ratio of the third invariant to the second invariant [*8*] (p. 715):

$$\dot{\varepsilon} = 6\frac{\mathrm{III}_{\mathbf{D}}}{\mathrm{II}_{\mathbf{D}}} \qquad \textbf{(30–4)}$$

For a pure uniaxial elongational flow, this quantity provides the actual stretch rate.

Important

It is important to remember that the quantity defined by *Equation 30–4* (p. 610) vanishes for a planar flow.

The magnitude $\dot{\omega}$ of the vorticity tensor $\mathbf{\Omega}$ (the antisymmetric part of the velocity gradient) is given by:

$$\dot{\omega} = \sqrt{\;|2\mathrm{tr}\left(\mathbf{\Omega}^2\right)|} \qquad \textbf{(30–5)}$$

where the absolute value is considered in order to avoid questions about the definition and the sign of $\mathbf{\Omega}^2$. For a rigid rotation motion, this quantity provides the actual rotation velocity.

The Giesekus function G is defined as:

$$G = \frac{\mathrm{II}_{\mathbf{D}} - |\mathrm{II}_{\mathbf{\Omega}}|}{\mathrm{II}_{\mathbf{D}} + |\mathrm{II}_{\mathbf{\Omega}}|} \tag{30–6}$$

and ranges between -1 (rigid rotation) and 1 (pure extension). It equals 0 for a pure shear flow.

The component of the rate of deformation tensor $\mathbf{D}$ along the velocity field $\mathbf{v}$ is defined as:

$$D_{vv} = \frac{\mathbf{v} \cdot \mathbf{D} \cdot \mathbf{v}}{\mathbf{v} \cdot \mathbf{v}} = \hat{\mathbf{v}} \cdot \mathbf{D} \cdot \hat{\mathbf{v}} \tag{30–7}$$

where $\hat{\mathbf{v}}$ is the unit vector along $\mathbf{v}$. The quantity D_{vv} vanishes in a pure shear flow, while it provides the stretch rate under elongation. It is interesting to note that this quantity also provides nonvanishing values for a planar flow.

30.4.2. Possible Viscosity Models with Distinct Behaviors in Shear and Extension

With the above quantities, it is possible to build a generalized Newtonian fluid model whose viscosity exhibits distinct behaviors in shear and extension. Here, one can, e.g., combine the quantities given by equations *Equation 30–3* (p. 610) and *Equation 30–4* (p. 610) with appropriate algebra. A fluid model exhibiting a constant shear viscosity and strain hardening is found in [*8*] (p. 715) and is given by:

$$\eta\ (\dot{\varepsilon})\ = \eta_0 \cosh\ (m\lambda\, \dot{\varepsilon}) \tag{30–8}$$

where the quantity η_0 is the zero-shear viscosity, while the parameters m and λ control the strain hardening. Its translation in CLIPS language for usage as a UDF involving the local stretch rate (as single dependence field) is typically given in the following example:

```
;
; Viscosity law referred to as 32.
;
; Follow the instructions; do not alter anything else
; (note that a real number MUST contain a "."; "2." differs from "2")
;
(deffunction viscosity (?e)
;
;          Specify here the zero-shear viscosity
    (bind ?eta0 1000.)
;
;          Specify here the strain hardening factor (positive)
    (bind ?m 2.)
;
;          Specify here the time constant for cosh function
    (bind ?lambda 3.)
    (bind ?R (* ?eta0 (cosh(* ?m ?lambda ?e)) ) )
?R
)
```

A fluid model exhibiting a shear thinning and strain hardening could be given by:

$$\eta\ (\dot{\gamma}, \dot{\varepsilon})\ = \eta_{\infty} + \left(\eta_{0} - \eta_{\infty}\right) \frac{(1 +\ (\lambda \dot{\gamma})^{\ 2})^{\frac{n-1}{2}} \cosh\ (m\lambda\, \dot{\varepsilon})}{(1 + 3\ (\lambda\, \dot{\varepsilon})^{\ 2})^{\frac{n-1}{2}}} \qquad \textbf{(30–9)}$$

where $\dot{\gamma}$ and $\dot{\varepsilon}$ respectively denote the local shear rate and local stretch rate. The quantities η_0, η_∞, λ, m, and n are the independent parameters of the law. The translation of Equation 30–9 (p. 612) in CLIPS language for usage as a UDF involving the local shear rate and local stretch rate is typically given by the following example:

```
;
; Viscosity law referred to as 31.
;
; Follow the instructions; do not alter anything else
; (note that a real number MUST contain a "."; "2." differs from "2")
;
(deffunction viscosity (?g ?e)
;
;          Specify here the zero-shear viscosity
    (bind ?eta0 1000.)
;
;          Specify here the viscosity at infinite shear
    (bind ?etainf 0.)
;
;          Specify here the strain hardening factor (positive)
    (bind ?m 0.4)
;
;          Specify here the time constant for Bird-Carreau
    (bind ?lambda 3.)
;
;          Specify here the power index for Bird-Carreau viscosity
    (bind ?n 0.7)
    (bind ?R (+ ?etainf (* (- ?eta0 ?etainf) (/ (* (** (+ 1. (**
    (* ?lambda ?g) 2.)) (/ (- ?n 1.) 2.) ) (cosh(* ?m ?lambda ?e)))
    (** (+ 1. (* 3. (** (* ?lambda ?e) 2.))) (/ (- ?n 1.) 2.))) )))
?R
)
```

When invoking the UDF function with two arguments (the local shear rate and local stretch rate) for such a law, it is important to select them in that sequence. It is also important to note that the last bind statement in the UDF file is actually written on one single line that terminates in this example with the six closing parentheses.

Eventually, a fluid model exhibiting a shear thinning and a constant elongation viscosity could be given by:

$$\eta = \eta_{\infty} + \left(\eta_{0} - \eta_{\infty}\right) \left(\frac{1 +\ (\lambda\, \dot{\gamma})^{\ 2}}{1 + 3\ (\lambda\, \dot{\varepsilon})^{\ 2}}\right)^{\frac{n-1}{2}} \qquad \textbf{(30–10)}$$

where $\dot{\gamma}$ and $\dot{\varepsilon}$, respectively denote the local shear rate and local stretch rate. The quantities η_0, η_∞, λ, ? and n are the independent parameters of the law. The translation of Equation 30–10 (p. 612) into CLIPS language for usage as a UDF involving the local shear rate and local stretch rate is typically given by the following example:

```
;
; Viscosity law referred to as 33.
;
; Follow the instructions; do not alter anything else
; (note that a real number MUST contain a "."; "2." differs from "2")
;
(deffunction viscosity (?g ?e)
;
;           Specify here the zero-shear viscosity
    (bind ?eta0 1000.)
;
;           Specify here the viscosity at infinite shear
    (bind ?etainf 0.)
;
;           Specify here the time constant for Bird-Carreau
    (bind ?lambda 3.)
;
;           Specify here the power index for Bird-Carreau viscosity
    (bind ?n .7)
    (bind ?R (+ ?etainf (* (- ?eta0 ?etainf) (** (/ (+ 1. (**
    (* ?lambda ?g) 2.)) (+ 1. (* 3. (** (* ?lambda ?e) 2.)) ) )
    (/ (- ?n 1.) 2.) ) )))
?R
)
```

When invoking the UDF function with two arguments (the local shear rate and local stretch rate) for such a law, it is important to select them in that sequence. It is also important to note that the last bind statement in the UDF file is actually written on one single line that terminates here with the five closing parentheses.

It is important to remember that these laws are analytically more complex than those involving the local shear rate only, and that this may have consequences on the numerical treatment.

Other laws can be created, but they are mainly a matter of algebra.

Chapter 31: User-Defined Templates (UDTs)

When using a CFD-related software tool in a team environment, it is advisable to have more experienced users designing models and less experienced users running simulations. The user-defined template, or UDT, is a feature that allows you to create a baseline case that can be easily modified using a simplified ANSYS POLYDATA interface. This capability provides access to preselected parameters only, and saves time when you need to run a series of similar simulations that vary by only a few parameters.

This chapter contains the following sections:

31.1. Introduction

The data file that contains a user-defined template (UDT) is similar to a data file produced during a normal ANSYS POLYDATA session. The only difference is the identification of parameters. All parameters that you want to be able to access in the simplified ANSYS POLYDATA interface must be identified (or flagged) in the UDT data file. The process of flagging a parameter is similar to defining a parameter as a function of time or evolution variable, in a typical ANSYS POLYDATA session.

31.2. Defining a UDT

Defining a UDT is a two-step process. It involves first defining a UDT the way you would define a normal simulation, and then flagging the parameter(s) you want to access in an ANSYS POLYDATA session. The basic steps for flagging parameters in a UDT are as follows:

1. Create a template entry for each parameter you want to flag.
2. Review a template entry.
3. Modify a template entry.

31.2.1. Creating a New Template Entry to Flag a Parameter

You will need to create a new template entry in order to flag a parameter. To do this:

1. Select the appropriate menu for the parameter you want to flag. For example, if the parameter is density, then select the **Density** menu in the **Material data** panel.
2. Click the **UPDT** button at the top of the ANSYS POLYDATA menu to enable template inputs.
3. Select the appropriate parameter menu item (e.g., **Modification of density**).
4. Enter the new value for the parameter when prompted by ANSYS POLYDATA.
5. Click **OK** to accept the new value. This will open the **Create template entry** menu.
6. Click **Create a new template entry** to flag the parameter.
7. Select **Modify comment**, enter a reference name for your new template entry, and click **OK**.

Note that you will refer to the flagged parameter by this name when managing template entries.

8. Select **Upper level menu**.
9. Click **UPDT** to disable the template inputs.

Important

Note that you can define a template entry for all parameters for which the **EVOL** or **PMAT** buttons are available (i.e., the parameters of material properties, the amplitudes of a mesh deformation preprocessor, and the parameters of the boundary conditions).

31.2.2. Reviewing a Template Entry

To review template entries, click the **LSDT** button.

This will open the **UPDT summary** menu which will list all the template entries currently defined. Parameters are referenced by the comment entered during template creation (see *Creating a New Template Entry to Flag a Parameter* (p. 615)).

Note that the **LSDT** button is similar to the **LSEV** button. Where the **LSEV** button lists all parameters defined as a function of time or evolution variable, the **LSDT** button lists all the template entries. Choose one item in the list to modify the corresponding parameter, without going into the specific menu containing this parameter.

31.2.3. Modifying a Template Entry

To modify a template entry:

1. Click the **LSDT** button. This will open the **UPDT summary** menu.
2. Select the entry corresponding to the parameter you want to modify.

   This will open a menu similar to the one you used to create the entry template.
3. Select the required operation to delete the template entry, modify the comment, or modify the value of the parameter.
4. Click **Upper level menu**.
5. Select **Save & exit** to apply the changes and exit.

31.3. Using UDTs

You can use a UDT in three different ways. These methods are described in subsequent sections. In summary, a UDT can be used to:

- List and modify all parameters using the **LSDT** button in ANSYS POLYDATA. This is the simplest usage of a UDT, and is referred to as the “As Usual" method (see *As Usual Method* (p. 617)).
- Start a simplified ANSYS POLYDATA interface and modify only the flagged parameters using the ANSYS POLYDATA `-TPL` command line option. This is referred to as the “Real UDT" method (see *Real UDT Method* (p. 617)). Note that ANSYS POLYMAN will always launch ANSYS POLYDATA in this mode when the data file contains flagged parameters.
- Set the new value of the parameter(s) in the `templates.upd` file, which is created by ANSYS POLYDATA during the creation of a UDT, and execute ANSYS POLYDATA in scripts using the ANSYS POLYDATA

`-TPL -UPDT templates.upd` command line option. This is referred to as the “Script" method (see *Script Method* (p. 617)). Note that ANSYS POLYDATA updates its database on the basis of the contents in `templates.upd`.

31.3.1. As Usual Method

The **As Usual** method is the simplest UDT method and allows you to modify any detail of the problem setup.

1. Launch ANSYS POLYDATA as usual.
2. Read the mesh file (not required if running from ANSYS POLYMAN).
3. Read the old data file (not required if running from ANSYS POLYMAN).
4. Click the **LSDT** button. This will open a **UPDT summary** menu.
5. Select the entry corresponding to the parameter you want to modify.

   This will open a menu similar to the one you used to create the entry template.
6. Select the required operation to delete the template entry, modify the associated comment, or modify the value of the parameter.
7. Click **Upper level menu**.
8. Select **Save & exit** to apply changes and exit.

31.3.2. Real UDT Method

The **Real UDT** method allows you to modify only the parameters flagged during the definition of a UDT. This method allows a more experienced user to set up a case and a less experienced user to use the UDT to explore solutions around the baseline.

1. Launch ANSYS POLYDATA with the `-TPL` command line option or double click the mesh or data file in ANSYS POLYMAN.
2. Read the mesh file (not required if running from ANSYS POLYMAN).
3. Read the old data file (not required if running from ANSYS POLYMAN).
4. Select **Modify template parameters**. This will open a **UPDT summary** menu.
5. Select the entry corresponding to the parameter you want to modify.

   This will open a menu similar to the one you used to create the entry template.
6. Select the required operation to delete the template entry, modify the associated comment, or modify the value of the parameter.
7. Click **Upper level menu**.
8. Select **Save & exit** to apply changes and exit.

31.3.3. Script Method

The **Script** method is used when you call ANSYS POLYDATA within a script. It controls the value of the flagged parameters via a template file (`*.upd`) which is automatically created by ANSYS POLYDATA during the creation of the UDT. Below is an example of a template file.

```
DV #1
 expo
0.4000000E+00
```

```
 DV #2
 fac
0.4000000E+00
 DV #3
 facinf
0.4000000E+00
 DV #4
 tnat
0.4000000E+00
```

The sample file corresponds to a UDT in which the four parameters of the Bird-Carreau law were flagged. This can be easily used to modify the value of any parameter of the viscosity model.

The name of each parameter appearing in the template file (e.g., `expo`) corresponds to the comment that is entered in the **Create template entry** process during the creation of the UDT (see *Creating a New Template Entry to Flag a Parameter* (p. 615)).

Now, suppose you want to run a simulation with a value of the exponent of the power law (e.g., `expo`) set to.35 instead of 0.40. To do this, edit the template file and change the corresponding value. An example of an edited template file is shown below.

```
 DV #1
expo
 0.3500000E+00
DV #2
fac
 0.4000000E+00
DV #3
facinf
 0.4000000E+00
DV #4
tnat
 0.4000000E+00
```

To directly update the value of the parameter, launch ANSYS POLYDATA with the `-UPDT` command option following these steps:

1. Launch ANSYS POLYDATA with the `-UPDT templates.upd` command line option.
2. Read the mesh file.
3. Read the old data file.
4. Select **Save & exit**.

Chapter 32: Die Shape Parameterization

ANSYS POLYFLOW's mesh deformation capabilities allow you to have meshes in your simulation that deform as a result of the flow solution. This capability is used in problems involving free surfaces (e.g., an extrusion) and problems involving fluid-structure interactions (e.g., a die lip that widens as a result of the flow). This chapter, on the other hand, describes how these capabilities have been extended to allow you to deform the geometry (and the associated mesh) before any flow simulations are run. In other words, you can alter the position and shape of geometric entities (volumes, faces, lines, and points) as part of a preprocessor task. The new mesh in the deformed geometry will have the same number of nodes and will be obtained prior to and independent of any flow calculations. Thus, the deformation applied to your model is strictly the result of user input and is not related to the flow.

The advantages of altering the mesh in a preprocessor task include the following:

- There is no need to revisit the mesh generator to perform "slight" modifications to the geometry of your initial mesh.
- Depending on how the mesh deformation is defined, it may be possible to apply evolution or user-defined template (UDT) options on the geometric parameters. If evolution is applied on the deformation parameters, a series of meshes will be generated from the initial mesh based on a specific deformation criteria. By applying UDT options, on the other hand, you can save a much smaller version of the data file, which then only allows you to change specified geometric parameters as part of future parametric studies.
- Because the mesh deformation is computed using ANSYS POLYFLOW, it is possible to analytically compute the sensitivities of the flow variables with respect to the parameters controlling the mesh deformation.
- The mesh generation can be simplified, because the points controlling the mesh deformation can be a node of the mesh and not only a point defined in the mesh generator.

The main applications of this capability include the following:

- parametric studies

  By prescribing the deformation of the mesh, you can perform a parametric study to evaluate the impact of geometric modifications on different aspects of the flow (e.g., pressure drop, stagnation zones, maximum shear rate). This application is described in this chapter.
- optimization studies

  You can perform an optimization study by allowing the optimizer to find better geometries than the one you originally provided, in order to improve some aspects of the flow (e.g., better flow balance at the die exit, reduced pressure drop). This application is described in *Optimization* (p. 651).

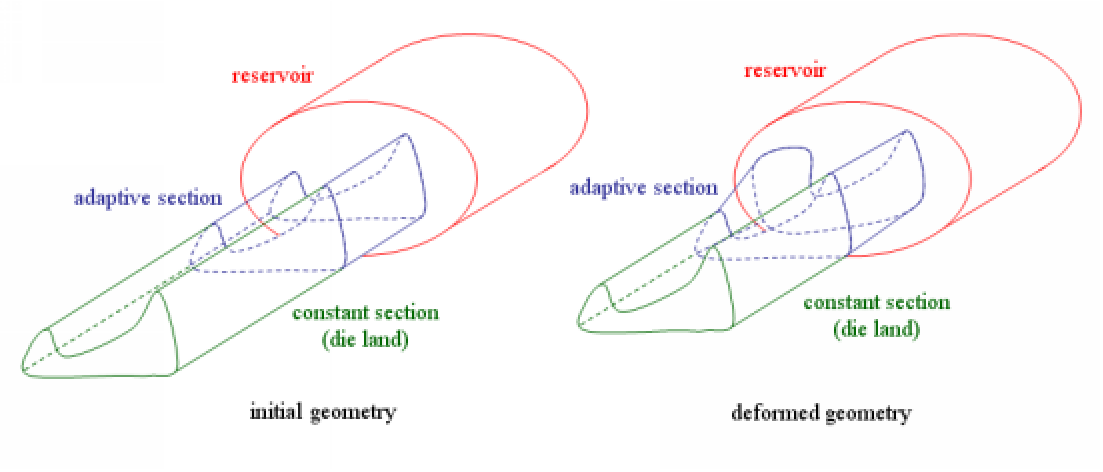
This chapter provides information about using geometry parameterization to modify the die shape, and contains the following sections:

32.1. Introduction

For many extrusion simulations, the goal is to predict the die shape needed to produce a desired extrudate shape. This can be accomplished using the inverse extrusion approach (described in *Inverse Extrusion and Die Design* (p. 331)), as long as the flow at the die exit is well balanced. When it is not well balanced (i.e., has large velocity variations), it is necessary to modify the shape of the die.

This chapter focuses on mesh parameterization for dies with a simple shape. *Figure 32.1* (p. 620) shows the three main parts of such dies: the constant section (die land), the adaptive section, and the reservoir. This figure also provides an example of the difference between the initial geometry and the deformed geometry that balances the flow in the die.

Figure 32.1 The Main Sections of a Die



In order to balance the die, you can make the following modifications:

- modify the shape of the inlet for the adaptive section

  You can “open" (enlarge) the zones where the velocity is low and thereby bring more fluid in this zone. You can also “close" (reduce) the zones where the velocity is large.
- modify the length of the adaptive section

  If the length is too short, stagnation zones may result. If it is too long, it is less efficient.
- modify the length of the constant section

  If the constant section is too long, the effect of adaptive section deformations will be reduced, because the fluid will have a “long" distance to reorganize itself.

In order to make such modifications, you need to prescribe the deformation of volumes, surfaces, lines, and points.

32.2. Theory

The following section describes the equations governing the different methods of deformation or motion that can be applied on the geometric entities of a mesh (i.e., volumes, surfaces, lines, and points). All these methods compute new coordinates ($\mathbf{X}$) by modifying the initial coordinates ($\mathbf{X}_{ini}$) that are read from the mesh files. The new coordinates will be used by the flow problem after the mesh deformation has been applied.

Mesh deformation is mainly governed by four equations:

- The first equation sets the new coordinates of a node to be equal to the original coordinates, so that the node does not move:

  $$\mathbf{X} = \mathbf{X}_{ini} \tag{32–1}$$

- The second equation defines a rigid translation, which is characterized by a direction (D) and an amplitude (a). The new coordinates $\mathbf{X}$ are given by:

  $$\mathbf{X} = \mathbf{X}_{ini} + a\mathbf{D} \tag{32–2}$$

  During a rigid translation, all nodes experience the same displacement. An evolution function can be used to define the amplitude, and hence a series of meshes may be generated at different magnitudes of translation. Note that an evolution function cannot be used to define the direction.

- The third equation defines an elastic remeshing, governed by the pseudo-elastic equation for the new coordinates $\mathbf{X}$:

  $$\nabla \cdot \left(E \left(\nabla \mathbf{X} + \nabla^{T} \mathbf{X} \right) \right) = 0 \tag{32–3}$$

  The previous equation represents an elastic material analogy, where E functions in the role of the Young's modulus and the Poisson's ratio is ignored (i.e., set equal to 0). The reference configuration is related to the initial values of the coordinates ($\mathbf{X}_{ini}$). ANSYS POLYFLOW allows you to specify how E is calculated; by default, the values that make up this field are functions of the quality of the individual mesh elements, in order to control mesh distortion during the remeshing (see *Element Stiffness During Elastic Remeshing* (p. 622) for further details). Note that in terms of defining the remeshing, it is not the absolute value of E for an element that is critical, but rather its value relative to the values of the other elements.

  Equation 32–3 (p. 621) requires boundary conditions that prescribe the motion of the domain boundaries. In order to have a well-posed problem, the boundary conditions must be such that the whole domain does not undergo rigid motion (i.e., translation and rotation).

- The fourth equation defines a constraint on the normal displacement. This equation imposes a vanishing normal displacement:

  $$(\mathbf{X} - \mathbf{X}_{ini}) \cdot \mathbf{n} = 0 \tag{32–4}$$

  where $\mathbf{n}$ is the vector normal to the surface.

By combining *Equation 32–1* (p. 621) – *Equation 32–4* (p. 621) for volumes, surfaces, lines, and points, you can obtain a large set of mesh transformations. For the sake of simplicity, the methods are organized by the geometric dimension on which they are applied, and they are combined to result in predefined motions. In order to have greater flexibility, there are no predefined boundary conditions. The boundary conditions are imposed by defining the motion of the geometry of lower dimension. Thus, a volume may have boundary conditions defined on its surfaces, which may have boundary conditions defined on its lines, which may have boundary conditions defined on its points.

For this reason, the methods are described on the basis of the geometric dimension of the domain. *Domain Transformations* (p. 625) is related to the domain transformation, including 2D and 3D domains. *Surface Transformations* (p. 626) is related to the surface transformations used to impose the boundary conditions on a 3D domain transformation. *Line Transformations* (p. 629) concerns the line transformations used to define the boundary conditions for either a 2D domain transformation or a surface transformation. Finally, *Point Displacement* (p. 634) describes the motion of points that set the boundary conditions for line transformations.

With this approach, a surface, line, or point may have more than one equation applied to it. *Hierarchy of Equations* (p. 634) describes how multiple equations on a single entity are addressed.

32.2.1. Element Stiffness During Elastic Remeshing

Whenever the boundary of a mesh undergoes significant displacement (e.g., due to the deformation of the fluid region during parameterization or optimization of the geometry), the mesh must be adjusted to accommodate the new configuration. When adjustments are made using elastic remeshing, ANSYS POLYFLOW treats the mesh as a pseudo-elastostatic medium and employs the same general algorithm that is used to calculate the displacement of an elastic solid. The mesh boundary conditions differ from those for the solid, however, in that only displacement boundary conditions are allowed, because boundary conditions that involve stresses or forces proportional to displacement have no meaning in the context of the mesh.

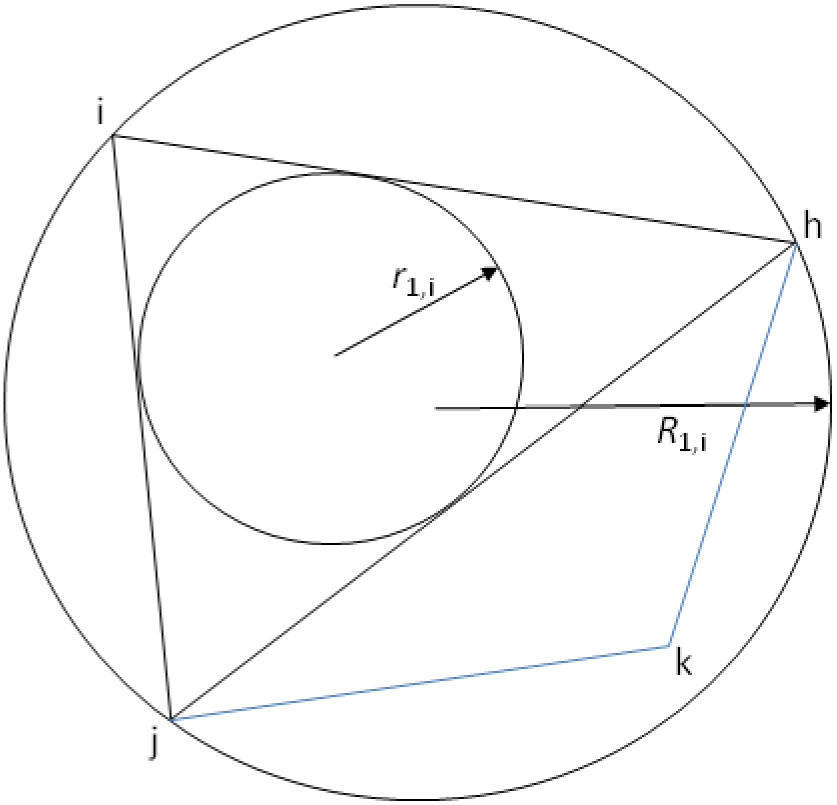
A potential drawback of modeling the mesh as an elastostatic medium is that the solver does not guarantee the production of a mesh in which element distortion is absent. In most practical applications, the mesh is not uniform and contains elements that are skewed, have high aspect ratios, or otherwise have bad quality. A straightforward treatment of the mesh as an isotropic homogeneous pseudo-elastic medium tends to distort all of elements in the same way, including the “bad" ones. In order to control mesh distortion due to remeshing, it is preferable to manipulate the pseudo-material elasticity matrix in such a way as to increase the stiffness of bad elements or elements whose quality is worsening. The major advantage of such an approach is that it can be done efficiently and is physically based.

In order to understand how the ANSYS POLYFLOW elastic remeshing algorithm can control mesh distortion, it is necessary to examine how the mesh quality and new nodal positions are calculated. Consider an example where element 1 is made up of nodes h, i, j, and k. A quality indicator ($Q_{1,i}$) for node i of element 1 is calculated using the following equation:

$$Q_{1,i} = \frac{R_{1,i}}{r_{1,i}} \tag{32–5}$$

In the previous equation, $R_{1,i}$ and $r_{1,i}$ are radii of the circumcircle and the incircle, respectively, of a triangle whose points include node i and the two nodes attached to i by a single face in element 1 (i.e., nodes h and j), as shown in *Figure 32.2* (p. 623).

Figure 32.2 The Circumcircle and Incircle for Node i on Element 1



Note that the minimum value $Q_{1,i}$ can be is 2, which would be achieved if triangle h-i-j was equilateral. *Equation 32–5* (p. 622) gives a large value for very thin elements (even if the element is a rectangle), or for elements that have an angle close to 0 or 180°. *Table 32.1: $Q_{1,i}$ as a Function of Height (h)* (p. 624) shows how the value of $Q_{1,i}$ changes with height (h), if triangle h-i-j were one of the triangles illustrated in *Figure 32.3* (p. 624).

Figure 32.3 Height h for Two Kinds of Triangles

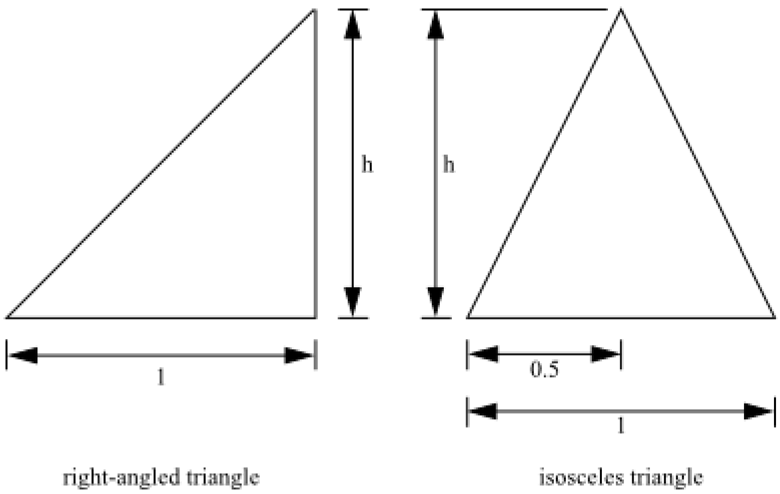


Table 32.1 $Q_{1,i}$ as a Function of Height (h)

h	$Q_{1,i}$	
	Right-Angled Triangle	**Isosceles Triangle**
100	100.5	100.5
10	10.6	10.5
5	5.7	5.6
2	2.9	2.7
1	2.4	2.0
0.5	2.9	2.4
0.2	5.7	7.5
0.1	10.6	26.3
0.01	100.5	2,501.3
0.001	1,000.5	250,001.3

While *Figure 32.2* (p. 623) only shows how $R_{1,i}$ and $r_{1,i}$ are defined for a 2D element, these variables can be also be defined for a 3D element. For a 3D element, $R_{1,i}$ and $r_{1,i}$ are radii of the circumsphere and insphere, respectively, of a tetrahedron, whose points include a given node and the three nodes of an element that are attached to it by a single edge.

After the quality indicator is calculated at node i for element 1, the same process is repeated for all of the other elements that share node i. A total quality indicator for node i (Q_i) is then calculated by summing all of these values:

$$Q_i = \sum Q_{e,i} \tag{32–6}$$

where e represents the element index of every element that shares node i.

After the total quality indicators have been calculated for each node of the mesh, these values are linearly interpolated in order to simultaneously calculate a field of element quality indicators (Q) and the new nodal positions. The quality indicators and nodal positions are recalculated during every iteration of the solution.

The relationship between Q and the new nodal positions depends on your setup parameters. As stated previously, the elastic remeshing scheme is governed by the pseudo-elastic equation (*Equation 32–3* (p. 621)), which includes a variable that functions as Young's modulus (E). Your setup can result in E behaving like one of the following:

- a function of the current and initial fields of element quality indicators

  $$E = \left(\frac{Q}{Q^0} \right)^n \tag{32–7}$$

  where Q^0 is the initial field of element quality indicators (calculated prior to any deformation) and n is a user constant that is set to 2 by default. With this equation, all elements have the same stiffness (i.e., $E = 1$) for the first iteration. As the solution iterates, the stiffness of each element will increase as it becomes more distorted relative to its initial state, while the stiffness will decrease as it becomes less distorted relative to its initial state. Note that *Equation 32–7* (p. 625) is the default definition for E.

- a function of the current field of element quality indicators

  $$E = (Q)^n \tag{32–8}$$

  where n is a user constant that is set to 2 by default. With this equation, the elements that have a worse quality indicator for a given iteration are stiffer than those whose quality indicator is better. Therefore, the good elements will be deformed more than the bad elements during remeshing, immediately during the first iteration of the solution.

- a unit value

  $$E = 1 \tag{32–9}$$

  In this case, the Young's modulus is constant and the quality of the elements are not factored into the remeshing algorithm.

The effect of the exponent n in *Equation 32–7* (p. 625) and *Equation 32–8* (p. 625) is to amplify the stiffness of distorting / distorted elements. Note that the problem becomes nonlinear if *Equation 32–7* (p. 625) or *Equation 32–8* (p. 625) is used to calculate E; in such circumstances n governs the nonlinearity, such that higher values for n make the convergence more complicated and the runtime longer. Despite such complications, it is recommended that you use *Equation 32–7* (p. 625) or *Equation 32–8* (p. 625) for complex deformations (e.g., along a deformation line); you should only use *Equation 32–9* (p. 625) for simple deformations, such as a uniform change in the length of a domain.

32.2.2. Domain Transformations

You have the following options for deforming or moving sub-domains (i.e., volumes in a 3D mesh, or surfaces in a 2D mesh):

- fixed domain

  For the “fixed domain" method, the motion of the nodes is governed by *Equation 32–1* (p. 621). As a result, all of the nodes of the domain remain fixed at their original coordinates. This motion is applied on all topological entities included in the domain definition (i.e., surfaces, lines, and points). Hence, it is not possible to apply another motion on these entities during the motion definition of an adjacent domain.

- rigid translation

  For the “rigid translation" method, the motion of the nodes is governed by *Equation 32–2* (p. 621). This motion is applied on all topological entities included in the domain definition (i.e., surfaces, lines, and points). Hence, it is not possible to apply another motion on these entities during the motion definition of an adjacent domain.

- elastic remeshing

  For the “elastic remeshing" method, the motion of the nodes is governed by *Equation 32–3* (p. 621). This equation requires boundary conditions, which for 3D domains can be selected from the surface transformations described in *Surface Transformations* (p. 626), and for 2D domains can be selected from the line transformations described in *Line Transformations* (p. 629).

32.2.3. Surface Transformations

You can apply deformation methods on surfaces that act as the boundary conditions of a 3D domain on which the elastic remeshing method has been applied. You have the following options for deforming or moving surfaces:

- no constraints on surface

  The “no constraints on surface" method does not apply an equation on the surface. The coordinates of the nodes belonging to the surface are governed by the equations applied on the 3D domain that contains the surface.

- elastic remeshing

  For the “elastic remeshing" method, the motion of the nodes is governed by *Equation 32–3* (p. 621). The pseudo-elastic equation defined on the surface replaces the 3D equation for the interior nodes of the surface. This equation requires boundary conditions, which can be selected from the line transformations described in *Line Transformations* (p. 629).

- planar elastic remeshing

  The “planar elastic remeshing" method can only be applied on a flat surface, and the methods applied on its borders must induce deformations that will stretch the surface without any torsion (i.e., the surface must remain flat). The motion of the nodes is governed by *Equation 32–3* (p. 621), coupled with the constraint on the normal displacement shown in *Equation 32–4* (p. 621). The pseudo-elastic equation defined on this surface will replace the 3D equation for the interior nodes of the surface. This equation requires boundary conditions, which can be selected from the line transformations described in *Line Transformations* (p. 629).

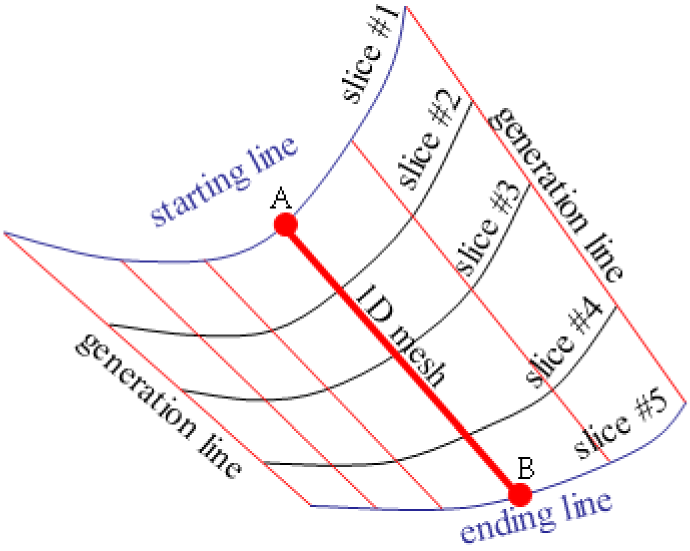
- sliceable ruled surface

  The “sliceable ruled surface" method can be applied when you want the interior nodes to move along a ruled surface, which is initially defined between two lines (referred to as the “starting line" and the “ending line"). For this method to be appropriate, the mesh related to the surface must meet the following conditions:

- All the nodes on the mesh must align in straight lines between the nodes of the starting and ending lines. Note that the straight lines on the edges of the surface can be referred to as “generation lines".
- The mesh must be made up entirely of quadrilateral elements. The edges of the quadrilateral elements must be able to be grouped into “slices" that cut across the generation lines, where each slice has the same number of segments.

See *Figure 32.4* (p. 627) for an example of a sliceable ruled surface. The surface can be open (i.e., having 4 boundaries, as shown in *Figure 32.4* (p. 627)), or closed (e.g., a tube). For closed surfaces, there are only two boundaries: the starting and the ending lines.

Figure 32.4 An Example of a Sliceable Ruled Surface



The surface can be described as a set of 1D meshes, where each of these meshes is made of segments that start from a node (A) on the starting line and end on a corresponding node (B) on the ending line. The motion of the nodes along these 1D meshes is governed by the pseudo-elastic equation (*Equation 32–3* (p. 621)). You define the starting line, and set up boundary conditions on the starting and ending lines using the line transformations described in *Line Transformations* (p. 629).

In the cases where there are several adjacent sliceable ruled surfaces, you must ensure that the surfaces are oriented appropriately. For details, see *Remarks and Limitations* (p. 636).

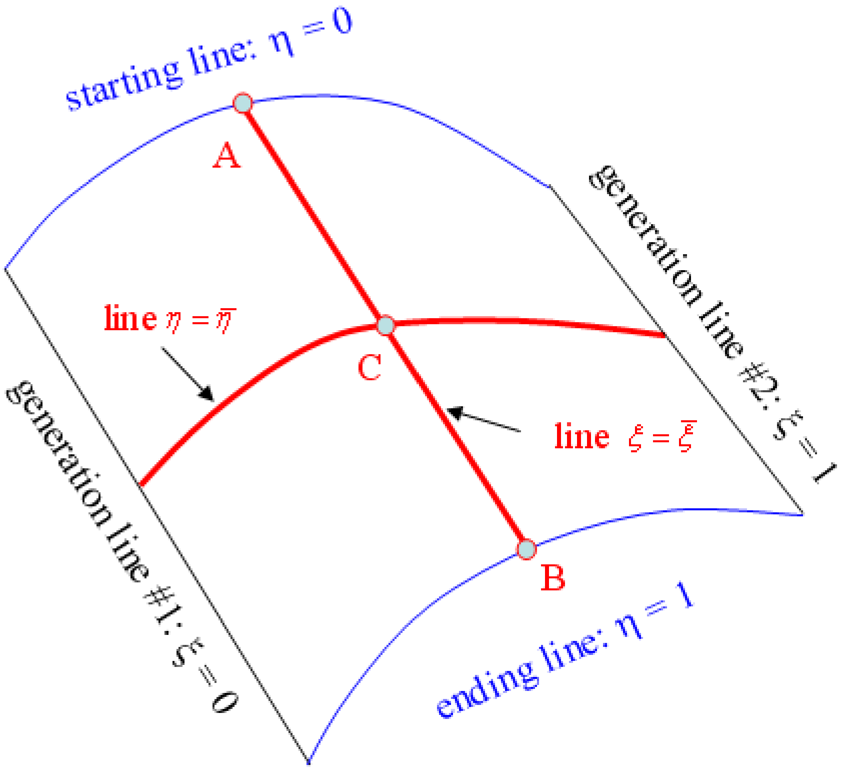
- ruled surface

  The “ruled surface" method can be applied when you want the interior nodes to move along a ruled surface, which is initially defined between two lines (referred to as the “starting line" and the “ending line"). For this method to be appropriate, the mesh related to the surface must meet the following conditions:

  - The surface must have four boundary lines, (i.e., it cannot be a closed surface, like a tube). The two opposing boundary lines referred to as the starting and ending lines may be curved, while the other two opposing lines (called the “generation lines") must be straight.
  - All the nodes of the mesh must be located directly between a point on the starting line and a point on the ending line.

While the ruled surface method shares some similarities with the sliceable ruled surface described earlier, there are some differences. First of all, the ruled surface is not restricted to meshes that are composed entirely of quadrilateral elements. Secondly, the surface is not converted into a set of 1D meshes. Instead, the nodes are constrained to only move along imaginary lines that begin on the starting line and end on the ending line. To generate these imaginary lines, a local coordinate system (ξ, η) is created for the surface (see *Figure 32.5* (p. 628)). The ξ coordinate varies linearly along the surface from generation line #1 (where $\xi = 0$) to generation line #2 (where $\xi = 1$). The η coordinate varies linearly along the surface from the starting line (where $\eta = 0$) to the ending line (where $\eta = 0$). During the mesh deformation, each node (e.g., C) on the surface of a given local coordinate $(\overline{\xi}, \overline{\eta})$ is moved in order to stay on a straight line (e.g., line AB) where $\eta = \overline{\eta}$, and on the curved line where $\xi = \overline{\xi}$. Linear interpolation is used to determine the movement of the node, rather than the pseudo-elastic equation.

Figure 32.5 An Example of a Ruled Surface


starting line: η = 0
A
generation line #2: ξ = 1
line $\eta = \overline{\eta}$
C
line $\xi = \overline{\xi}$
generation line #1: ξ = 0
B
ending line: η = 1

You define the starting line, and set up boundary conditions on the starting and ending lines using the line transformations described in *Line Transformations* (p. 629).

In the cases where there are several adjacent ruled surfaces, you must ensure that the surfaces are oriented appropriately. For details, see *Remarks and Limitations* (p. 636).

- rigid translation

For the “rigid translation" method, the motion of the nodes is governed by *Equation 32–2* (p. 621). This method is applied on all topological entities included in the domain definition (i.e., lines and points). Hence, it is not possible to apply another motion on these entities when defining the deformation of an adjacent domain.

- fixed surface

  For the “fixed surface" method, the motion of the nodes is governed by *Equation 32–1* (p. 621). As a result, all the nodes of the surface remain fixed at their original coordinates. This method is applied on all topological entities included in the domain definition (i.e., lines and points). Hence, it is not possible to apply another motion on these entities when defining the deformation of an adjacent domain.

- symmetry

  For the “symmetry" method, a flat plane is treated as a plane of symmetry. In other words, the surface is deformed due to the methods applied on its borders, with the constraint that the nodes remain in the original plane. The motion of the nodes is governed by the elastic remeshing method applied on the 3D domain, with the constraint on the normal displacement shown in *Equation 32–4* (p. 621). Note that the selection of the symmetry method for the surface transformation does not define the flow as being symmetrical about this surface, as the two conditions are not linked.

32.2.4. Line Transformations

You can apply deformation methods on lines to define the boundary conditions of a surface in a domain on which the elastic remeshing method has been applied. You have the following options for deforming or moving lines:

- free line

  The “free line" method does not apply an equation on the line. The coordinates of the nodes belonging to this line are governed by the equations applied on the 2D domain that contains this line.

- elastic remeshing

  For the “elastic remeshing" method, the motion of the nodes is governed by *Equation 32–3* (p. 621). The pseudo-elastic equation defined on this line replaces the 2D equation for the interior nodes of the line. This equation also requires boundary conditions, which can be selected from the point displacement methods described in *Point Displacement* (p. 634).

- line deformation

  For the “line deformation" method, the motion of each node is governed by the equation used for the rigid translation method (*Equation 32–2* (p. 621)). But the line deformation method differs from the rigid translation method, in that the displacement direction and amplitude need not be uniform for all the nodes. The amplitude a and the component direction D_i are given by second order differential equations:

  $$\Delta a = 0 \tag{32–10}$$

  and

$$\Delta D_i = 0 \tag{32–11}$$

where Δ is the Laplace operator.

From a mathematical point of view, *Equation 32–10* (p. 629) and *Equation 32–11* (p. 630) a and one value for D_i in order to yield a solution (you may define more). At these points

$$a = \bar{a} \tag{32–12}$$

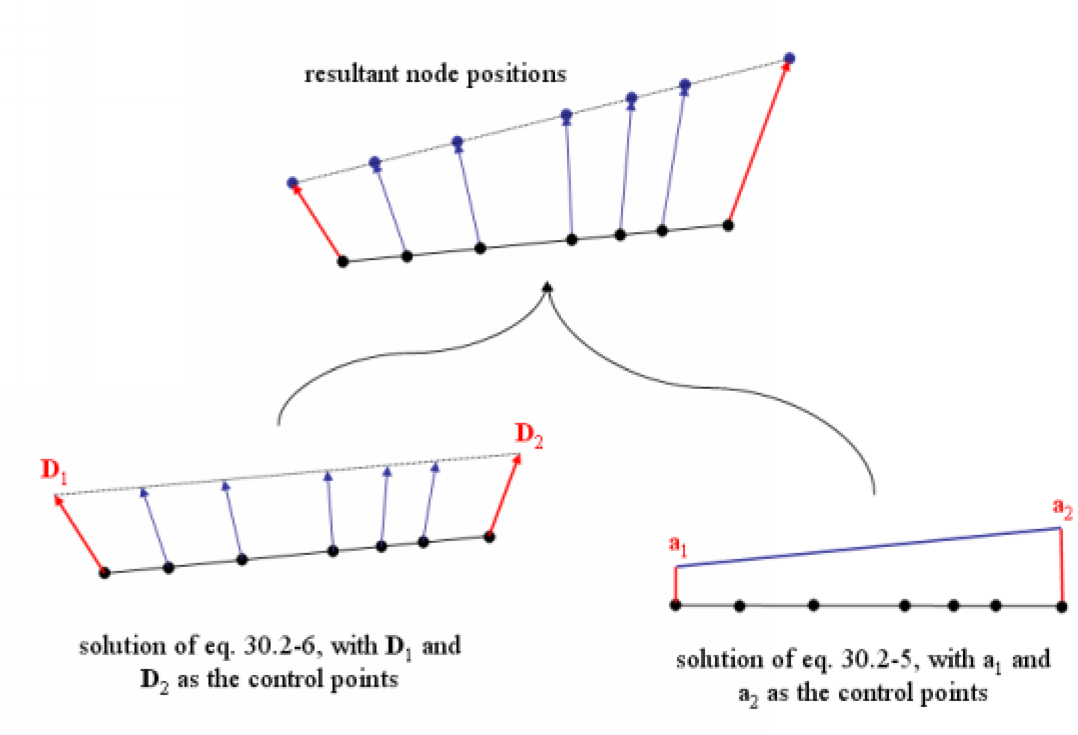
and

$$D_i = \overline{D_i} \tag{32–13}$$

where $\bar{a}$ and $\overline{D_i}$ are the values you defined for the amplitude and the direction components, respectively.

The nodes where you defined the direction and/or the displacement amplitude are referred to as the “control points". The result of *Equation 32–10* (p. 629) and *Equation 32–11* (p. 630) is a linear variation between two successively defined values. *Figure 32.6* (p. 630) shows the solution of *Equation 32–11* (p. 630) for the direction with $\mathbf{D}_1$ and $\mathbf{D}_2$ for the boundary conditions (left bottom), the solution of *Equation 32–10* (p. 629) for the displacement amplitude with a_1 and a_2 for the boundary conditions (bottom right), and the solution of *Equation 32–2* (p. 621) with the new node positions (top).

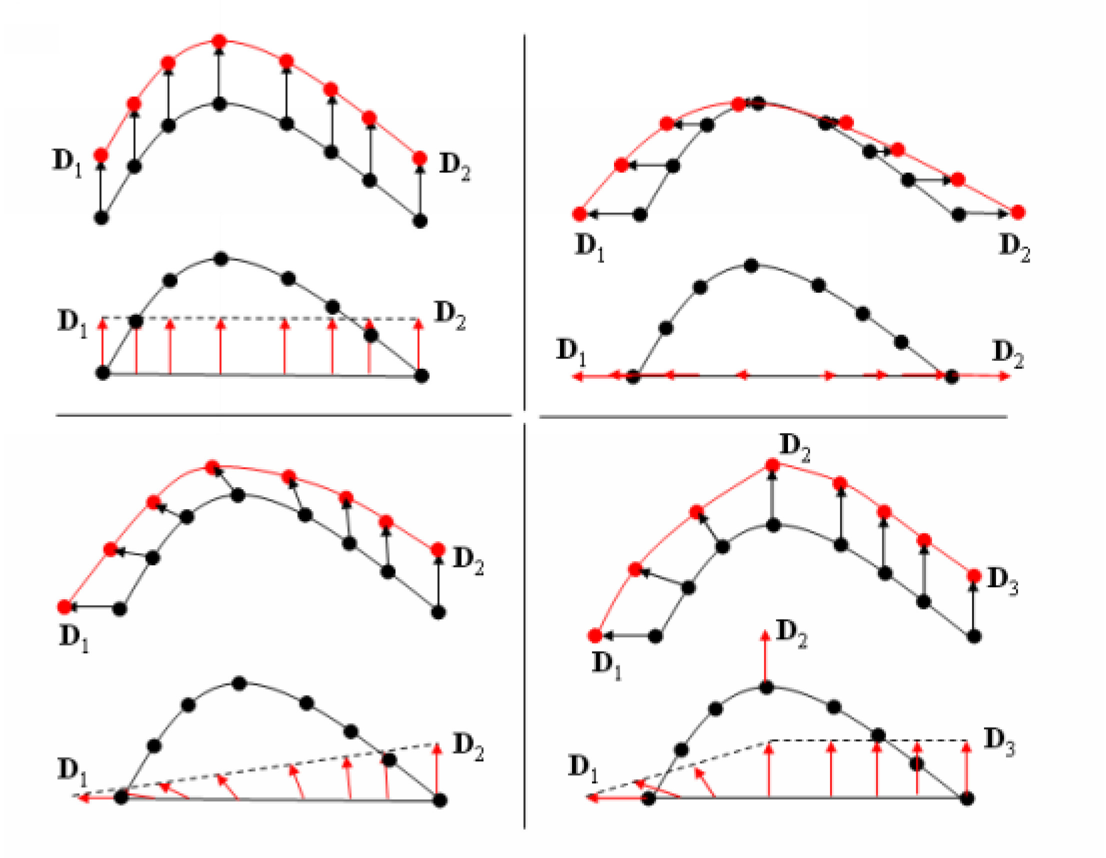
Figure 32.6 The Motion of Nodes due to the Line Deformation Method


resultant node positions
D1
D2
a1
a2
solution of eq. 30.2-6, with D1 and D2 as the control points
solution of eq. 30.2-5, with a1 and a2 as the control points

This method can also be applied on curved lines. For curved lines, note that *Equation 32–11* (p. 630) does not take the curvature into account, and the direction vectors are not the normal vectors to the curve.

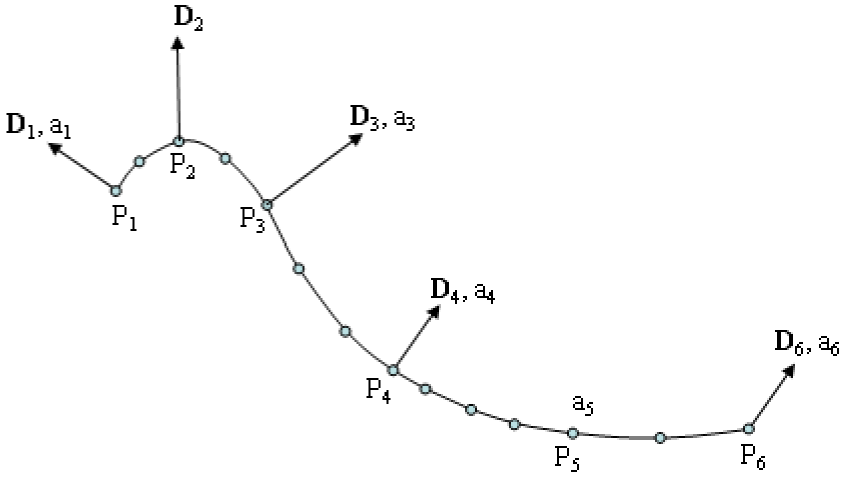
Figure 32.7 (p. 631) shows examples of different direction vectors $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{D}_3$ imposed on a curved line, with the displacement amplitude set to 1. For each case, the lower portion of the figure depicts the initial curve and the directions computed on the basis of $\mathbf{D}_1$, $\mathbf{D}_2$, and $\mathbf{D}_3$, while the upper portion shows the resultant node positions.

Figure 32.7 The Line Deformation Method on a Curved Line



You do not have to impose a value on both the direction and amplitude for a given node. In other words, you can impose a value for one, or the other, or both, as shown in *Figure 32.8* (p. 632).

Figure 32.8 The Line Deformation Method with Multiple Control Points



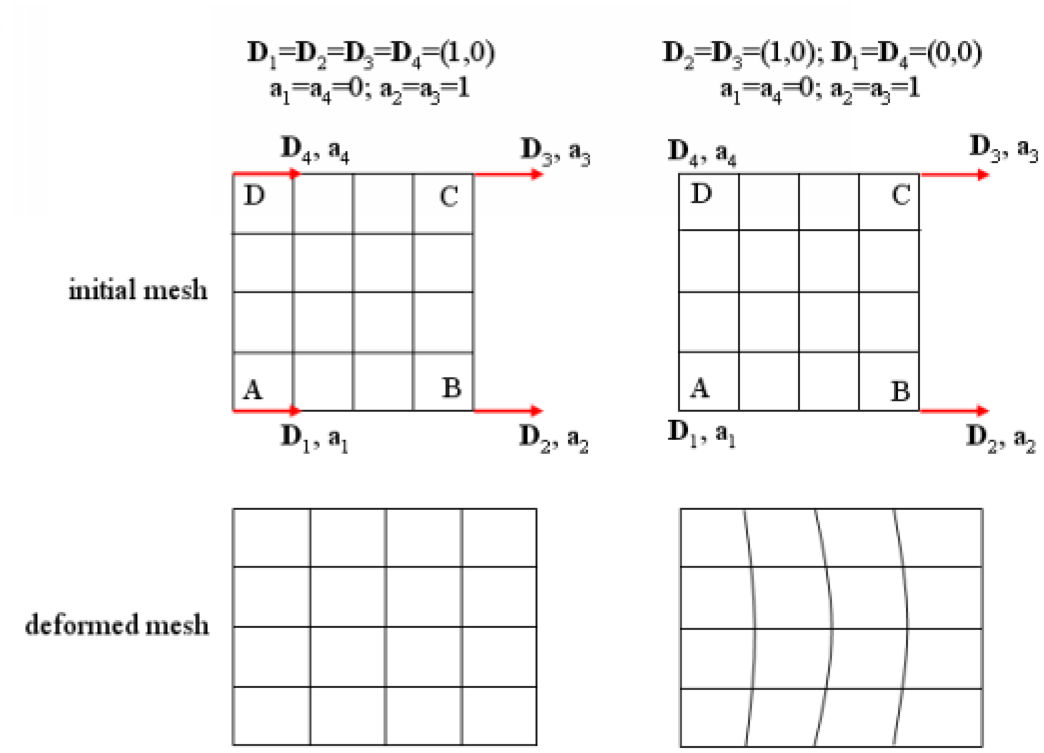
The following is a description of the amplitude and direction components on the nodes between P_1 and P_3 in *Figure 32.8* (p. 632). The direction components vary linearly between node P_1 and node P_2 (ranging from $\mathbf{D}_1$ to $\mathbf{D}_2$), as well as between node P_2 and node P_3 (ranging from $\mathbf{D}_2$ to $\mathbf{D}_3$). The amplitude values, on the other hand, vary linearly between node P_1 and node P_3 (ranging from a_1 to a_3), as there is no imposed amplitude at node P_2. A similar description could be made for the nodes between P_4 and P_6, which have imposed amplitude values at P_4, P_5, and P_6, but have direction components only imposed at P_4 and P_6.

It is recommended that the line deformation method not be used on lines that are made of several disconnected parts. Such a circumstance can arise, for example, when a line is defined by the intersection of two surfaces that meet in separate locations. If you do decide to use the line deformation method for such a line, then you must define at least one control point on each of the disconnected parts.

In order to provide you with flexibility when defining the point displacement, the direction you specify does not need to be normalized. Still, it is recommended that you specify unit vectors for the control points, as this can lead to better control.

If you want a control point in a line to be fixed, it is recommended that you set the amplitude value equal to 0. You could also accomplish this by setting the direction equal to the null vector. *Figure 32.9* (p. 633) illustrates the results of both methods on a line that belongs to a surface, on which is applied the elastic remeshing method.

Figure 32.9 Mesh Deformation Including Fixed Points



On the left side of the *Figure 32.9* (p. 633), a unit vector (1,0) is imposed for the four directions $\mathbf{D}_1$, $\mathbf{D}_2$, $\mathbf{D}_3$, and $\mathbf{D}_4$ at control points A, B, C, and D, respectively. The amplitude is set to 0 at points A and D, and set to 1 at points B and C. The resultant displacement is then 0 along line AD, and the vector (1,0) along line BC. For the nodes in lines AB and DC, the direction is constant (1,0) and the amplitude displacement varies linearly from 0 (points A and D) to 1 (points B and C). Hence, the resulting displacement varies linearly along lines AB and DC.

The settings for the control points on the right side of *Figure 32.9* (p. 633) are similar, except that the null vector (0,0) is imposed for the directions $\mathbf{D}_1$ and $\mathbf{D}_4$ at points A and D, respectively. While the resultant displacements along lines AD and BC are the same as on the left side of the figure, they are different along lines AB and DC. For lines AB and DC, the x-component of the direction varies linearly from 0 to 1, while at the same time the amplitude displacement varies linearly from 0 to 1. The resultant displacement is therefore no longer linear along AB and DC, and the nodes are not equidistant. Note that the interior nodes behave differently than the nodes of line AB and DC, as the elastic remeshing method attempts to respect the initial distribution.

- rigid translation

  For the “rigid translation" method, the motion of the nodes is governed by *Equation 32–2* (p. 621). This method is applied on all topological entities included in the domain definition (i.e., points). Hence, it is not possible to apply another motion on these points when defining the deformation of an adjacent domain.

- fixed surface

  For the “fixed surface" method, the motion of the nodes is governed by *Equation 32–1* (p. 621). As a result, all the nodes of the surface remain fixed at their original coordinates. This method is applied on all topological entities included in the domain definition (i.e., points). Hence, it is not possible to apply another motion on these points when defining the deformation of an adjacent domain.

- symmetry

  For the “symmetry" method, a straight line on a 2D geometry is treated as a line of symmetry. In other words, the line is deformed due to the methods applied on its extremities, with the constraint that the nodes remain in the original line. Note that the selection of the symmetry method for line deformation does not define the flow as symmetrical about this line, as the two conditions are not linked.

32.2.5. Point Displacement

You can apply point displacement methods on nodes in order to define the boundary conditions of a line in a domain on which the elastic remeshing method has been applied. You can also apply them to define the displacement of control points for lines on which the line deformation method is applied. You have the following options for moving points:

- free point

  No motion is prescribed for a node designated as a free point. The motion of the node is governed by the method applied on the domain of higher dimension that includes the free point.

- direction imposed

  When only a direction is imposed on a node, the motion has an amplitude value determined by *Equation 32–10* (p. 629) and direction component determined by *Equation 32–13* (p. 630).

- direction and amplitude imposed

  When a direction and amplitude is imposed on a node, the motion has an amplitude value determined by *Equation 32–12* (p. 630) and direction component determined by *Equation 32–13* (p. 630).

- amplitude imposed

  When only an amplitude is imposed on a node, the motion has an amplitude value determined by *Equation 32–12* (p. 630) and direction component determined by *Equation 32–11* (p. 630).

- fixed point

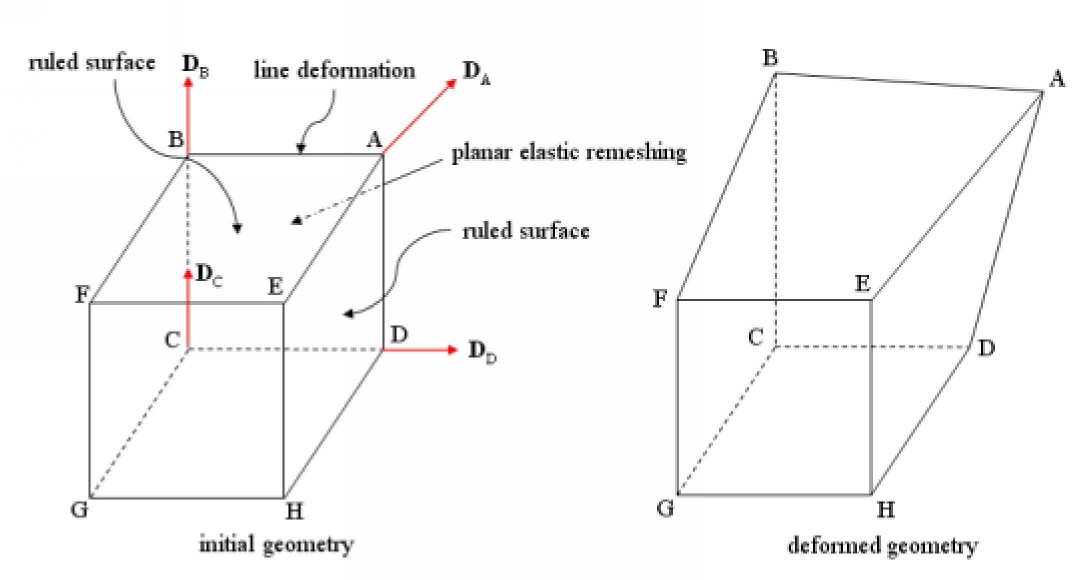
  When a node is designated as a fixed point, the motion of the node is governed by *Equation 32–1* (p. 621). As a result, the node remains fixed at its original coordinates.

32.2.6. Hierarchy of Equations

When deformation methods are defined for a mesh, it is possible that a node may have more than one equation applied to it. For example, while a node may have a point displacement method applied on it, it may also be associated with a line, surface, and volume, on which different equations are applied. A hierarchy is imposed on the equations to address this situation, such that the equations on the nodes have the greatest importance, then those on the lines, then those on the surfaces, and finally those on the volumes.

In order to illustrate the hierarchy of equations applied on domains of different dimensions, consider *Figure 32.10* (p. 635). For this example, the intention is to move nodes A and B in the plane of face ABCD.

Figure 32.10 The Deformation of a Cube by Displacement of Two Nodes



This example requires the solution of many equations, and several equations are defined at the same locations. Consider node A: this point belongs to one volume, three faces, and three lines, and a mesh deformation method is defined on each of them. Therefore, there are six deformation methods defined at point A:

- 3D elastic remeshing
- planar elastic remeshing on face ABCD (rear)
- ruled surface on face ABFE (top)
- ruled surface on face AEHD (right)
- line deformation on line ABCD
- displacement and amplitude imposed at point A.

Or consider the top face: it belongs to the volume, and thus there are two mesh deformation methods (3D elastic remeshing and ruled surface) applied on the interior nodes of the face.

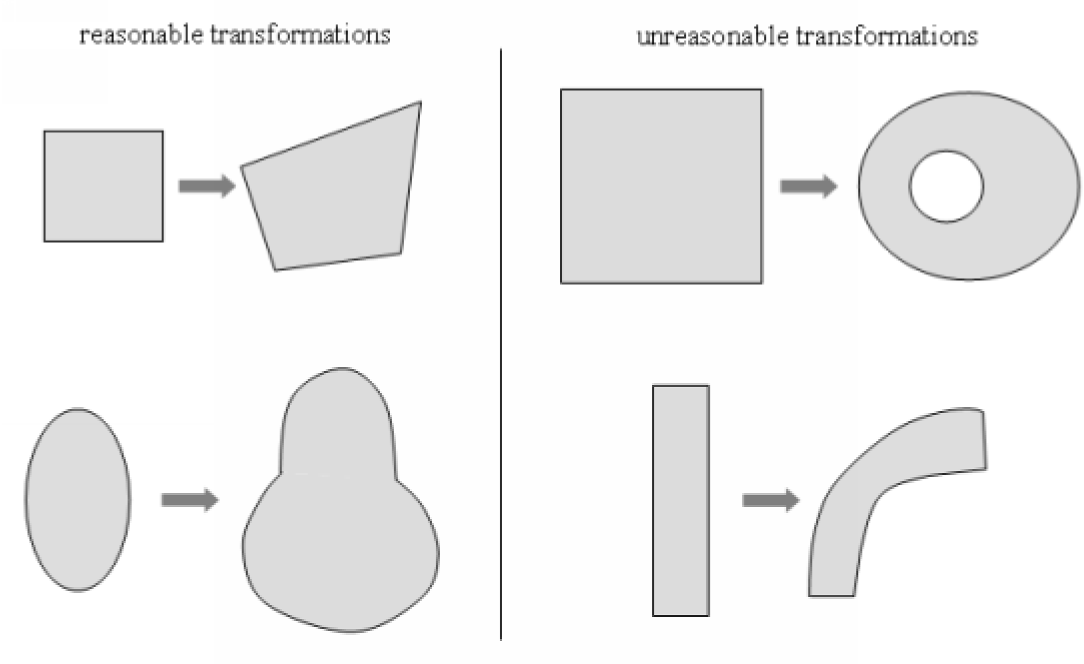
In order to avoid the summation of the contribution of different equations, a hierarchy is imposed so that the domain of the smaller dimension is solved first. After a value has been computed for a given dimension, its equation is not taken into account when solving for the domain of higher dimension. Hence, the computational order is as follows:

1. The direction and amplitude for the displacement of points are computed.
2. The direction and amplitude for the displacement of lines are computed.
3. The direction and amplitude for the displacement of faces are computed.
4. The direction and amplitude for the displacement of volume are computed.
5. The coordinates for points are computed.
6. The coordinates for lines are computed.
7. The coordinates for faces are computed.
8. The coordinates for volumes are computed.

32.3. Remarks and Limitations

The mesh deformation techniques described in the previous section are intended to result in reasonable deformations of the mesh. A reasonable deformation requires that the topology remain almost the same, with the same number of domains and same number of boundaries. It is not possible, for example, to start from a square and obtain a circle with a hole inside. *Figure 32.11* (p. 636) depicts examples of reasonable and unreasonable deformations.

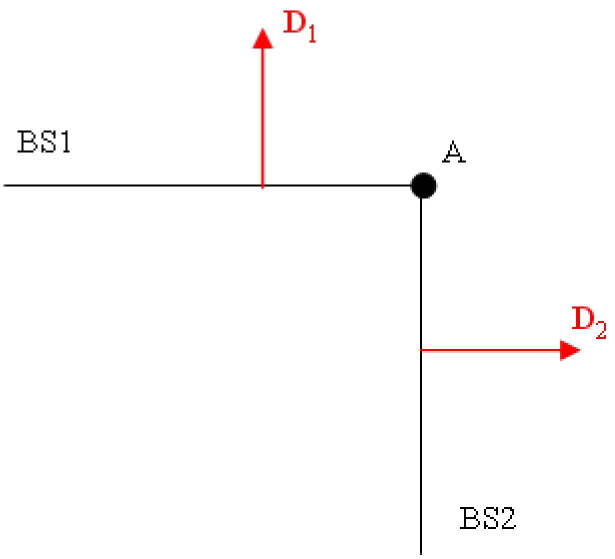
Figure 32.11 Reasonable and Unreasonable Mesh Transformations



Some considerations when evaluating the reasonableness of a deformation include the following:

- Geometric entities cannot be rotated.
- It is not possible to impose two independent rigid translations on two adjacent entities, as shown in *Figure 32.12* (p. 637). In this example, a rigid translation with $\mathbf{D}_l = (0,1,0)$ is imposed on BS1, while another rigid translation with $\mathbf{D}_l = (1,0,0)$ is imposed on BS2. Such a circumstance leads to a conflict for the direction at point A.

Figure 32.12 An Example of an Invalid Setup for Rigid Translations D_1 and D_2



- Directions can only be given as constant values, and it is not possible to define them using an evolution parameter, a UDF, or a UDT.
- You must make sure that adjacent ruled surfaces (sliceable or not) are oriented the same way. For example, *Figure 32.13* (p. 637) illustrates the case when two adjacent ruled surfaces share a generation line. To ensure a compatible orientation, you must select their starting lines so that they share a vertex.

Figure 32.13 Adjacent Ruled Surfaces Connected at Generation Lines

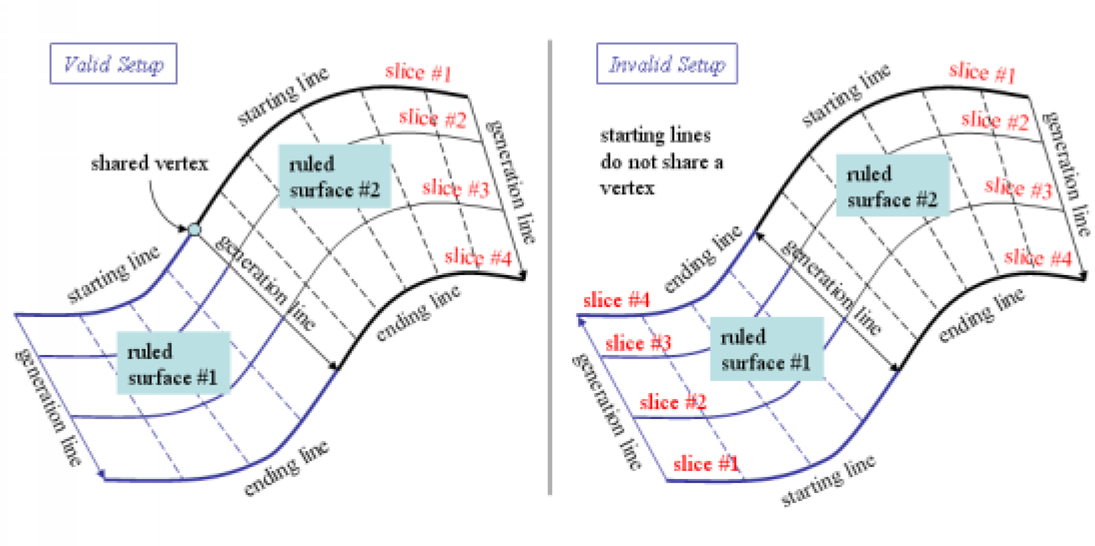
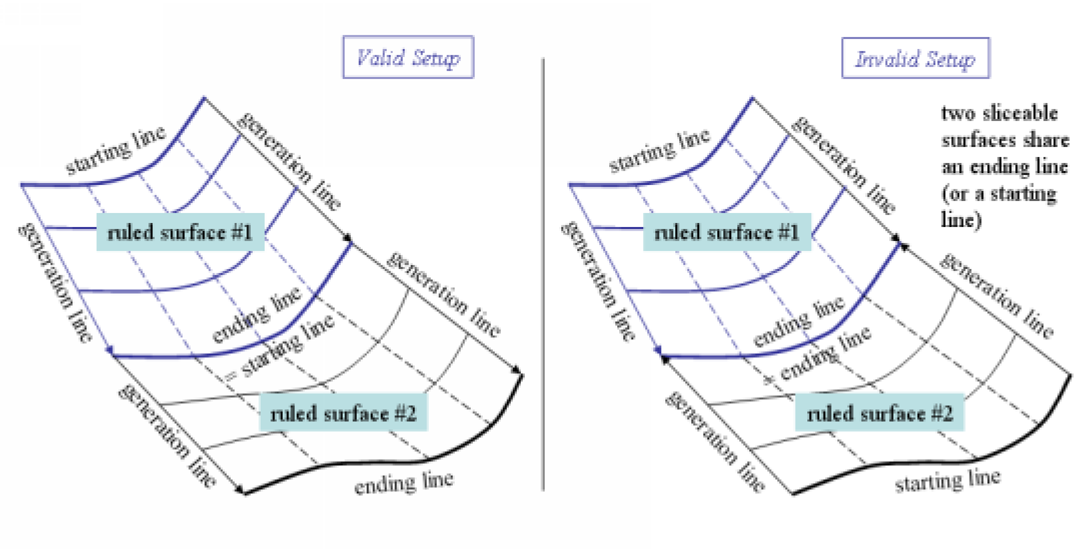


Figure 32.14 (p. 638) depicts another possible configuration with adjacent ruled surfaces, in which they share a starting and/or ending line. In this situation, you must make sure that the starting line of one of the surfaces corresponds to the ending line of the adjacent surface. In this manner, both surfaces are oriented in the same way.

Figure 32.14 Adjacent Ruled Surfaces Connected at Ending / Starting Lines



- It is a common practice to perform an initial simulation with a coarse mesh in order to test a model; if the simulation is successful, this same setup is then used with a more refined mesh. You must exercise caution, however, when attempting such a procedure if you defined methods of mesh deformation using control points. ANSYS POLYFLOW references control points by their vertex ID, therefore if you alter the mesh without updating the data file, the vertex IDs will point to vertices other than the ones you expect. As a result, the mesh will deform in an unexpected manner. In order to avoid such a circumstance, always be sure to load your data file in ANSYS POLYDATA with your new mesh. ANSYS POLYDATA will automatically update the data structure in the following ways:
  - For vertices that are at the extremities of lines, the vertex IDs will be updated such that the new vertex IDs correspond exactly to the same line extremities.
  - For vertices that are included in a line (corresponding to added control points), the vertex IDs will be updated such that the new vertex IDs correspond to the closest vertex on the same line.

  You should check if the closest vertex selected by ANSYS POLYDATA corresponds well with the defined setup (e.g., direction of displacement); if not, you can select another vertex on the line. When you have reviewed any questionable vertices, you can save and exit ANSYS POLYDATA. It is recommended that you save the data file with a new name, in order to distinguish it from the original data file.

32.4. Problem Setup

The definition of the mesh deformation methods is done by creating sub-tasks of type **Preprocessor** in the task menu. Each preprocessor sub-task is defined on a specific domain, and their domains must not overlap. After your mesh deformation sub-tasks are defined, you can define the setup of the flow problem as usual. As indicated in the name, the preprocessor sub-tasks will all be solved in ANSYS POLYFLOW in specific solvers prior to the evaluation of the main flow problem.

To begin defining the mesh deformation for a preprocessor sub-task, read a mesh and create a task. Then perform the following steps for each domain:

1. Create a new sub-task.

   Create a sub-task

a. Click **Preprocessor** for the sub-task type.

Preprocessor

b. When prompted, enter a name for the sub-task and click **OK**.

2. Define the appropriate domain transformation method, based on the manner in which you want the mesh to deform.

   Elastic remeshing

   Fixed domain

   Rigid translation

   The steps for using each of these transformation methods are described in detail in the sections that follow, starting with the fixed domain method.

32.4.1. Fixed Domain

You can use the fixed domain method to specify that all of the nodes of the domain remain fixed at their original coordinates.

1. Click **Fixed domain** in the **Preprocessor** menu.

   Fixed domain

   The **Preprocessor: mesh deformation** menu will open.

2. Specify the domain of the fixed domain method.

   Note that the previous procedure for defining the deformation domain is the same for all transformation methods.

   a. Click **Deformation domain** to open the **Domain of the sub-task** menu.

      Deformation domain

   b. Modify the upper list of subdomains until it matches your best estimate of the domain on which the sub-task will be defined. To remove a subdomain, select it in the upper list and click the **Remove** button. To add a subdomain, select it in the lower list and click the **Add** button.

   c. Click the **Upper level menu** button to return to the **Preprocessor: mesh deformation** menu.

      If one or more of your selected subdomains have already been used to define another preprocessor sub-task for mesh deformation, you will be warned. If you receive this warning, you must modify the domain of the appropriate preprocessor sub-task, in order to avoid any overlap in their domain definitions.

3. Click **Upper level menu** to return to the **Task** menu, and continue defining the task.

32.4.2. Rigid Translation

You can use the rigid translation method to specify that all of the nodes of the domain have a uniform motion in a given direction.

1. Click **Rigid translation** in the **Preprocessor** menu.

   Rigid translation

   The **Preprocessor: mesh deformation** menu will open.

2. Specify the domain of the rigid translation method using the **Domain of the sub-task** menu, in the manner described previously for the **Fixed domain** method.

   Deformation domain

3. Specify the direction and the amplitude for the displacement of the nodes.

   a. Click **Model parameters** to open the **Parameters for rigid translation** menu.

      Model parameters

   b. Click **Modify direction of movement** and revise the default values in the panels that open, as necessary.

   c. Click **Modify amplitude of movement** and revise the default value in the panel that opens, as necessary.

      Important

      You can use an evolution function or a user-defined template to define the amplitude. Simply click the **EVOL** or **UPDT** button under the ANSYS POLYDATA menu bar prior to clicking **Modify amplitude of movement**. After you have approved the amplitude value in the panel, the appropriate menus will open. Click **Upper level menu** to return to the **Parameters for rigid translation** menu.

   d. Click **Upper level menu** to return to the **Preprocessor: mesh deformation** menu.

4. Click **Upper level menu** to return to the **Task** menu, and continue defining the task.

32.4.3. Elastic Remeshing

You can use the elastic remeshing method to specify that the motion of the interior nodes is induced by the motion of the boundaries.

1. Select the **Elastic remeshing** menu.

   Elastic remeshing

2. Specify the domain of the elastic remeshing method using the **Domain of the sub-task** menu, in the manner described previously for the fixed domain method.

   Deformation domain

3. Define the deformation of the domain by specifying the deformation method for each of the topological entities, setting the mesh distortion check criteria, and defining the modulus used in the pseudo-elastic equation. Note that the elastic remeshing method requires sufficient boundary conditions for the entities to ensure that the whole domain does not undergo rigid motion.

   Because the number of entities can be quite large, they are grouped in lists according to their geometric dimension (i.e., surfaces, lines, and points). It is recommended that you follow a top-down approach, to methodically define the methods on each entity: first the surfaces, then the lines, and finally the points. At any point during this process you can click **Upper level menu** to automatically check if the methods applied on adjacent entities are compatible. If an incompatibility is detected, a message will appear in the output text window of ANSYS POLYDATA (see *Figure 2.14* (p. 28)) that explains the cause of incompatibility and the topological entities involved.

   a. For 3D geometries, define the method of deformation and associated boundary conditions for surfaces.

      Methods on surfaces

      The **Methods on surfaces of domain <domain name>** menu will open (where **<domain name>** is the name of the domain you defined in the previous step), displaying a list of all of the relevant surfaces. By default, no methods are applied on the surfaces, and so they are listed in this menu as having **no constraints on surface**. If a surface has a method imposed on it because an adjacent domain already has a deformation method defined (e.g., **Fixed domain**, **Rigid translation**), the imposed method will be displayed in the list and you cannot modify it.

      i. Select a surface in the list and click the **Modify** button to enter the **Method along <surface name>** menu.

      ii. Click **Modify method: <current method name>**.

          Modify method:<current method name>

          The **Select method along <surface name>** menu will open.

      iii. Select the appropriate method for the surface. You have the following choices:

           - To allow the position of the nodes to be computed by the remeshing method defined on the domain of higher dimension, select **no constraints on surface**.

             Select method: no constraints on surface

             Click **Upper level menu** to return to the **Method along <surface name>** menu.

           - To specify that the motion of the interior nodes is induced by the boundary motions, select **elastic remeshing**.

             Select method: elastic remeshing

             This model requires boundary conditions, which you can define either by clicking **Boundary conditions** in the **Method along <surface name>** menu, or by clicking **Methods on lines** in the **Preprocessor: mesh deformation** menu.

             Click **Upper level menu** to return to the **Method along <surface name>** menu.

- If the surface is a plane and you would like the interior nodes to move in this plane with a motion that is induced by the boundary motions, select **planar elastic remeshing**.

  Select method: planar elastic remeshing

  This model requires boundary conditions, which you can define either by clicking **Boundary conditions** in the **Method along <surface name>** menu, or by clicking **Methods on lines** in the **Preprocessor: mesh deformation** menu.

  Click **Upper level menu** to return to the **Method along <surface name>** menu.

- To specify that the interior nodes move along a ruled surface induced by the motion of two opposing boundary lines, select **ruled surface**.

  Select method: ruled surface

  After you have selected this option, the **Specify the starting line of the ruled surface** menu item will be available for selection, so that you can specify the starting line. Select this option to open the **Select a starting line** menu. Select the appropriate line and click the **Modify** button. The selected line will then be listed as the **Starting line**.

  This model requires boundary conditions, which you can define either by clicking **Boundary conditions** in the **Method along <surface name>** menu, or by clicking **Methods on lines** in the **Preprocessor: mesh deformation** menu. It is recommended that you allow the generation lines to be free lines (i.e., have no constraints imposed).

  Click **Upper level menu** repeatedly until you return to the **Method along <surface name>** menu.

  ### Important

  The **ruled surface** menu item should only be selected if the surface meets the criteria described for the ruled surface method in *Surface Transformations* (p. 626). If the surface also meets the criteria for the sliceable ruled surface method, you should consider using this latter method (as described in the option that follows), as the sliceable ruled method moves the nodes according to the pseudo-elastic equation. The ruled surface method, on the other hand, uses linear interpolation for the motion of the nodes.

- To specify that the interior nodes move along a ruled surface induced by the motion of two opposite boundary lines, select **sliceable ruled surface**.

  Select method: sliceable ruled surface

  After you have selected this option, the **Specify the starting line of the ruled surface** item will be available for selection, so that you can specify the starting line. Select this option to open the **Select a starting line** menu. Select the appropriate line and click the **Modify** button. The selected line will then be listed as the **Starting line**.

  This model requires boundary conditions, which you can define either by clicking **Boundary conditions** in the **Method along <surface name>** menu, or by clicking

Methods on lines in the **Preprocessor: mesh deformation** menu. It is recommended that you allow the generation lines to be free lines (i.e., have no constraints imposed).

Click **Upper level menu** repeatedly until you return to the **Method along <surface name>** menu.

> **Important**
>
> ANSYS POLYDATA does not check if the surface is really sliceable. It is your responsibility to provide a sliceable mesh on the surface.

- To specify uniform motion for the nodes in a given direction, select **rigid translation**.

  Select method: rigid translation

  After you have selected this option, the **Modify direction of movement** and **Modify amplitude of movement** items will become available. Click these items and revise the default values in the panels that open, as necessary.

  > **Important**
  >
  > You can use an evolution function or a user-defined template to define the amplitude. Simply click the **EVOL** or **UPDT** button under the ANSYS POLYDATA menu bar prior to clicking **Modify amplitude of movement**. After you have approved the amplitude value in the panel, the appropriate menus will open. Click **Upper level menu** to return to the **Select method along<surface name>** menu.

  After the direction and amplitude are defined, click **Upper level menu** to return to the **Method along <surface name>** menu.

- To specify that the nodes of the surface remain fixed at their original coordinates, select **fixed surface**.

  Select method: fixed surface

  Click **Upper level menu** to return to the **Method along <surface name>** menu.

- To specify that the surface be treated as a plane of symmetry, select **symmetry**.

  Select method: symmetry

  This model requires boundary conditions, which you can define either by clicking **Boundary conditions** in the **Method along <surfacename>** menu, or by clicking **Methods on lines** in the **Preprocessor: mesh deformation** menu.

  Click **Upper level menu** to return to the **Method along <surface name>** menu.

iv. Specify the boundary conditions of the surface.

Boundary conditions

The **Methods on lines of <surface name>** menu will open, presenting a list of boundary lines for the current surface. By default, any undefined line will be listed as a **free line** (i.e., no constraints).

Select a line in the list and click the **Modify** button. The **Method along <line name>** menu will open, which provides options for defining the deformation of the line. See the section that follows related to the setup for the **Methods on lines of domain <domain name>** menu for further details.

Click **Upper level menu** to return to the **Methods on lines of <surface name>** menu and continue selecting lines for modification until all the boundary conditions for the surface are appropriately defined. Then click the **Upper level menu** button to return to the **Method along <surface name>** menu.

v. Click **Upper level menu** to return to the **Method on surfaces of domain <domain name>** menu.

vi. Repeat steps 3.(a)i.–3.(a)v. for each of the remaining surfaces.

vii. Click **Upper level menu** to return to the **Preprocessor: mesh deformation** menu.

b. Define the method of deformation and associated boundary conditions for lines. Note that this step is only necessary if you did not specify all of the boundary conditions when defining the motion of the surfaces, or if you want to modify an existing line setup.

Methods on lines

The **Methods on lines of domain <domain name>** menu will open (where **<domain name>** is the name of the domain you defined in a previous step), displaying a list of all of the relevant lines. By default, no method is applied on a given line, and it is listed as being a **free line**. A line may be listed with a deformation method as a result of your boundary condition definitions for the motion of surfaces, or if a method has been imposed on it because an adjacent domain already has a deformation method defined. In the latter case, you cannot modify the imposed method.

i. Select a line in the list and click the **Modify** button to open the **Method along <line name>** menu.

ii. Click **Modify method: <current method name>**.

Modify method:<current method name>

The **Select method along <line name>** menu will open.

iii. Select the appropriate method for the line. You have the following choices:

- To allow the position of the nodes to be computed by the remeshing method defined on the domain of higher dimension, select **free line**.

  Select method: free line

  Click **Upper level menu** to return to the **Method along <line name>** menu.

- To specify that the motion of the interior nodes is induced by the boundary motions, select **elastic remeshing**.

Specify method: elastic remeshing

This model requires boundary conditions, which you can define either by clicking **Boundary conditions** in the **Method along <line name>** menu, or by clicking **Methods on points** in the **Preprocessor: mesh deformation**menu.

Click **Upper level menu** to return to the **Method along <line name>** menu.

- To specify the deformation of a line by defining control points, select **line deformation**.

Select method: line deformation

At a minimum, this model requires that you specify a value for the direction at one node and a value for the amplitude at one node. You may define further control points as desired, imposing a direction and/or the amplitude. You can define a control point either by clicking **Boundary conditions** in the **Method along <line name>** menu, or by clicking **Methods on points** in the **Preprocessor: mesh deformation** menu.

Click **Upper level menu** to return to the **Method along <line name>** menu.

- To specify uniform motion for the nodes in a given direction, select **rigid translation**.

Select method: rigid translation

After you have selected this option, the **Modify direction of movement** and **Modify amplitude of movement** items will become available. Click these items and revise the default values in the panels that open, as necessary.

Important

You can use an evolution function or a user-defined template to define the amplitude. Simply click the **EVOL** or **UPDT** button under the ANSYS POLYDATA menu bar prior to clicking **Modify amplitude of movement**. After you have approved the amplitude value in the panel, the appropriate menus will open. Click **Upper level menu** to return to the **Method along <line name>** menu.

After the direction and amplitude are defined, click **Upper level menu** to return to the **Method along <line name>** menu.

- To specify that the nodes of the line remain fixed at their original coordinates, select **fixed line**.

Select method: fixed line

Click **Upper level menu** to return to the **Method along <line name>** menu.

- To specify that the line be treated as a line of symmetry, select **symmetry**.

Select method: symmetry

Important

The **symmetry** method for lines is only available for 2D geometries.

This model requires boundary conditions, which you can define either by clicking **Boundary conditions** in the **Method along <line name>** menu, or by clicking **Methods on points** in the **Preprocessor: mesh deformation** menu.

Click **Upper level menu** to return to the **Method along <line name>** menu.

iv. Specify the boundary conditions of the line.

Boundary conditions

The **Methods on points of <line name>** menu will open, presenting a list of boundary nodes for the current line. By default, any undefined point will be listed as a **free point** (i.e., no constraints).

You can perform the following actions in this menu:

- modify the method on a control point

  Select a node in the list and click the **Modify** button. The **Modify control point** menu will open, which provides options for defining the motion of the control point. See the section that follows related to the setup for the **Methods on points of domain <domain name>** menu for further details.

  Click **Accept current setup - Vertex #<ID>** to return to the **Methods on points of <line name>** menu, and continue selecting nodes for modification until the boundary conditions for the line are appropriately defined.

- add a new control point

  Click the button to the left of **Add a new control point** to open the **Modify control point** menu, where you can select a node and designate the deformation method of the point. Note that the **Add a new control point** button is only available if you have specified the **line deformation** method for the associated line.

  By default, the first node of the line that is not already designated as a control point will be selected. You can select a different node by clicking either the **Move upstream -1** / **-5** item or the **Move downstream +1** / **+5** item as many times as necessary, thus moving along the line in increments of 1 or 5 nodes. The number of the node that is currently selected will be displayed in the **Modify control point** menu, and the graphics display window will show where it is located on the line.

  Next, select a deformation method for the new control point. See the section that follows related to the setup for the **Methods on points of domain <domain name>** menu for further details.

  Click **Accept current setup - Vertex #<ID>** to return to the **Methods on points of <line name>** menu, and continue adding new control points as necessary.

Click the **Upper level menu** button to return to the **Method along <line name>** menu.

v. If you selected **line deformation** for the method in step 3.(b)iii. and you added a control point to the line, then you have the option of removing a control point. To begin, click **Remove control points** to open the **Delete control points of <line name>** menu. Then, select the node you no longer want as a control point, and click the **Delete** button. After you approve the deletion in the panel that opens, the control point will be removed.

Important

Only added control points can be removed. A panel will open with a warning if you attempt to delete a boundary node, and no action will be taken.

When you have finished deleting control points on the line, click **Upper level menu** to return to the **Method along <line name>** menu.

vi. Click **Upper level menu** to return to the **Methods on lines of domain <domain name>** menu, and repeat steps 3.(b)i.–3.(b)v. for each of the remaining lines.

vii. Click **Upper level menu** repeatedly until you return to the **Preprocessor: mesh deformation** menu.

c. Define the method of motion for points. Note that this step is only necessary if you did not specify all of the boundary conditions when defining the deformation of the lines, or if you want to modify an existing point setup.

Methods on points

The **Methods on points of domain <domain name>** menu will open, listing all of the nodes and their current methods. No methods are applied by default, on the nodes, and so each node for which this is true is listed as being a **free point**. A node may be listed with a motion method as a result of your boundary condition definitions for the deformation of lines, or if a method has been imposed on it because an adjacent domain already has a deformation method defined. In the latter case, you cannot modify the imposed method.

i. Select a node in the list and click the **Modify** button to enter the **Modify control point** menu.

ii. Select the appropriate method for the node. You have the following choices:

- To allow the position of the node to be computed by the remeshing method defined on the domain of higher dimension, select **free point**.

  Select method: free point

  Click **Accept current setup - Vertex #<ID>** to return to the **Methods on points of domain <domain name>** menu.

- To specify only the direction of the displacement of the node, select **direction imposed**.

  Select method: direction imposed

  Note that the amplitude will be determined by the deformation method applied on the associated line.

After you have selected this option, the **Modify direction of movement** item will become available. Click this item and revise the default values in the panels that open, as necessary.

Click **Accept current setup - Vertex #<ID>** to return to the **Methods on points of domain <domain name>** menu.

- To specify the direction and the amplitude of the displacement of the node, select **direction and amplitude imposed**.

Select method: direction and amplitude imposed

After you have selected this option, the **Modify direction of movement** and **Modify amplitude of movement** items will become available. Click these items and revise the default values in the panels that open, as necessary.

> **Important**
>
> You can use an evolution function or a user-defined template to define the amplitude. Simply click the **EVOL** or **UPDT** button under the ANSYS POLYDATA menu bar prior to clicking **Modify amplitude of movement**. After you have approved the amplitude value in the panel, the appropriate menus will open. Click **Upper level menu** to return to the **Modify control point** menu.

Click **Accept current setup - Vertex #<ID>** to return to the **Methods on points of domain <domain name>** menu.

- To specify only the amplitude of the displacement of the node, select **amplitude imposed**.

Select method: amplitude imposed

Note that the direction will be determined by the deformation method applied on the associated line.

After you have selected this option, the **Modify amplitude of movement** item will become available. Click this item and revise the default value in the panel that opens, as necessary.

> **Important**
>
> You can use an evolution function or a user-defined template to define the amplitude. Simply click the **EVOL** or **UPDT** button under the ANSYS POLYDATA menu bar prior to clicking **Modify amplitude of movement**. After you have approved the amplitude value in the panel, the appropriate menus will open. Click **Upper level menu** to return to the **Modify control point** menu.

Click **Accept current setup - Vertex #<ID>** to return to the **Methods on points of domain <domain name>** menu.

- When a node remains fixed at its original coordinates because it is part of a surface or line that is fixed, the only available option is **fixed point**.

Select method: fixed point imposed

Note that this option is not available for nodes that are part of a surface or line that is not fixed. In such circumstances the only way to fix the node is by selecting **amplitude imposed** and specifying an amplitude of 0.

Click **Accept current setup - Vertex #<ID>** to return to the **Methods on points of domain <domain name>** menu.

iii. Repeat steps 3.(c)i. and 3.(c)ii. for each of the remaining points.

iv. Click **Upper level menu** to return to the **Preprocessor: mesh deformation** menu.

d. If the associated task is non-steady, define checks for element distortion by clicking **Element distortion check** in the **Preprocessor: mesh deformation** menu.

Element distortion check

The **Element distortion check** menu will open, allowing you to make changes to the criteria ANSYS POLYFLOW will use when evaluating the element distortion that results from the remeshing. The default settings are appropriate for most cases.

If any of the elements reach the specified distortion limits during the simulation, the manner in which ANSYS POLYFLOW will proceed will depend on whether you selected **Stop if distortion limits exceeded** or **Warning if distortion limits exceeded**. If you selected the former, the current calculation is halted momentarily and adjustments are made, so that the global simulation can proceed. For example, a time-dependent simulation will halt the current time step and then proceed with a time step that is half of the original value. If you selected **Warning if distortion limits exceeded**, the simulation will continue unaltered but you will receive a warning.

When you are satisfied with the distortion check criteria, click **Accept the current setup**. ANSYS POLYDATA will perform an immediate evaluation of the initial mesh quality in the domain of the current sub-task. If the initial mesh exceeds the criteria in any way, you will receive a warning. You can then perform one of the following options:

- Save your session, go back into your meshing tool, and create a better initial mesh.
- Modify the distortion check criteria such that the initial mesh is within the limits.
- Make sure that **Warning if distortion limits exceeded** is selected and proceed with the simulation.

Creating a better initial mesh is the best option of those described previously, as it is not a good idea to start with a highly distorted initial mesh. The other options should only be used for the sake of interest, and even then, only if you know in advance that the mesh deformations you defined will not yield elements that are too distorted.

e. Define how the modulus E is calculated for the pseudo-elastic equation (*Equation 32–3* (p. 621)), and therefore specify whether and in what manner the mesh element quality indicators are factored into the elastic remeshing.

Numerical parameters for elastic remeshing

The **Numerical parameters for elastic remeshing** menu will open, where you have the following choices:

- To specify that the mesh is treated as an isotropic homogeneous pseudo-elastic medium, select **Select modulus: constant**. This option is only recommended for simple deformations, as it does not account for the quality of the mesh elements.

  Select modulus: constant

- To specify that the modulus is a function of the element quality, such that the elements with worse quality at a given iteration of the solution are stiffer than those with better quality, select **Select modulus: function of quality indicator Q**. This option corresponds to *Equation 32–8* (p. 625), and is recommended for complex geometries.

  Select modulus: function of quality indicator Q

  After you have selected this option, the **Modify modulus exponent n = <current value>** item will become available. The exponent defined by this item corresponds to n in *Equation 32–8* (p. 625). Click this item and revise the default value in the panel that opens, as necessary.

- To specify that the modulus is a function of the element quality, such that the stiffness of an element increases when it becomes more distorted relative to its initial undeformed state (and vice versa), select **Select modulus: function of quality indicator ratio Q/Q0**. This option corresponds to *Equation 32–7* (p. 625), and is recommended for complex geometries. This option is selected by default, as it is most likely to provide suitable results.

  Select modulus: function of quality indicator ratio Q/Q0

  After you have selected this option, the **Modify modulus exponent n = <current value>** item will become available. The exponent defined by this item corresponds to n in *Equation 32–7* (p. 625). Click this item and revise the default value in the panel that opens, as necessary.

Click **Upper level menu** to return to the **Preprocessor: mesh deformation** menu.

4. Click the **Upper level menu** button to return to the **Task** menu, and continue defining the task.

Chapter 33: Optimization

This chapter describes ANSYS POLYFLOW's internal optimizer, which allows you to find the ideal geometry of the die shape, material data parameters, inlet flow rates, boundary conditions parameters, and/or sub-model parameters in order to achieve your design objectives.

This chapter is divided into the following sections:

33.1. Introduction

In order to use ANSYS POLYFLOW's internal optimizer, you must first define your mesh deformation problem as described in *Die Shape Parameterization* (p. 619) (if you are optimizing the die shape) and set up the flow problem in ANSYS POLYDATA. Then you must define the following items for the optimization:

- objective functions

  The objective functions are the aspects of the simulation that you want to minimize or maximize. Examples of such aspects include the flow balance at the die exit, the pressure drop through the die land, the difference between the radius of the free jet at a given position with respect to a prescribed value, the temperature at a point, swelling, and viscous heating.

- design variables

  The design variables are quantities whose values you allow to vary in order to find the optimum solution. When optimizing the die shape (see *Figure 32.1* (p. 620) for an illustration), the amplitudes of displacement you assigned on the topological entities when defining the mesh deformation can be design variables. Besides the displacement amplitudes, you can also define the design variables to be the flow rates in different inlet sections, material data parameters, and certain boundary condition and sub-model parameters. For a complete list of the quantities that can be designated as design variables, see *Design Variables* (p. 661).

- constraints

  The constraints define the bounds and in/equalities that the objective functions, design variables, and other solution functions must observe.

After the previous items have been defined, the internal optimizer can then find the values of the design variables that minimize all the objectives functions, while respecting the set of constraints. In many ways, the optimizer functions as an evolution scheme (see *Evolution* (p. 517)), but with a different goal: optimization intends to minimize an objective function, rather than reach the final value of an evolution function.

During the optimization, ANSYS POLYFLOW has to perform two CPU-consuming tasks: the evaluation of the solution (including the cost function), and the evaluation of the sensitivities of the cost function with respect to the design parameters. ANSYS POLYFLOW will perform an optimization loop in which it computes the new values for the design variables (e.g., the shape of the geometry), the flow solution, and the objective and constraint functions. When it is required by the optimizer, ANSYS POLYFLOW will also compute the sensitivities of the solution with respect to the design variables.

In order to reduce the CPU time, the optimizer does not search for a global optimum. The assumption is that the number of minima within the range of the design variables is small; when this condition is met, a global optimum is generally found, but it is possible that it is instead a local minimum. In the case of die shape optimization, the number of minima is reduced when the initial geometry is close to the optimized geometry, so you should start with your best estimation.

In this chapter, *Theory* (p. 652) presents the method implemented for the optimizer under constraints, independently of any mesh deformation and the flow problem. *Optimization in ANSYS POLYFLOW* (p. 661) describes the links that exist between any defined mesh deformation, the flow problem, and the optimizer. *Problem Setup* (p. 673) explains how to define an optimization problem in ANSYS POLYDATA, and *Files and Output for Optimization* (p. 689) describes the output of the optimization. To view examples that demonstrate the setup and results of optimization problems, see Example 95, 98, and 100 in the ANSYS POLYFLOW Examples Manual.

33.2. Theory

This theoretical section presents the optimization method implemented in ANSYS POLYFLOW. First, the constrained optimization problem is defined, and then details of the algorithm used to solve such a problem are provided.

33.2.1. Constrained Optimization

The standard form of a nonlinear constrained optimization problem consists of one objective function:

$$\text{Minimize } F\,(\mathbf{S}) \tag{33–1}$$

which is subject to the following constraints:

- general linear and nonlinear inequality constraints:

$$g_j\,(\mathbf{S}) \le 0 \quad j = 1\,,\ n_I \tag{33–2}$$

- general linear and nonlinear equality constraints:

$$h_k\,(\mathbf{S}) = 0 \quad k = 1\,,\ n_E \tag{33–3}$$

- side constraints:

$$S_i^L \leq S_i \leq S_i^U \quad i = 1\,,\; n_D \tag{33–4}$$

In the previous equations, $\mathbf{S}$ represents a set of design variables, and n_E and n_I are the number of inequality and equality constraints, respectively. S_i is a component of one of a number (n_D) of design variables, with lower and upper bounds S_i^L and S_i^U.

The objective and constraint functions may be linear or nonlinear. It is possible that there are no inequality or equality constraints defined for a problem (i.e., it may be unconstrained). On the other hand, all problems need to have defined side constraints.

If the optimization problem is a maximization of $F\,(\mathbf{S})$, ANSYS POLYFLOW will automatically change it to minimize $-F\,(\mathbf{S})$. Similarly, ANSYS POLYFLOW will automatically change inequality constraints of the form $g_{\mathbf{j}}\,(S) \;\geq 0$ to be $-g_j\,(S) \;\leq 0$.

It is assumed that the set of design variables $\mathbf{S}$ is continuous between the lower and the upper bounds. The number of design variables must be reasonably small (i.e., less than 20).

The optimizer has only one objective function ($F\,(\mathbf{S})$). If you want to address multiple objective functions, you must either combine them into a single objective function, or optimize them individually (see *Objective Functions* (p. 667) for further details). It is assumed that $F\,(\mathbf{S})$ is normalized, which means the initial value of $F\,(\mathbf{S}_0)$ is 1.

33.2.2. Solving the Optimization Problem

In ANSYS POLYFLOW, the original constrained problem is converted so that it can be solved as a series of unconstrained problems. The basic approach is to

$$\text{Minimize } \Phi\,(\mathbf{S}) \;= F\,(\mathbf{S}) \;+ P\,(\mathbf{S}) \tag{33–5}$$

where $\Phi\,(\mathbf{S})$ is called the pseudo-objective function. $P\,(\mathbf{S})$ is the penalty term, which depends on the method being used.

The penalty term $P\,(\mathbf{S})$ is calculated in ANSYS POLYFLOW using an iterative process called the augmented Lagrange multiplier (ALM) method. In this method, the unconstrained problems are solved by the Fletcher-Reeves (FR) method, which itself is an iterative process. The FR method uses a line search (LS) technique (yet another iterative process) to find a minimum in a given search direction. Each of these methods are described in the sections that follow.

33.2.2.1. The Augmented Lagrange Multiplier (ALM) Method

The following section addresses an optimization problem like that described in *Constrained Optimization* (p. 652). In order to take into account the constraints for an iteration, the objective function and the constraints are combined into a single augmented Lagrange function:

$$A\left(\mathbf{S},\boldsymbol{\lambda}_p,r_p\right) = 100 + F(\mathbf{S}) + \sum_{j=1}^{n_I}\left[\lambda_{j,p}\Psi_{j,p} + 2r_p\Psi_{j,p}^2\right] + \sum_{k=1}^{n_E}\left\{\lambda_{k+n_I,p}h_k(\mathbf{S}) + 2r_p\left[h_k(\mathbf{S})\right]^2\right\} \tag{33–6}$$

where p is the iteration index of the ALM method, $\boldsymbol{\lambda}_p$ is the vector of the Lagrange multipliers of dimension $n_I + n_E$, r_p is an adjustable penalty coefficient, and $\Psi_{j,p}$ is given by:

$$\Psi_{j,p} = \max\left[g_j(\mathbf{S}) \ , \frac{-\lambda_{j,p}}{2r_p}\right] \tag{33–7}$$

For each iteration of the augmented Lagrange multiplier (ALM) method, the Lagrange multipliers and the penalty coefficient are updated. The formulas for updating the Lagrange multipliers are as follows:

- for the inequality constraints:

$$\lambda_{j,p+1} = \lambda_{j,p} + 2r_p\Psi_{j,p} \quad j = 1, n_I \tag{33–8}$$

- for the equality constraints:

$$\lambda_{k+n_I,p+1} = \lambda_{k+n_I,p} + 2r_p h_k(\mathbf{S}) \quad k = 1, n_E \tag{33–9}$$

with $\boldsymbol{\lambda}_0 = \mathbf{0}$.

The formula for updating the penalty coefficient is:

$$r_{p+1} = KR * r_p \tag{33–10}$$

where $r_{p+1} \le r_{MAX}$. KR is a multiplier that is used to increase the penalty coefficient, and r_{MAX} represents the largest value of the penalty coefficient. By default $r_0 = 1$, $KR = 2$, and $r_{MAX} = 500,000$.

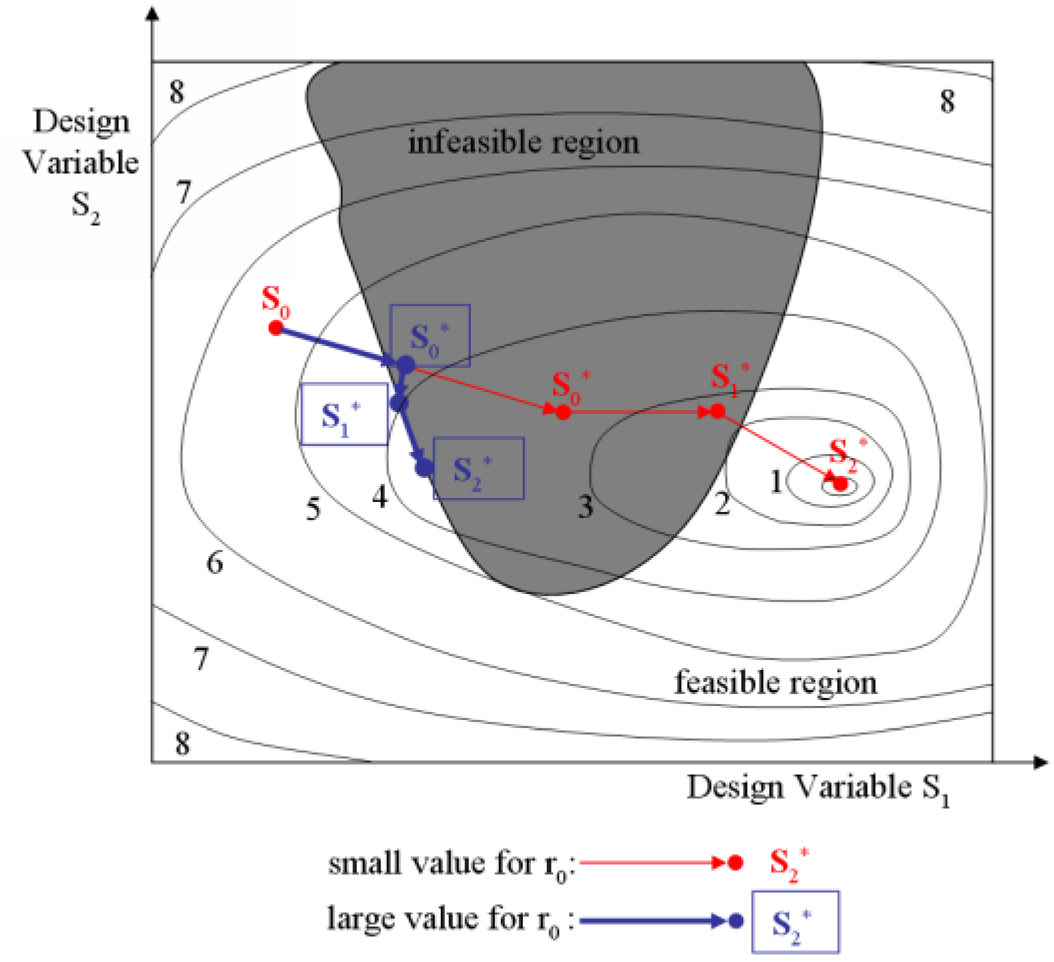
The algorithm for the ALM method is as follows:

1. With $p = 0$, define a tolerance ε and initial values for point $\mathbf{S}_0$ and the penalty coefficient r_0. Also, set the initial Lagrange multipliers λ_0 equal to $\mathbf{0}$.
2. Perform unconstrained optimization using the Fletcher-Reeves (FR) method on the augmented Lagrangian function $A\left(\mathbf{S},\boldsymbol{\lambda}_p,r_p\right)$ to get $\mathbf{S}_p^*$.
3. Update the Lagrange multiplier using *Equation 33–8* (p. 654) and *Equation 33–9* (p. 654).
4. Increase the penalty coefficient using *Equation 33–10* (p. 654).

5. Check for convergence. If convergence is reached, then the process is complete. Otherwise, set $p = p + 1$, set $\mathbf{S}_p = \mathbf{S}_{p-1}{}^*$, and return to step 2.

The initial value of the penalty coefficient has an impact on the behavior of the algorithm. When r_0 is set to a small value, a violation of the constraints does not greatly impact the pseudo-objective function for the first iterations, because the objective function $F(\mathbf{S})$ in *Equation 33–5* (p. 653) decreases faster than the penalty term $P(\mathbf{S})$ increases. On the contrary, when r_0 is set to a large value, a violation of the constraints will dominate the pseudo-objective function, so the solution complies better with the constraints. When the penalty coefficient is small, it is possible to cross over a region of solutions that are infeasible due to their violation of the constraints, and eventually reach an optimum solution in a feasible region (where the constraints are satisfied). Whereas with a large penalty coefficient value, it is impossible to penetrate deeply or cross over an infeasible region. *Figure 33.1* (p. 655) illustrates both situations. The thin, red line corresponds to a case where the initial penalty coefficient is small. For the first iteration, the solution is located deep in the infeasible region, and during the next iteration the solution crosses over the infeasible region. Eventually the final solution corresponds to the global optimum. The thick, blue line corresponds to a larger value for initial penalty coefficient. For the first iteration, the solution is located close to the boundary of the infeasible region. During the next iteration the solution goes back on the same side as the initial solution, and eventually reaches a final solution at a local minimum.

Figure 33.1 The Influence of the Initial Value of the Penalty Coefficient



There are two criteria used to check the convergence of the method. The first is related to the objective function:

$$|F\left(\mathbf{S}_p{}^*\right) - F\left(\mathbf{S}_p\right)| \le 100\varepsilon_O \quad \text{or} \quad |F\left(\mathbf{S}_p{}^*\right)| \le 10^{-14} \qquad \textbf{(33–11)}$$

The second criterion is related to the convergence of the constraints, as shown in the following equations:

$$\frac{\Psi_{j,p}}{g_{j,0}\left(\mathbf{S}_0\right)} \le \varepsilon_C \quad j = 1, n_I \qquad \textbf{(33–12)}$$

and

$$\left|\frac{h_{k,p}\left(\mathbf{S}_p\right)}{h_{k,0}\left(\mathbf{S}_0\right)}\right| \le \varepsilon_C \quad k = 1, n_E \qquad \textbf{(33–13)}$$

The default value for ε_C is 10^{-5}.

Note that there is a difference between the convergence and the satisfaction of the constraints. The criteria expressed in *Equation 33–12* (p. 656) and *Equation 33–13* (p. 656) are measuring convergence, that is, whether the value of the constraint function has been dramatically reduced. On the other hand, the constraint criteria shown in *Equation 33–2* (p. 652) and *Equation 33–3* (p. 652) are related to the satisfaction of the constraint. Therefore, it is possible to have a converged constraint that is not well satisfied.

33.2.2.2. The Fletcher-Reeves (FR) Method

In the second step of the algorithm for the ALM method, the function $A\left(\mathbf{S}, \boldsymbol{\lambda}_p, r_p\right)$ is minimized to get $\mathbf{S}_p^*$ using the Fletcher-Reeves (FR) method. The FR method belongs to the conjugate gradient methods to minimize the function $\Phi\left(\mathbf{S}\right)$ without constraints. In ANSYS POLYFLOW, $\Phi\left(\mathbf{S}\right) = A\left(\mathbf{S}, \boldsymbol{\lambda}_p, r_p\right)$.

The FR method involves multiple iterations. During an iteration, the intent is to find a minimum along a given direction in which the function $\Phi\left(\mathbf{S}\right)$ is decreasing. At the end of the iteration, a new direction is calculated for the next iteration. This process is repeated until convergence is reached.

The algorithm for the FR method is as follows:

1. With the iteration index of the FR method (q) set to 0 and a starting point $\mathbf{S}_{p,q}$ equal to $\mathbf{S}_p$, define a tolerance ε. Also, calculate an initial direction $d_q = -\mathbf{g}_q$, where $\mathbf{g}$ is the column vector of gradients of the objective function:

$$\mathbf{g} = \frac{\partial \Phi\ (\mathbf{S})}{\partial \mathbf{S}} \quad \textbf{(33–14)}$$

2. Using the direction $\mathbf{d}_q$, find the new point $\mathbf{S}_{p,q+1}$ using the following equation:

   $$\mathbf{S}_{p,q+1} = \mathbf{S}_{p,q} + \alpha_q{}^* \mathbf{d}_q \quad \textbf{(33–15)}$$

   where the coefficient $\alpha_q{}^*$ is found using the line search method.

3. Calculate the new conjugate gradient direction $\mathbf{d}_{q+1}$, according to:

   $$\mathbf{d}_{q+1} = -\mathbf{g}_{q+1} + \beta_{q+1} \mathbf{d}_q \quad \textbf{(33–16)}$$

   where

   $$\beta_{q+1} = \frac{\left\| \mathbf{g}_{q+1} \right\|}{\left\| \mathbf{g}_q \right\|} \quad \textbf{(33–17)}$$

   and $\mathbf{g}_{q+1}$ is the gradient of the objective function at a new point:

   $$\mathbf{g}_{q+1} = \frac{\partial \Phi \left(\mathbf{S}_{p,q+1} \right)}{\partial \mathbf{S}} \quad \textbf{(33–18)}$$

4. Check for convergence. If convergence is reached, then the FR method is stopped and $\mathbf{S}_p{}^*$ is set to $\mathbf{S}_{p,q+1}$ for iteration p of the ALM method. Otherwise, set $q = q + 1$ and return to step 2 of the FR method.

The convergence criterion for the FR method is the following:

$$\frac{\left| \Phi \left(\mathbf{S}_{p,q+1} \right) - \Phi \left(\mathbf{S}_{p,q} \right) \right|}{\Delta} < \varepsilon \Phi \left(\mathbf{S}_{p,0} \right) \ or\ \left| \Phi \left(\mathbf{S}_{p,q+1} \right) - \Phi \left(\mathbf{S}_{p,q} \right) \right| < 10^{-14} \quad \textbf{(33–19)}$$

with

$$\Delta = \sum_{i=0}^{q} \left(\Phi \left(\mathbf{S}_{p,i} \right) - \Phi \left(\mathbf{S}_{p,i+1} \right) \right) \quad \textbf{(33–20)}$$

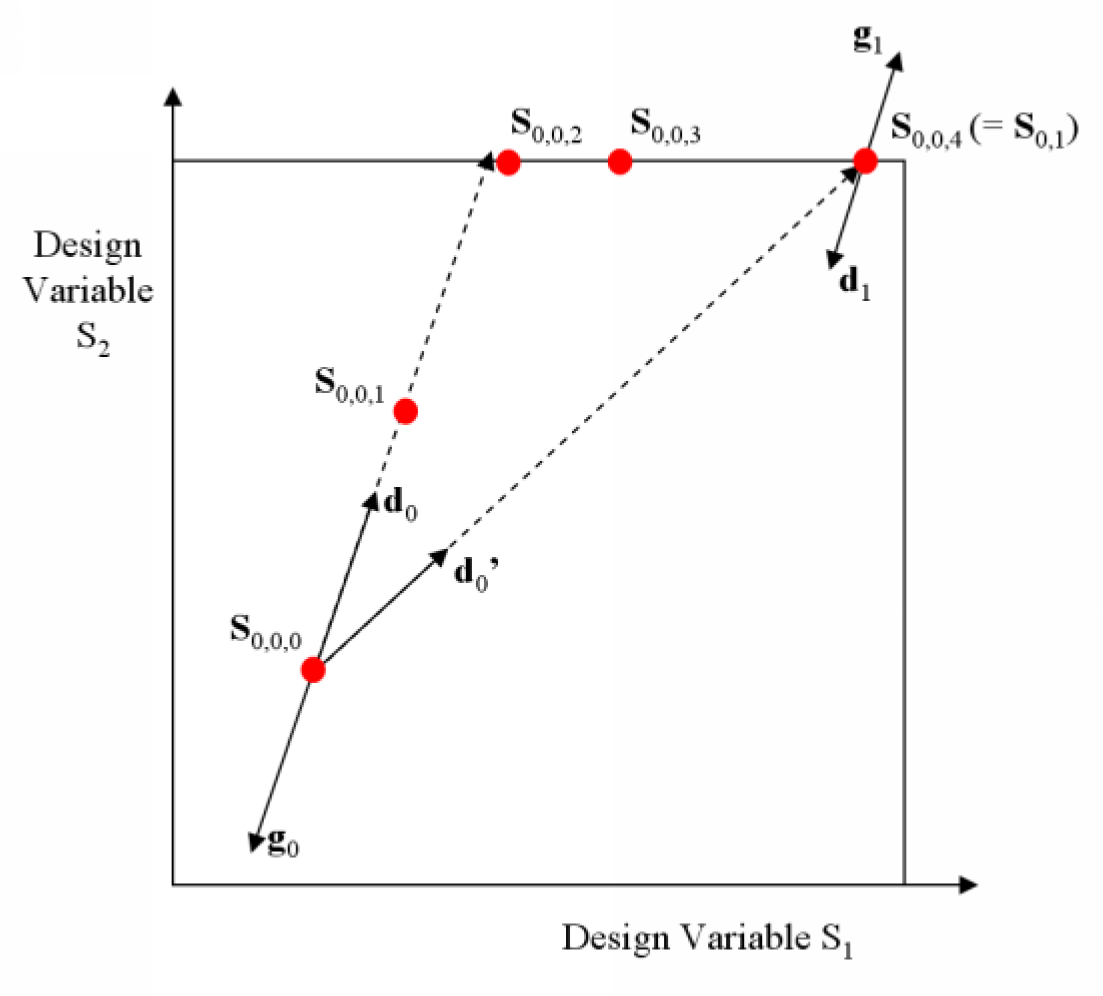
The direction is not always updated using *Equation 33–16* (p. 657). Sometimes, $\mathbf{d}_{q+1}$ is reset to

$$\mathbf{d}_{q+1} = -\mathbf{g}_{q+1} \tag{33–21}$$

There are two cases when *Equation 33–21* (p. 658) is used:

- The efficiency of *Equation 33–16* (p. 657) and *Equation 33–17* (p. 657) is less and less interesting as the number of iterations of the FR method increases. Hence, a maximum number of iterations is specified, and when the simulation reaches this point the direction is set to be equal to the opposite of the gradient.
- When a design variable reaches its limit, it is possible that the line search method will "move" the points along this limit. Such a situation is illustrated in *Figure 33.2* (p. 658) with the points $\mathbf{S}_{p,q,t}$, where t is the iteration index of the line search. Because the points $\mathbf{S}_{0,0,2}$, $\mathbf{S}_{0,0,3}$, and $\mathbf{S}_{0,0,4}$ are located along the limit of design variable S_2, the initial direction $\mathbf{d}_0$ and the direction $\mathbf{d}_0{}'$ between the initial and the final points of the line search are not the same. In such circumstances, the direction is reset to the opposite of the gradient.

Figure 33.2 Resetting the Direction to the Opposite of the Gradient



33.2.2.3. The Line Search (LS) Method

In the second step of the algorithm for the FR method, it is necessary to find a value for the coefficient α_q^* that minimizes $\Phi\left(\mathbf{S}_{p,q}+\alpha_q^*\mathbf{d}_q\right)$. In order to do this, the line search (LS) method performs several calculations of $\Phi_t=\Phi\ (\alpha_t)\ =\Phi\left(\mathbf{S}_{p,q}+\alpha_t\mathbf{d}_q\right)$, during which the target value α_t is varied. Note that the subscript t refers to the iteration index of the LS method, whereas the subscripts p and q refer to the iteration indices of the ALM and FR method, respectively.

Figure 33.3 An Example of the LS Algorithm

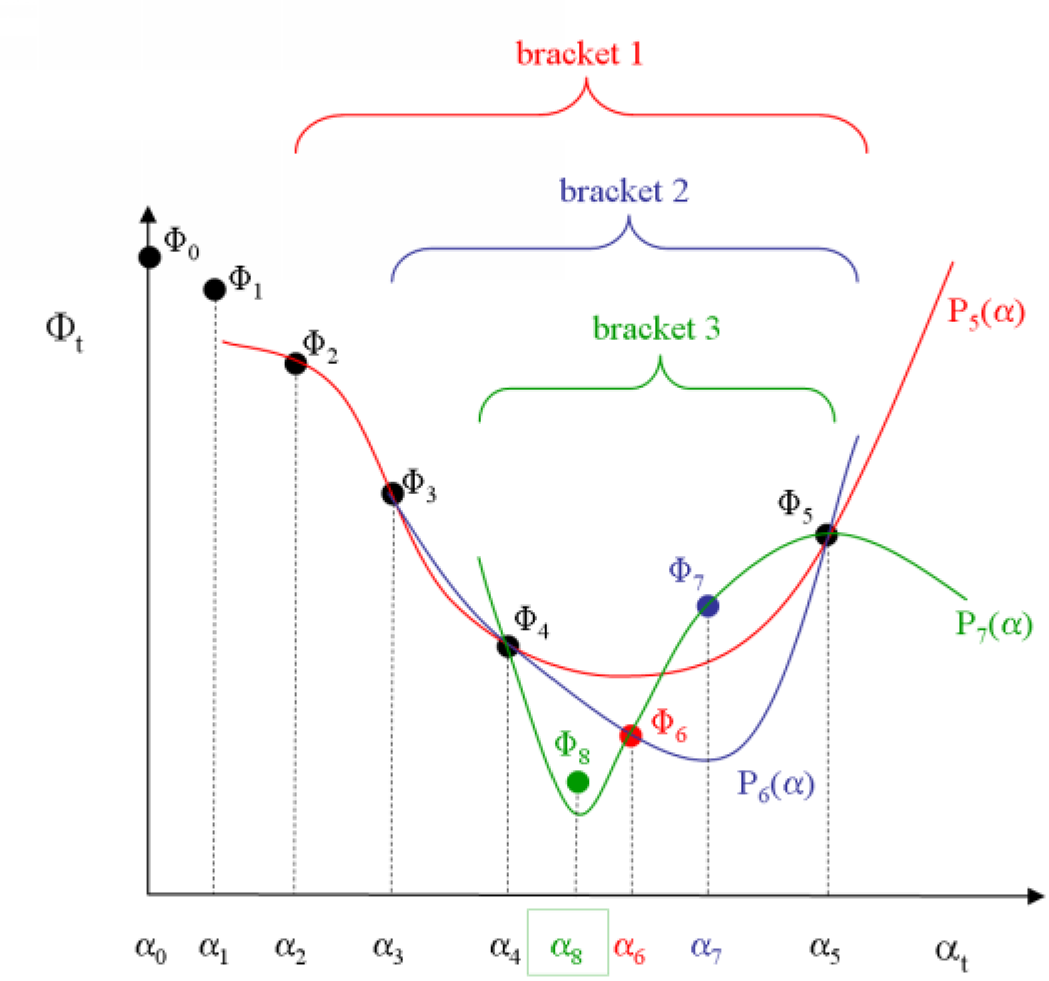


Figure 33.3 (p. 659) illustrates an example of the LS method algorithm. Starting with $\alpha_0=0$, an initial value $\Phi_0=\Phi\left(\mathbf{S}_{p,q}\right)$ is calculated. The slope of the function at this point is negative, as the direction $\mathbf{d}_q$ was specifically defined to follow a decreasing gradient. A target value α_1 is then used to calculate Φ_1. The initial target value α_1 is based upon the distance of the current point from the bounds of the design variable, the value and the slope of the function Φ, and sometimes the value of α_{q-1}^* (the coefficient of the previous FR method iteration). As shown in Figure 33.3 (p. 659), Φ_1 is smaller than Φ_0, so α is increased to the new target value α_2, and Φ_2 is calculated. This process is repeated to generate the successive target values α_3, α_4, and α_5. The process is interrupted when $\Phi_t\geq\Phi_{t-1}$ (i.e., when

$t = 5$). At this point, the minimum is assumed to be located somewhere between α_2 and α_5. As a first attempt at finding it, a new target value α_6 is generated that minimizes $P_5(\alpha)$. $P_5(\alpha)$ is an interpolation of the target values in bracket 1, and can be expressed thusly:

$$\begin{aligned} P_5(\alpha_2) &= \Phi_2; \\ P_5(\alpha_3) &= \Phi_3; \\ P_5(\alpha_4) &= \Phi_4; \\ P_5(\alpha_5) &= \Phi_5 . \end{aligned} \tag{33–22}$$

Having found α_6, it is then possible to calculate Φ_6. In a similar fashion, the target values in bracket 2 are used to calculate $P_6(\alpha)$, α_7, and Φ_7, and the target values in bracket 3 are used to calculate $P_7(\alpha)$, α_8, and Φ_8. The line search ends at this point because convergence is reached.

The general algorithm for the LS method can be stated as follows:

1. With the iteration index of the LS method (t) equal to 0, set $\alpha_t = 0$ and calculate Φ_t using the following relation:

   $$\Phi_t = \Phi(\alpha_t) = \Phi\left(\mathbf{S}_{p,q} + \alpha_t \mathbf{d}_q\right) \tag{33–23}$$

   where p and q are the current iteration indices of the ALM and FR methods, respectively.
2. Set $t = t + 1$ and generate a value for α_t such that $\alpha_t > \alpha_{t-1}$. Then calculate Φ_t using *Equation 33–23* (p. 660).
3. If $\Phi_t < \Phi_{t-1}$, return to step 2. Otherwise, proceed to step 4.
4. Generate a function $P_t(\alpha)$ that interpolates the last four values of Φ (use less values if $t < 3$). Then set $t = t + 1$ and set α_t equal to the minimum value of $P_{t-1}(\alpha)$.
5. Calculate Φ_t using *Equation 33–23* (p. 660).
6. Check for convergence. If convergence is reached, then the LS method is stopped and $\alpha_q{}^*$ is set to α_t for iteration q of the FR method. Otherwise, return to step 4 of the LS method.

The process is deemed to be converged if the number of iterations has reached a predefined “maximum number of iterations to reach minimum", if $\alpha_t = 0$, or if two successive solutions are very close.

The LS method algorithm is CPU intensive, because it corresponds to the computation of the flow solution. For this reason, it is more efficient to perform a small number of iterations (with a resulting accuracy that is lower) for the LS method. Hence, the default value for “the maximum number of iterations to reach minimum" is $t + 3$, where t is the iteration number at the beginning of step 4 of the LS method algorithm.

If the flow problem is governed by nonlinear equations, the LS method can fail when trying to calculate Φ_t. Under such circumstances, the value of α is reduced and subiterations are used to reach a target value, resulting in intermediate values of the function Φ. If a target value cannot be reached, the last successful intermediate value is assumed as the target value, and the LS method continues.

When it becomes necessary to calculate intermediate values of Φ, the LS method proceeds to find a minimum Φ_t that is based on the target values only. A comparison is then made to see if any of the intermediate values of Φ are smaller than Φ_t; if this is the case, then the associated intermediate value of α is taken as the solution of the LS method.

33.3. Optimization in ANSYS POLYFLOW

Optimization introduces the concepts of design variables, objective functions and constraints. Further details about these concepts are provided in this section, along with explanations of how the quantities involved in mesh deformation and in the flow problem are linked to the optimizer.

33.3.1. Design Variables

In order to designate that a quantity is a design variable, you must first flag the quantity as a template entry in the same manner as when setting up a user-defined template (UDT). The following quantities are eligible to be design variables:

- amplitude of displacement

  When setting up mesh deformation, you can define the displacement of geometric entities (i.e., volumes, surfaces, lines, and points) by specifying a direction and amplitude. For the purposes of die shape optimization, however, only the amplitude of the displacement is eligible as a design variable. By default, the initial value of a given design variable will be the value specified for the amplitude of the displacement during the mesh deformation setup.

- the inlet flow rates
- material data parameters

  The viscosity, the relaxation times, the Arrhenius function parameters, and other viscoelastic parameters can be designated as design variables. Defining a material data parameter as a design variable is useful when you are employing a reverse engineering approach (see Example 100 in the ANSYS POLYFLOW Examples Manual).

- boundary condition parameters for a heat conduction problem sub-task

  The quantities that can be design variables are:

  - the constant value for an imposed temperature
  - the parameters for an imposed flux density (q_c, α, T_α, σ, T_σ, and T_0 from *Equation 13–7* (p. 284))

- slip condition parameters

  The quantities that can be design variables are:

  - the generalized Navier’s law parameters (F_{slip} and e_{slip} from *Equation 8–1* (p. 176))
  - the threshold law parameters (F_{slip}, $F_{slip,2}$, and y_c from *Equation 8–2* (p. 176))
  - the asymptotic law parameters (F_{slip} and v_c from *Equation 8–3* (p. 177))
  - the approximate Arrhenius / Arrhenius law parameters, if the slip law is temperature dependent (α, T_α, and T_0 from *Equation 10–17* (p. 213) and *Equation 10–19* (p. 214))

  Note that the velocity of the wall ($\mathbf{v}_{wall}$ in *Equation 8–1* (p. 176), *Equation 8–2* (p. 176), and *Equation 8–3* (p. 177)) *cannot* be a design variable.

- concentration boundary condition parameters for a transport of species sub-task

  The quantities that can be design variables are:

  – the constant value of an imposed mass fraction

  – the constant value of an imposed flux density

- the constant value of an imposed potential boundary condition parameters for a potential problem sub-task
- the constant value of the porous medium thickness (h in *Equation 8–9* (p. 194)) for a porous jump model sub-model
- the shell thickness for a heat conduction problem sub-model

There may be circumstances in which you want to link design variables, so that the value of one is imposed on another. When this is the case, you can force an equality between a “proxy" (S_{Pr}) and a “master" (S_M) design variable, such that:

$$S_{Pr}=S_M \tag{33–24}$$

In this way, the proxy design variables are not seen by the optimizer either as design variables or as equality constraints of the type $h_k\,(\mathbf{S})\ =0$. Consequently, the size and the complexity of the optimization problem is reduced.

Typically, you should link design variables that are of the same kind. For example, you could impose the same amplitude of displacement on several topological entities that are not connected, or the same flow rate on several inlets. While it is possible in ANSYS POLYFLOW to link design variables of different kinds (i.e., you could impose the value of an amplitude of displacement on the rate of a flow inlet), such linkages should be avoided or used with care.

Note that a given master design variable may have more than one proxy. Also, you must not designate the same design variable as both a master and a proxy.

After you have flagged a quantity as a template entry, you must define the following design variable parameters:

- name

  Short names are recommended for design variables, for the sake of readability in the output files or in ANSYS POLYDIAG.

- upper bound: S_i^U
- lower bound: S_i^L
- initial value
- (optional) link with a master design variable through an equality

33.3.2. Extractors

In optimization, the fields used in the calculation of an objective or constraint function must be in the form of a single scalar value. However, the fields computed by ANSYS POLYFLOW rarely meet this criteria. For this reason, you must define “extractors", that is, a set of functions that reduces a field in a given domain to a single scalar value (which is then referred to as an “extracted value"). An example of such an extracted value is the minimum of the temperature on wall Ω_1, where Ω_1 is a sub-part of the domain definition of the temperature field. Another example of an extracted value is the average of the velocity norm along the line Ω_2, where Ω_2 is a sub-part of the domain definition of the velocity field.

An extractor operates on the values of a field $\mathbf{X}$ in a domain Ω. The set of values of $\mathbf{X}$ in the domain Ω are noted as $\mathbf{X}_\Omega$. Note that though the tensor type of field can be either a scalar or a vector, for the sake of simplicity it will always be denoted as $\mathbf{X}$.

Typically, the extraction of a value has the following steps:

1. Reduce a non-scalar field on a given domain to a scalar field on the same domain.

   If the field $\mathbf{X}$ is not a scalar, you have to use a restriction function $R\left(\mathbf{X}_\Omega\right)$ to reduce the non-scalar field to a scalar one on the domain Ω. This resulting scalar field is called the restricted field and noted r_f. For example, you can reduce a velocity field by calculating the norm at every node.

2. Extract a single value from the scalar field.

   If the restricted field r_f is not a single value, you must use a function $Q\left(R\left(\mathbf{X}_\Omega\right)\right)$ (which can also be noted as $Q\left(r_f\right)$). This function either selects a value of interest from the restricted field, or combines all the values into a single one.

Thus, the extracted value E that results from the extractor can be written as follows:

$$E = Q(R(\mathbf{X}_\Omega)) \tag{33–25}$$

The following examples illustrate the previous definitions:

- For the extracted value “ $E = \min\left(T \text{ on } \Omega_1\right)$ ":
  - The field X is the temperature on the wall Ω_1: $\mathbf{X}_\Omega = T \text{ on } \Omega_1$.
  - The restriction function R is a simple equality, since the temperature is already a scalar field: $R\left(\mathbf{X}_\Omega\right) = T$
  - The function Q is the minimum: $Q\left(r_f\right) = \min\left(r_f\right)$.
- For the extracted value “ $E = \text{average}\left(\text{norm}\left(\mathbf{v}\right) \text{ on the line } \Omega_2\right)$ ":
  - The field $\mathbf{X}$ is the velocity on the line Ω_2: $\mathbf{X}_\Omega = \mathbf{v} \text{ on } \Omega_2$.
  - The restriction function R is the norm: $R\left(\mathbf{X}_\Omega\right) = \|\mathbf{v}\|$.
  - The function Q is the average: $Q\left(r_f\right) = \overline{r_f}$.

Extractors are automatically generated for the amplitude of displacement defined in a mesh deformation preprocessor, as well as for fields that are already a single scalar value. Otherwise, you must define suitable extractors for the fields used in your optimization functions (see *Problem Setup* (p. 673) for further details).

The available fields $\mathbf{X}$ for the extractors include:

- output fields of the main task
  - coordinates
  - velocity
  - pressure
  - temperature
  - displacement
  - species
- output fields of postprocessors
  - flow rate (see *Flow Rate* (p. 563))
  - flow balance (see *Quantification of Die Balancing* (p. 569) for details)
- fields in a problem with a constrained free jet
  - force applied on the free jet
  - torque applied on the free jet
  - flow rate at the die exit (when a single fluid is extruded)
  - flow balance at the die exit (when a single fluid is extruded)
  - tangential velocity in the die exit (when a single fluid is extruded) $\mathbf{v}_{||} = \mathbf{v} - (\mathbf{v} \cdot \mathbf{n})\,\mathbf{n}$ (in Cartesian coordinates), where $\mathbf{n}$ is the unit vector perpendicular to the die exit section.

The available restriction functions R include:

- the current value

  $$R(\mathbf{X}) = \mathbf{X} \tag{33–26}$$

  This function consists of the current values of $\mathbf{X}$ and is only applied when the field $\mathbf{X}$ is a scalar field (e.g., pressure, temperature). This is selected automatically, since scalar values are detected.

- the norm of a vector

  $$R(\mathbf{X}) = \|\mathbf{X}\| = \sqrt{\sum_{i=1}^{n} (\mathbf{X}_i)^2} \tag{33–27}$$

  This function computes the norm of a vector $\mathbf{X}$ of n components. This is the default selection for a vector field.

- the i^{th} component of a vector

  $$R(\mathbf{X}) = \mathbf{X}_i \tag{33–28}$$

  This function extracts the i^{th} component of a vector.

The available functions Q include:

- the minimum

$$Q\left(r_f\right) = \min\left(r_f\right) \quad \textbf{(33–29)}$$

This function computes the minimum of the restricted field r_f. This function must be used with care, because the location of the minimum can change when the geometry is reconfigured during the optimization process. Such a jump in the location of the minimum may lead to convergence trouble during the optimization.

- the maximum

$$Q\left(r_f\right) = \max\left(r_f\right) \quad \textbf{(33–30)}$$

This function computes the maximum of the restricted field r_f. This function must be used with care, because the location of the maximum can change when the geometry is reconfigured during the optimization process. Such a jump in the location of the maximum may lead to convergence trouble during the optimization.

- the average

$$Q\left(r_f\right) = \overline{r_f} = \frac{\sum_{i=1}^{n} r_{f,i}}{n} \quad \textbf{(33–31)}$$

This function computes the average of the restricted field r_f. This average is based on the nodal values ($r_{f,i}$): the sum of nodal values divided by the number of nodal values (n).

- the value at the closest node (in deformed configuration)

$$Q\left(r_f\right) = r_f\left(N_{x,y,z}\right) \quad \textbf{(33–32)}$$

This function extracts the value of restricted field r_f at node $N_{x,y,z}$, which is whatever node is closest to the coordinates (x,y,z) at a given optimization step. Note that if the point (x,y,z) is located in an area that is subject to mesh deformation, the node closest to the coordinates may change. Such a jump in the location of the closest node may lead to convergence trouble during the optimization.

- the value at the closest node (in the initial configuration)

$$Q\left(r_f\right) = r_f\left(N^0_{x,y,z}\right) \quad \textbf{(33–33)}$$

This function extracts the value of restricted field r_f at node $N^0_{x,y,z}$, which is the node that is initially closest to the coordinates (x,y,z) at the first step of optimization. Even if the point (x,y,z) is located in an area that is subject to mesh deformation, $N^0_{x,y,z}$ will not change during the remaining steps (although the value of r_f may change).

- the weighted average

$$Q\left(r_f\right) = \overline{\langle r_f \rangle} = \frac{\int\limits_{\Omega} r_f d\Omega}{\int\limits_{\Omega} d\Omega} \qquad \textbf{(33–34)}$$

This function computes the weighted average of the restricted field r_f. This weighted average is obtained from the ratio of the integral of r_f over the domain Ω and the integral of 1 over the domain Ω (i.e., the length, area, or volume).

- the integral

$$Q\left(r_f\right) = \langle r_f \rangle = \int\limits_{\Omega} r_f d\Omega \qquad \textbf{(33–35)}$$

This function computes the integral of the restricted field r_f over the domain Ω.

In addition to the functions listed previously, there are special functions for the coordinates. These functions compute the length, area, or volume of a domain that has 1, 2, or 3 dimensions, respectively. This function is only available if the coordinates have been selected for the field.

Note that the function Q is ignored for fields that have only one value. This is the case for fields whose interpolation type is constant over the domain (e.g., flow rate, flow balance), or if the domain from which the value is extracted is a point.

Table 33.1: Extracted Values (p. 666) lists the extracted values by combining the functions *R* and *Q*.

Table 33.1 Extracted Values

		Restriction Functions		
		X (Scalar)	$\|\|\mathbf{X}\|\|$	$\mathbf{X}_i$
Functions for Obtaining the Extracted Value	Minimum	$\min(X)$	$\min(\|\|\mathbf{X}\|\|)$	$\min(\mathbf{X}_i)$
	Maximum	$\max(X)$	$\max(\|\|\mathbf{X}\|\|)$	$\max(\mathbf{X}_i)$
	Average	$\overline{X}$	$\overline{\|\|\mathbf{X}\|\|}$	$\overline{\mathbf{X}_i}$
	Value at Closest Obtaining the Node (Deformed)	$X\left(N_{x,y,z}\right)$	$\|\|\mathbf{X}\left(N_{x,y,z}\right)\|\|$	$\mathbf{X}_i\left(N_{x,y,z}\right)$
	Value at Closest Node (Initially)	$X\left(N^0_{x,y,z}\right)$	$\|\|\mathbf{X}\left(N^0_{x,y,z}\right)\|\|$	$\mathbf{X}_i\left(N^0_{x,y,z}\right)$
	Weighted Average	$\overline{\langle X \rangle}$	$\overline{\langle \|\|\mathbf{X}\|\| \rangle}$	$\overline{\langle \mathbf{X}_i \rangle}$
	Integral	$\langle X \rangle$	$\langle \|\|\mathbf{X}\|\| \rangle$	$\langle \mathbf{X}_i \rangle$
Coordinate Functions		Length		
		Area		

	Volume

The next sections explain how the extracted values can be used when calculating the objective function and constraints.

33.3.3. Objective Functions

Although the internal optimizer of ANSYS POLYFLOW accepts only one objective function $F(\mathbf{S})$ for minimization, it is possible to define multiple objective functions noted $f_m(\mathbf{S})$, and each of these objective functions can be either minimized or maximized. When the goal is to maximize the objective function $f_m(\mathbf{S})$, ANSYS POLYFLOW will redefine it such that $-f_m(\mathbf{S})$ is minimized.

Three schemes are implemented in ANSYS POLYFLOW, which allow for single and multiple objective functions:

- single scheme

  When you define only one objective function for optimization ($f_1(\mathbf{S})$), the following equation is used:

  $$F(\mathbf{S}) = SG\sqrt{\left(\frac{f_1(\mathbf{S})}{f_1(\mathbf{S}_{ini})}\right)^2} \quad \textbf{(33–36)}$$

  where SG is a factor that depends upon the goal of the objective function (1 for minimization and -1 for maximization), and $\mathbf{S}_{ini}$ is the initial value of the design variable. For die shape optimization, this value is set by default to the amplitude of displacement you specified during the mesh deformation setup.

- multi-compromise scheme

  You can use the following equation when you have multiple objective functions to optimize, so that $F(\mathbf{S})$ is obtained by weighting the different objective functions:

  $$F(\mathbf{S}) = \sum_{m=1}^{n_{obj}} SG_m\sqrt{\left(w_m f_m(\mathbf{S})\right)^2} \quad \textbf{(33–37)}$$

  In the previous equation, n_{obj} is the number of objective functions, SG_m is a factor that depends upon the goal of objective function $f_m(\mathbf{S})$ (1 for minimization and -1 for maximization), and w_m is a weight factor associated with the objective function $f_m(\mathbf{S})$. The weight factor transforms $f_m(\mathbf{S})$ into a dimensionless quantity. w_m can be a very small or very large value, as it can be used not only to control the relative weight of an objective function with respect to the other objective functions, but also to ensure that the objective functions are of the same order of magnitude numerically.

- multi-priority scheme

Another approach for handling multiple objective functions involves defining a priority p_m for each objective function $f_m(\mathbf{S})$ (where m varies from 1 to the total number of objective functions). The objective functions are then optimized separately using Equation 33–36 (p. 667), in the order of decreasing priority. Consider the following example, in which p_1 is the highest priority and p_2 is the next highest priority. The first step is to optimize the die shape while only considering the objective function $f_1(\mathbf{S})$, in order to obtain the minimum V_1. Then $f_1(\mathbf{S})$ is removed from the list of objective functions, and is converted into the equality constraint $f_1(\mathbf{S}) - V_1 = 0$. Next, the die shape is optimized while only considering the objective function $f_2(\mathbf{S})$, in order to obtain the minimum V_2. Again, this objective function is removed from the list of objective functions, and is converted into the equality constraint $f_2(\mathbf{S}) - V_2 = 0$. This process continues until all of the objective functions have been optimized. During the optimization process, the number of constraints increases and the number of objective functions to be satisfied decreases accordingly.

Important

When the multi-priority scheme is used, it is a good practice to make sure that the different objective functions have the same order of magnitude numerically, in the manner described at the end of *Constraints* (p. 670).

Since the multi-priority scheme converts the objective functions into constraints, see *Constraints* (p. 670) for recommendations to consider when defining the form of each of the objective functions.

The parameters for an objective function include the following:

- name

  Short names are recommended for objective functions, for the sake of readability in the output files or in ANSYS POLYDIAG.

- priority, ranging between 0 (lowest) and 20 (highest)

  This parameter must be provided when the multi-priority scheme for multiple objective functions is used for optimization. When a single objective function is defined or the multi-compromise scheme is used, the priority is not taken into account.

- goal of the optimization: to minimize or maximize the function
- form of the function to minimize or maximize

  This parameter allows you to use predefined functions to specify the form (noted as F in the following set of equations) of the objective function, which will then be minimized or maximized for the set of design variables $\mathbf{S}$. These predefined functions involve parameters a, b, c, d and e, as well as one or two extracted values which are noted in the following list of equations as X and Y (these latter two variables represent E from Equation 33–25 (p. 663)). Note that the objective function must be defined in such a way so as to have a minimum. The predefined functions include the following:

  – identity function:

$$F = X \tag{33–38}$$

- linear function:

$$F = aX + b \tag{33–39}$$

- square of linear function:

$$F = (aX + b)^2 + c \tag{33–40}$$

- parabolic function:

$$F = aX^2 + bX + c \tag{33–41}$$

- fourth-order polynomial function:

$$F = a + bX + cX^2 + dX^3 + eX^4 \tag{33–42}$$

- square of two-variable function:

$$F = (aX + bY + c)^2 + d \tag{33–43}$$

- two-variable function:

$$F = aX + bY + c \tag{33–44}$$

Which predefined function you select will depend upon the behavior of the extracted values X and Y:

- Consider the case where X exhibits a minimum, such as when it represents the extracted value of the flow balance. Its value is always greater than 0, where 0 corresponds to a perfect die. In this situation, it is recommended that you use *Equation 33–38* (p. 669). Alternatively, you could use the linear function expressed in *Equation 33–39* (p. 669) if you want to shift and/or normalize the objective function.
- On the other hand, consider the case where the field X does not exhibit a minimum, such as when you want to minimize the difference between the temperature (i.e., X is the extracted value of T) and a given value (e.g., 150). In this situation, it is recommended that you use the square of the linear function expressed in *Equation 33–40* (p. 669) $a = 1$, $b = -150$, and $c = 0$. With such parameters, the function will reach a minimum when X equals the ideal temperature of 150.

- extracted value X
- extracted value Y, if required

- parameters a, b, c, d and e, if required

> **Important**
>
> You must choose values for parameters a, b, c, d and e such that $f_m\ (\mathbf{S})$ is always a positive value, otherwise the minimization of *Equation 33–36* (p. 667) or *Equation 33–37* (p. 667) will not necessarily yield correct results.

- weight factor w_m

  This parameter must be provided when the multi-compromise scheme for multiple objective functions is used for optimization. When a single objective function is defined or the multi-priority scheme is used, the weight factor is not taken into account.

33.3.4. Constraints

Although the internal optimizer treats only the inequality and equality constraints of types shown in *Equation 33–2* (p. 652) and *Equation 33–3* (p. 652), respectively, it is possible to take into account constraints of the following type:

$$g_j\ (\mathbf{S})\ \geq 0 \qquad \textbf{(33–45)}$$

This is because ANSYS POLYFLOW can transform them to be:

$$-g_j\ (\mathbf{S})\ \leq 0 \qquad \textbf{(33–46)}$$

The parameters for a constraint include the following:

- name

  Short names are recommended for constraints, for the sake of readability in the output files or in ANSYS POLYDIAG.
- goal of the constraint

  The goal of the constraint can be:

  – an equality of the form $F = 0$, where F represents $h_k\ (\mathbf{S})$ in *Equation 33–3* (p. 652)

  – an inequality of the form $F \leq 0$ or $F \geq 0$, where F represents $g_j\ (\mathbf{S})$ in *Equation 33–2* (p. 652)

- form of the function for the constraint

  The predefined functions available for the objective functions (*Equation 33–38* (p. 669) – *Equation 33–44* (p. 669)) are also valid for the constraints.

  If you want to define a constraint of the type

$$g_i(\mathbf{S}) \leq L \qquad \textbf{(33–47)}$$

where the value of limit L is not zero, you have to select a predefined function with a constant term like parameter b in *Equation 33–39* (p. 669) or parameter a in *Equation 33–44* (p. 669), and set this constant term to be equal to $-L$.

- extracted value X
- extracted value Y, if required
- parameters a, b, c, d and e, if required

The equality constraints can be difficult to satisfy when the initial value of such a constraint ($h_k(\mathbf{S}_{ini})$) is too small (e.g., $< 10^{-2}$). In such circumstances, the number of iterations needed to bring the ALM method to convergence is greatly increased. It is therefore recommended that the values of parameters a, b, c, d, and e in *Equation 33–38* (p. 669) – *Equation 33–44* (p. 669) be selected so as to yield a reasonable value for $h_k(\mathbf{S}_{ini})$. This discussion also impacts optimization problems that minimize multiple objective functions using the multi-priority scheme: as described earlier, when the objective function of a higher priority (e.g., $f_1(\mathbf{S})$) is optimized to yield a minimum value (e.g., V_1), it is then converted into an equality constraint (e.g., $f_1(\mathbf{S}) - V_1 = 0$) for subsequent optimization solutions. Since it would be too difficult to try and predict what the initial values of such converted constraints would be, the best practice you can follow is to define the multiple objective functions such that they are all of the same order of magnitude numerically. For example, it is not advisable to have one objective function that has a value of 10^6 (which is typical of a pressure drop) while another objective function has a value of 10^{-2} (which may be typical of the flow balance parameter).

33.3.5. Optimizer Parameters

You can control the optimizer by modifying the value of the following parameters:

- the scheme, if multiple objective functions are defined

  You have the choice of multi-compromise or multi-priority (see *Objective Functions* (p. 667) for details).

- the mode of optimization

  There are two modes available for the optimization:

  - full optimization

    For this mode, the optimizer searches for the optimum.

  - computation of sensitivities only

    For this mode, the optimizer does not search for the optimum, but instead computes only the sensitivities for the initial configuration. This is useful before starting a full optimization, in order to identify the most influential design variables. If certain sensitivities are found to have negligible impact, the associated design variables or the optimization functions can be removed from the calculation.

- the method for computing the sensitivities

Two methods exist to compute the sensitivities of the optimization functions with respect to the design variables:

- analytical

  For most problems, ANSYS POLYFLOW is able to compute the sensitivities of the optimization functions with respect to the design variables analytically. It is recommended that you use this method to save CPU time.

- finite difference

  The optimizer can apply an infinitesimal perturbation to the design variables (i.e., relative values of 10^{-5}) in order to compute the sensitivities by finite difference. This method is useful if ANSYS POLYFLOW is unable to do it using the analytical method. Such a circumstance can arise when UDFs are used to make the parameters vary depending on the coordinates, or if a discretization method (i.e., the routine that computes the local matrix associated with an equation) does not have all the derivatives; incorrect or even vanishing sensitivities can result. Therefore, the finite difference method would be preferable.

- the maximum number of solutions

  This value sets a limit to the number of solutions calculated by the main F.E.M. solvers. If the optimum has not been reached by this number of solutions, then the optimization process is stopped after the current LS method is complete.

- the maximum number of iterations to reach a minimum

  This value acts as a criterion for convergence, and corresponds to the maximum number of times $P_t(\alpha)$ is calculated as part of the LS method algorithm (as described in *The Line Search (LS) Method* (p. 659).

- the number of iterations before resetting the direction

  This value is used to determine when *Equation 33–21* (p. 658) should be used rather than *Equation 33–16* (p. 657) during the FR method algorithm (as described in *The Fletcher-Reeves (FR) Method* (p. 656)).

- the precision for the design variables (default 10^{-5})

  This precision value is used to check the convergence of the design variables during the LS method.

- the precision for constraints (default 10^{-5})

  This value contributes to the calculation of tolerance ε_C for the ALM method algorithm (see *The Augmented Lagrange Multiplier (ALM) Method* (p. 653) for details). The precision for constraints is a dimensionless quantity.

- the precision for the objective function (default 10^{-5})

  This value contributes to the calculation of tolerance ε_O for the ALM method algorithm (see *The Augmented Lagrange Multiplier (ALM) Method* (p. 653) for details). The precision for the objective function is a dimensionless quantity.

- the initial value for the penalty coefficient

  This initial value is noted as r_0 in the description of the ALM method algorithm (see *The Augmented Lagrange Multiplier (ALM) Method* (p. 653) for details).

- the penalty coefficient multiplier

  This multiplier is noted as KR in *Equation 33–10* (p. 654) of the ALM method algorithm (see *The Augmented Lagrange Multiplier (ALM) Method* (p. 653) for details).

- the maximum value of the penalty coefficient

  This value is noted as r_{MAX} in the discussion of the ALM method algorithm (see *The Augmented Lagrange Multiplier (ALM) Method* (p. 653) for details).

33.3.6. Remarks

The following are some general remarks regarding the use of the optimizer and the ANSYS POLYFLOW solver.

In the optimizer, the cost function is of the order of magnitude of 100. You should therefore note that when you specify an accuracy of 10^{-5} it is with respect to a value of 100 (i.e., the real accuracy is 10^{-3}).

For an optimization task, the accuracy of the solvers will automatically be set to 10^{-6}. Such a high level of accuracy is necessary, because the accuracy of the solvers has a major effect on the behavior of the optimizer.

If the flow problem is nonlinear, it is recommended that you perform an initial calculation (perhaps using evolution) in order to obtain a first solution. The optimization can then start from this solution. Be sure to save this new result file with a different name than the original result file.

An optimization task is not compatible with evolution or transient tasks. The incompatibility with evolution should not be understood as a limitation. If the flow problem is so complex that it requires evolution to obtain a first solution (e.g., due to shear rate dependence on the viscosity, viscous heating, or temperature dependent properties), evolution may still be implemented to calculate an initial solution prior to optimization. You can then start the optimization task using the results file obtained at the end of such an evolution simulation. Note that starting from the converged initial solution does not guarantee the convergence of the subsequent line search algorithm. Similar to the standard evolution scheme, if the solution is converging, the incremental increase of α is also increased until the final solution is reached. If the solution is diverging, then the incremental increase of α is reduced to allow the convergence of the otherwise nonlinear problem. These subiterations are terminated if either the target value of α is reached or the incremental change of α reaches a value below the minimum allowable value. The minimum allowable value is hardcoded as 1% of the total difference between the initial and target values of α.

Because the convergence criteria is relatively small (10^{-6}), it is recommended that you use the Newton-Raphson scheme during optimization. This scheme yields better sensitivities, and usually the new design variable values are close to the previous values (so the solver should converge easily). If the solver does not happen to converge, the optimizer will reduce the variations of the design variables.

During optimization, if the element distortion check is activated and a bad element is detected, the optimizer will reduce the variations of the design variables.

33.4. Problem Setup

This section provides the basic steps for setting up an optimization task in ANSYS POLYDATA.

1. Read the mesh.

   Read a mesh file

2. Read an existing data file that will act as the initial solution for the optimization task.

   Read an old data file

   ### Important

   The old data file that you read and the data file you create for optimization must be coherent with regard to the settings for the quantities you will define as design variables, in order to ensure that the initial solution used for optimization is a converged solution of all the defined sub-tasks.

   In the steps that follow, you will adapt the old data file for optimization.

3. Redefine the global parameters of the task to include optimization.

   Redefine global parameters of a task

   a. Select **Optimization problem(s)**

   b. Click **Accept the current setup**

4. Define the numerical parameters.

   a. Click **Numerical parameters** for the task (e.g., in the **F.E.M. Task 1** menu).

      Numerical parameters

   b. Select **Start from an old result file**.

   c. Select a result file (obtained from the simulation of the initial data file) in the panel that opens. You can either enter the file name and path in the text-entry box, or browse to it and click it. Click **OK** to complete the selection and close the panel.

   d. Click **Upper level menu**.

5. Flag each of the design variables for optimization as a template entry.

   a. Click the **UPDT** button to enable template inputs.

   b. Choose a parameter that you would like to designate as a design variable, and then open the panel that allows you to modify the value of that parameter. The following are examples of the options you could select to open the panel, depending on the kind of optimization you are performing (note that this list does not cover every quantity that can be designated as a design variable; for a complete list, see *Design Variables* (p. 661)).

      - For die shape optimization, click the **Modify amplitude of movement** option for a geometric entity (i.e., a volume, surface, line, or point). For example, if the geometric entity is a point with an amplitude imposed, this option is found in the **Modify control points** menu.
      - For material data parameter optimization, an example would be to click the **Modify fac = <current value>** option in the **Constant viscosity** menu.

- For inlet flow rate optimization, click the **Inflow** option in the **Flow boundary condition along boundary <#>** menu.

A panel will open in which you can revise the **New value**, which will act as the initial value for the design variable. Click **OK** to close the panel and to open the **Create template entry** menu.

c. Create a template entry for the design variable.

Create a new template entry

d. You have the option of changing the default name of the template entry, by clicking **Modify comment = “<default name>"** (where **<default name>** is the name automatically assigned).

Modify comment = “<default name>"

Enter the preferred name for **New value** in the panel that opens, and click **OK**.

e. You have the option of deleting the current template entry, so that this design variable is no longer flagged.

Delete the current template entry

f. Repeat steps 5.(b)–5.(e) for each remaining geometric entity, material data parameter, inlet flow rate, boundary condition parameter, or sub-model parameter you want to flag as a design variable.

g. Click the **UPDT** button to disable template inputs.

h. Click **Upper level menu** until you return to the task menu.

6. Define a postprocessor sub-task for every property/field/value that will be used to define the constraints and objective functions.

7. Define the optimization features.

Optimization

a. If the upper comment section of the **Optimization** menu indicates that external optimization is selected, switch to internal optimization.

Switch to internal optimization

b. Define the design variables.

Define design variables

The **Define design variables** menu will open, which lists all of the templates you created when flagging the design variables.

i. Select a template entry listed under − − − < **Template variables** > − − − and click the **Modify** button to open the **Create a design variable** menu.

ii. Specify that the template entry is a design variable.

Switch to design variable

iii. Define the parameters of the design variable. You have the following options:

- Click **Modify name = “<current name>"** if you want to modify the name of the design variable, and edit the panel that opens.

  Modify name = “<current name>"

  It is recommended that you create a name, as this can make it easier to define constraints and objective functions, to read the listing file generated by ANSYS POLYFLOW, and, in a later stage, to use ANSYS POLYDIAG. The default names may prove to be too similar if there are many design variables.

- Click **Modify value = <current value>** if you want to revise the initial value of the design variable that will be used by the optimizer, and edit the panel that opens.

  Modify value = <currentvalue>

- Click **Modify lower bound = <current value>** if you want to revise the minimum value acceptable for the current design variable, and edit the panel that opens. The reason it is important to have an appropriate lower bound depends upon the kind of optimization you are performing: for die shape optimization, you want an appropriate lower bound for the amplitude in order to avoid excessive deformations of the mesh; for other kinds of optimization, the lower bound helps you minimize the costs associated with searching for the best solution and avoid impossible or undesirable values (e.g., a negative viscosity).

  Modify lower bound = <current value>

- Click **Modify upper bound = <current value>** if you want to revise the maximum value acceptable for the current design variable, and edit the panel that opens. The reason it is important to have an appropriate upper bound depends upon the kind of optimization you are performing: for die shape optimization, you want an appropriate upper bound for the amplitude in order to avoid excessive deformations of the mesh; for other kinds of optimization, the upper bound helps you minimize the costs associated with searching for the best solution and avoid impossible or undesirable values (e.g., the material constant α in the Giesekus model must not exceed 1, the material parameter ξ in the Phan-Thien-Tanner model must be between 0 and 1).

  Modify upper bound = <current value>

- Click **Add/Modify the equality constraint with another design variable** if you want the design variable to be a proxy that is set to be equal to a master design variable, as illustrated in *Equation 33–24* (p. 662).

  Add/Modify the equality constraint with another design variable

  The **Add/Modify the equality constraint** menu will open, displaying all of the previously defined design variables that are not already defined as proxy design variables. Select the appropriate master design variable and click **Upper level menu** to return to the previous menu.

- Click **Remove the equality constraint with another design variable** if you want to remove an existing equality constraint with a master design variable.

  Remove the equality constraint with another design variable

iv. You have the option of changing the design variable back to a template entry by clicking **Reset to template entry**.

Reset to template entry

v. You have the option of deleting the template entry by clicking **Delete the current template entry**.

Delete the current template entry

You should exercise caution when considering this option, as it is not reversible.

vi. Click **Upper level menu** to return to the **Define design variables** menu. The name of the design variable that you defined will be listed under $---<$ **Master design variables** $>---$ or $---<$ **Proxy design variables** $>---$, along with the either the initial value and bounds, or the identifier of the associated master design variable.

Upper level menu

vii. Repeat steps 7.(b)i.–7.(b)vi. for each of the remaining template entries that you want to define as design variables.

viii. Accept the definitions for the design variables and return to the **Optimization (internal)** menu.

Accept

c. Define the extractors that will produce the extracted values (i.e., single scalar values from a particular field) that are used in the objective functions and constraints.

Define value extractors

The **Fields for extractors** menu will open, presenting a list of fields (with their corresponding domain and number of existing extractors). Any field that is already reduced to a single scalar value is not Select able, as it is already in the appropriate form.

i. Click a field from which you want an extracted value, to open the **Management of extractors** menu for that field.

ii. Click **Create an extractor** to open the **Extracted Value #<ID>** menu.

Create an extractor

iii. Click **Modify the name of the extracted value** if you want to revise the default name of the extracted value, and edit the panel that opens.

Modify the name of the extracted value

iv. Click **Select a restriction function** to select or modify the restriction function R that converts the non-scalar field into a scalar field (see *Equation 33–25* (p. 663)). This item is only available for non-scalar fields.

Select a restriction function

The **Select a restriction function** menu will open. Perform one of the following actions to define R:

- If the field is the coordinates, select **length/surface/volume of domain<domain name>** if you want to calculate the length, area, or volume (depending on the number of dimensions) of the domain.

  length/surface/volume of domain <domain name>

- Select **norm (magnitude)** if you want to calculate the norm of the vector (see *Equation 33–27* (p. 664)).

  norm (magnitude)

- Select one of the following if you want to use a particular component of the field (see *Equation 33–28* (p. 664)).

  first component of field

  second component of field

  third component of field

Click **Upper level menu** to return to the previous menu.

v. Click **Select a function** to select or modify the function Q that converts the scalar field into a single value (see *Equation 33–25* (p. 663)). This item is not available if the restriction function is **length/surface/volume of domain <domain name>**.

Select a function

The **Select a function** menu will open. Perform one of the following actions to define Q:

- Select **minimum value** if you want to compute the minimum of the restricted field r_f (see *Equation 33–29* (p. 665)).

  minimum value

- Select **maximum value** if you want to compute the maximum of the restricted field r_f (see *Equation 33–30* (p. 665)).

  maximum value

- Select **average value** if you want to compute the average of the restricted field r_f (see *Equation 33–31* (p. 665)).

  average value

- Select **value at the closest node (in deformed configuration)** if you want to extract the value of the restricted field r_f at the node that is closest to a particular position at the current optimization step (see *Equation 33–32* (p. 665)). Note that the node may change if the mesh is deformed.

value at the closest node (in deformed configuration)

Enter the coordinates of the position in the panels that open.

- Select **weighted average value** if you want to compute the weighted average of the restricted field r_f (see *Equation 33–34* (p. 666)).

  weighted average value

- Select **integral value** if you want to compute the integral of restricted field r_f (see *Equation 33–35* (p. 666)).

  integral value

- Select **value at the closest node (in initial configuration)** if you want to extract the value of the restricted field r_f at the node that closest to a particular position at the first optimization step (i.e., in the initial mesh). See *Equation 33–33* (p. 665) for details.

  value at the closest node (in initial configuration)

  Enter the coordinates of the position in the panels that open.

Click **Upper level menu** to return to the previous menu.

vi. Click **Select a domain** to modify the domain where the extracted value will be evaluated. The default domain is the domain of definition of the field.

Select a domain

The **Select the domain** menu will open, which allows you to perform the following actions:

- Select the **Define domain of the extractor** item in order to select or modify the domain from a list of the existing domains and boundaries.

  Define domain of the extractor

  In the **Domain of the extractor** menu that opens, the upper list displays the domain that is currently selected. To modify the domain, select one of the topo-objects (i.e., a domain or boundary) in the lower list and click the **Add** button. The upper list will be updated with your selection. Only one topo-object can be selected.

  Click **Upper level menu** to return to the **Select the domain** menu.

- If your desired domain was not displayed in the **Domain of the extractor** menu, you can create it by clicking **Create a new topo-object**.

  Create a new topo-object

In the **Add topo-objects** menu that opens, all of the existing topo-objects will be displayed. Select a topo-object in the list and click the **Select object 1** button, and then select another topo-object of the same dimension and click the **Select object 2** button. Next, specify how these two topo-objects will be used to create a new domain by making a selection in the **Select operator** drop-down list: **intersection** if you only want the region where topology objects intersect; or **union** if you want to combine the selected topology objects. Finally, enter a new name in the **Enter new name** text-entry box and click the **create** button.

After you have completed these steps, the new topo-object you created will be accessible in the list, and may be used to create yet another new topo-object. These new topo-objects can then be selected as the domain of the extractor in the **Domain of the extractor** menu.

Click **Upper level menu** to return to the **Select the domain** menu when you are finished creating topo-objects.

- You can change the names of the topo-objects by clicking **Modify the name of a topo-object**.

Modify the name of a topo-object

In the **Modify name of topo-objects** menu that opens, select a topo-object in the list and click the **Select object** button. Then, enter the new name in the **Enter new name** text-entry box and click the **modify** button to change the name.

Click **Upper level menu** to return to the **Select the domain** menu when you are finished modifying the names of the topo-objects.

- You can delete topo-objects by clicking **Delete a topo-object**.

Delete a topo-object

In the **Delete topo-objects** menu that opens, select a topo-object in the list and click the **Select object** button to confirm your choice. Then click the **delete** button.

Click **Upper level menu** to return to the **Select the domain** menu when you are finished deleting topo-objects.

After you have specified the domain where the extracted value will be evaluated, click **Upper level menu** repeatedly until you return to the **Management of extractors** menu.

vii. Repeat steps 7.(c)ii.–7.(c)vi. if you want to create additional extractors for the current field.

viii. Click **Modify <extracted value name>** if you want to modify the setup of an existing extractor.

Modify <extracted value name>

In the **<extracted value name>** menu that opens, perform any of the actions described in steps 7.(c)iii.–7.(c)vi. Repeat this step as necessary.

When you are finished modifying the extractors, click **Upper level menu** repeatedly until you return to the **Management of extractors** menu.

ix. Click **Remove an extractor** if you want to delete an existing extractor.

Remove an extractor

In the **Deletion of an extracted value** menu that opens, click **Delete <extracted value name>**. You will return to the previous menu, where you can repeat this step as necessary.

When you are finished deleting extractors, click **Upper level menu** to return to the **Fields for extractors** menu.

x. If you want to define extractors for any other fields, repeat steps 7.(c)i.–7.(c)ix.

When you are finished defining extractors on all fields, click **Upper level menu** to return to the **Optimization (internal)** menu.

d. Define the objective functions.

Define objective functions

The **Management of objective functions** menu will open.

i. Click **Add an objective function** to define an objective function $f_m(\mathbf{S})$ (see *Equation 33–36* (p. 667) and *Equation 33–37* (p. 667)).

Add an objective function

The **Objective Function #<ID>**menu will open, which allows you to perform the following actions in order to set up objective function $f_m(\mathbf{S})$:

- Click **Modify the name=<objective function name>** if you want to modify the default name, and edit the panel that opens.

  Modify the name=<objective function name>

- Click **Modify the priority = <priority value>** to modify the priority p_m assigned to the objective function (as described in *Objective Functions* (p. 667)), and edit the panel that opens.

  Modify the priority= <priority value>

  The value of the priority must be between 0 (lowest) and 20 (highest). If you define only one objective function for optimization or do not select the multi-priority scheme in a later step, this value will be ignored.

- Click **Modify the goal= <goal> function F** to modify the displayed goal for the optimization of the objective function.

  Modify the goal= <goal> function F

  Your options for **<goal>** include **MINIMIZE** or **MAXIMIZE**.

- Click **Modify function F = <form>** to select or modify the form of the objective function.

  Modify function F = <form>

The **Select a function** menu will open, allowing you to select one of the following functions:

- identity function:

  F = X

- linear function:

  F = a*X + b

- square of a linear function:

  F = (a*X + b)ˆ2 + c

- parabolic function:

  F = a*Xˆ2 + b*x + c

- fourth-order polynomial function:

  F=a+b*X+c*Xˆ2+d*Xˆ3+ e*Xˆ4

- square of two-variable function:

  F = (a*X + b*Y + c)ˆ2 + d

- two-variable function:

  F = a*X + b*Y + c

For recommendations on choosing the form of the objective function, see the paragraph that immediately follows *Equation 33–44* (p. 669).

Click **Upper level menu** to return to the previous menu.

- Click **Modify value X = <extracted value name>** to select or modify the extracted value or field represented by X in the function defined in **Modify function F = <form>**.

  Modify value X = <extracted value name>

  The **Select value X** menu will open, allowing you to select an extracted value or field. After you have made your selection, click **Upper level menu** to return to the previous menu.

- Click **Modify value Y = <extracted value name>** if an extracted value Y is included in the function defined in **Modify function F = <form>**.

  Modify value Y = <extracted value name>

  The **Select value Y** menu will open, allowing you to select an extracted value or field that will be represented by Y. After you have made your selection, click **Upper level menu** to return to the previous menu.

- Click the appropriate items from the list that follows to define the parameters included in the function defined in **Modify function F = <form>**, and edit the panels that open:

Modify parameter a = <parameter a value>

Modify parameter b = <parameter b value>

Modify parameter c = <parameter c value>

Modify parameter d = <parameter d value>

Modify parameter e = <parameter e value>

Note that by default, the parameters a, b, c, d, and e are set to 0.

Important

You must choose values for parameters a, b, c, d and e such that $f_m\ (\mathbf{S})$ is always a positive value, otherwise the minimization of *Equation 33–36* (p. 667) or *Equation 33–37* (p. 667) will not necessarily yield correct results.

- Click **Modify weight factor = <weight factor value>** to modify the weight factor w_m associated with the objective function (see *Equation 33–37* (p. 667)), and edit the panel that opens.

  Modify weight factor = <weight factor value>

  The value of the weight factor is set to 1 by default. If you define only one objective function for optimization or do not select the multi-compromise scheme in a later step, this value will be ignored.

Click **Upper level menu** to return to the **Management of objective functions** menu, when the objective function has been fully defined.

ii. Repeat step 7.(d)i. if you want to define additional objective functions.

iii. Click **<objective function name>** if you want to modify the setup of an existing objective function.

<objective function name>

In the **<objective function name>** menu that opens, perform any of the actions described in the bullets of step 7.(d)i. Click **Upper level menu** to return to the **Management of objective functions** menu, and repeat this step as necessary.

iv. Click **Remove an objective function** to delete an existing objective function.

Remove an objective function

In the **Deletion of an objective function** menu that opens, click **Delete <objective function name>**. You will return to the **Management of objective functions** menu, where you can repeat this step as necessary.

When you are finished deleting objective functions, click **Upper level menu** to return to the **Optimization (internal)** menu.

e. Define the constraints.

   Define constraints

   The **Management of constraints** menu will open.

   i. Click **Add a constraint** to define a constraint.

      Add a constraint

      The **Constraint #<ID>** menu will open, which allows you to perform the following actions in order to set up the constraint:

      - Click **Modify the name= <constraint name>** if you want to modify the default name, and edit the panel that opens.

        Modify the name= <constraint name>

      - Click **Modify the goal=[function F<symbol> 0]** to modify the displayed goal for the constraint.

        Modify the goal: function F<symbol> 0

        Your options for **<symbol>** include: **==** (equal to) for an equality constraint $h_k(\mathbf{S})$ (see *Equation 33–3* (p. 652)); and **>=** (greater than or equal to) or **<=** (less than or equal to) for an inequality constraint $g_j(\mathbf{S})$ (see *Equation 33–2* (p. 652)). Repeatedly click **Modify the goal: function F <symbol> 0** until the appropriate **<symbol>** is displayed.

      - Click **Modify function F = <form>**to select or modify the form of the constraint.

        Modify function F = <form>

        The **Select a function** menu will open, allowing you to select one of the following functions:

        – identity function:

          F = X

        – linear function:

          F = a*X + b

        – square of a linear function:

          F = (a*X + b)ˆ2 + c

        – parabolic function:

          F = a*Xˆ2 + b*x + c

- fourth-order polynomial function:

  F=a+b*X+c*X^2+d*X^3+ e*X^4

- square of two-variable function:

  F = (a*X + b*Y + c)^2 + d

- two-variable function:

  F = a*X + b*Y + c

Click **Upper level menu** to return to the previous menu.

- Click **Modify value X = <extracted value name>** to select or modify the extracted value or field represented by X in the function defined in **Modify function F = <form>**.

  Modify value X = <extracted value name>

  The **Select value X** menu will open, allowing you to select an extracted value or field. After you have made your selection, click **Upper level menu** to return to the previous menu.

- Click **Modify value Y = <extracted value name>** if an extracted value Y is included in the function defined in **Modify function F = <form>**.

  Modify value Y = <extracted value name>

  The **Select value Y** menu will open, allowing you to select an extracted value or field represented by Y. After you have made your selection, click **Upper level menu** to return to the previous menu.

- Click the appropriate items from the list that follows to define the parameters included in the function defined in **Modify function F = <form>**, and edit the panels that open:

  Modify parameter a = <parameter a value>

  Modify parameter b = <parameter b value>

  Modify parameter c = <parameter c value>

  Modify parameter d = <parameter d value>

  Modify parameter e = <parameter e value>

  Note that by default, the parameters a, b, c, d, and e are set to 0. For recommendations on choosing the values for these parameters, see *Constraints* (p. 670).

Click **Upper level menu** to return to the **Management of constraints** menu, when the constraint has been fully defined.

ii. Repeat step 7.(e)i. if you want to define additional constraints.

iii. Click **<constraint name>** if you want to modify the setup of an existing constraint.

<constraint name>

In the **<constraint name>** menu that opens, perform any of the actions described in the bullets of step 7.(e)i. Click **Upper level menu** to return to the **Management of constraints** menu, and repeat this step as necessary.

iv. Click **Remove a constraint** to delete an existing constraint.

Remove a constraint

In the **Deletion of a constraint** menu that opens, click **Delete <constraint name>**. You will return to the **Management of constraints** menu, where you can repeat this step as necessary.

When you are finished deleting constraints, click **Upper level menu** to return to the **Optimization (internal)** menu.

f. Define the numerical parameters that guide the optimizer.

Numerical parameters for optimization

The **Numerical parameters for optimization** menu will open.

i. Click **Modify scheme: <scheme name>** to modify the displayed scheme that the optimizer will use to handle multiple objective functions.

Modify scheme:<scheme name>

The options for **<scheme name>** include the following:

- **multi-priority**

  When the multi-priority scheme is used, each objective function is optimized separately in decreasing order of priority, as described in *Objective Functions* (p. 667).

- **multi-compromise**

  When the multi-compromise scheme is used, the weighted objective functions are all optimized together, according to *Equation 33–37* (p. 667).

Click **Modify scheme: <scheme name>** repeatedly until the appropriate scheme is displayed. Note that if only one objective function is defined in the **Management of objective functions** menu, your scheme selection will be ignored and the optimization will be performed using *Equation 33–36* (p. 667).

ii. Click **Modify method: <current method>** to modify the displayed method of optimization.

Modify method: <current method>

The options for **<current method>** include the following:

- **full optimization**

For this method, the optimizer performs the full optimization.

- **only sensitivities**

  For this method, the optimizer computes only the sensitivities (as described in *Optimizer Parameters* (p. 671).

Click **Modify method: <current method>** repeatedly until the appropriate method is displayed.

iii. Click **Modify method for sensitivities: <current method>** to modify the displayed method for computing the sensitivities.

Modify method for sensitivities: <current method>

The options for **<current method>** include the following:

- **analytical**

  This derivation method is generally recommended.

- **finite differences**

  This method should be used if the **analytical** method is unsuccessful.

iv. Click **Modify maximum number of solutions = <#>** to modify the maximum number of solutions calculated by the main F.E.M. solvers before stopping the optimization, and edit the panel that opens.

Modify maximum number of solutions = <#>

The default value is 100. The optimizer will stop the simulation as soon as it finds an optimum, or when the maximum number of solutions is exceeded after the iterating of the line search method is complete.

v. Click **Show advanced options** if you want to modify additional parameters for the optimizer. Note that these additional parameters should only be used at the request of the support team, as the optimizer generally behaves well with the default values. See *Theory* (p. 652) for additional information on the way the system uses these parameters and evolves during optimization. The additional parameters that can be modified using the advanced options include:

- the number of iterations to reach minimum (default = 3)

  This value acts as a criterion for convergence for the LS method algorithm.

- the number of iterations before reset (default = 4)

  This value affects how the direction is calculated during the FR method algorithm.

- parameters related to the accuracy of various quantities used during the optimization algorithms:
  - the precision for design variables (default = 1e-5)
  - the precision for constraints (default = 1e-5)

        - the precision for objective functions (default = 1e-5)
    - parameters that are used to define the penalty coefficient for the ALM method algorithm:
        - the initial penalty coefficient (default = 1)
        - the penalty coefficient multiplier (default = 2)
        - the maximum penalty coefficient (default = 500,000)

    When you are finished modifying the additional parameters, you can hide the advanced options so that they are not displayed in the **Numerical parameters for optimization** menu by clicking **Hide advanced options**.

    Hide advanced options

vi. You can click **Reset to default values** at any point if you want to revert to the default values for all parameters modified via the **Numerical parameters for optimization** menu.

vii. Click **Upper level menu** to return to the **Optimization (internal)** menu.

    Upper level menu

g. Click **Upper level menu** to return to the task menu.

    Upper level menu

8. Set the convergence criteria and number of iterations for the task to appropriate values.

    Numerical parameters

    The **Numerical parameters** menu will open.

    a. Click **Numerical parameters for iterations**.

        Numerical parameters for iterations

    b. Modify the values using the menu items of the **Numerical parameters for iterations** menu. Note that the convergence criteria must be much lower than it is in usual tasks, to ensure the accuracy of the optimization. The numerical error acts as noise for the optimizer when it is evaluating sensitivities with high precision. It is mandatory to have convergence criteria that is lower than the accuracy requested by the optimizer.

    c. Click **Upper level menu** repeatedly to return to the initial menu.

9. If you want to distinguish the result files that are produced by the optimization from those generated from the initial setup, specify a new prefix for the files.

    Filename syntax

    The **Filename syntax** menu will open. Click **Prefix** and edit the panel that opens. Then click **Upper level menu** to return to the previous menu.

10. Save and exit the ANSYS POLYDATA session.

    Save and exit

In the panels that open, provide a name for the data file (making sure that it is different than the name of the initial data file) and the `.udp` file, and click **Open**.

11. Launch ANSYS POLYFLOW with the new data file to start the optimization.

33.5. Files and Output for Optimization

There are several output files generated by ANSYS POLYFLOW during optimization:

- a standard listing file that lists some optimization information
- a second listing file that is specific for the optimization
- a sensitivities file
- result files for every successful evaluation of the solution

The prefix you specified during the ANSYS POLYDATA session will be used to name the sensitivities file (e.g., `MyExample.sen`).

33.5.1. The Standard Listing File

Instead of combining the optimizer and solver information into a listing file that would be overly complex, a minimum of optimizer information is printed in the standard listing file. The standard listing file first displays the problem definitions, and then the data related to the optimization problem, as shown in the example that follows.

```
*************************
*                       *
*     OPTIMIZATION      *
*                       *
*************************

   - Internal optimizer:                                    Yes.
   - Sensitivities computed by POLYFLOW:           Analytical.
   - Maximum number of solutions to reach optimum:          100
   - Max iteration to find bracketed minimum:                 3
   - Max iteration before resetting search direction:         4
   - Prec. for convergence of design variable:    0.1000000E-04
   - Prec. for convergence of constraints:        0.1000000E-04
   - Prec. for convergence of objective func.:    0.1000000E-04
   - Initial penalty coefficient:                 0.1000000E+01
   - Penalty coefficient multiplier               0.2000000E+01
   - Max penalty:                                 0.5000000E+06

 DESIGN VARIABLES
 ----------------
 * End_Parallel_Dis
     0.0000000E+00 < s  < 0.4990000E+01
     Initial value = 0.0000000E+00
 * J_F_Displ.
     0.0000000E+00 < s < 0.1000000E+01
     Initial value = 0.0000000E+00

 OBJECTIVE FUNCTIONS
 -------------------
 *  Minimization of Objective Function #1
     Multi-objective function priority : 0
     F = X
     with
       X = FLOW_BALANCE

 CONSTRAINT FUNCTIONS
 --------------------
 *  G > 0
```

```
G = a*X + b
with
  X = weighted average(PRESSURE)
  a = 0.1000000E+01
  b = -0.1500000E+07
```

Information related to the current values that will be used for the next evaluation task are displayed in the next section of the standard listing file, as shown in the example that follows. These current values are listed before each evaluation of the solution or before the evaluation of the sensitivities.

```
Optimizer next task : Function evaluation
 Design Variables: next value
  - DV 1 : End_Parallel_Dis
  - DV 2 : J_F_Displ.

Name | Cmp |   Previous    |   Current     |     Target
DV 1 |  1  | 0.9980000E+00 | 0.2612798E+01 | 0.2612798E+01
DV 2 |  1  | 0.1192911E+00 | 0.3123082E+00 | 0.3123082E+00
```

The following examples demonstrates what is displayed for an evaluation of the gradient:

```
Optimizer next task : Gradient evaluation
 Design Variables: next value
  - DV 1 : End_Parallel_Dis
  - DV 2 : J_F_Displ.

Name | Cmp |    Previous   |    Current    |     Target
DV 1 |  1  | 0.3781609E+01 | 0.3781609E+01 | 0.3781609E+01
DV 2 |  1  | 0.4520164E+00 | 0.4520164E+00 | 0.4520164E+00

ANALYTICAL SENSITIVITIES COMPUTATION

Solver : Sensitivities
```

After each evaluation of the solution, the values of the optimization functions (constraint and objective) are printed:

```
**************************
*                        *
*      OPTIMIZATION      *
*                        *
**************************

 Optimizer current task : Function evaluation
  Design Variables: current value
   - DV 1 : End_Parallel_Dis
   - DV 2 : J_F_Displ.

 Name | Cmp |    Previous   |    Current    |     Target
 DV 1 |  1  | 0.3346795E+01 | 0.3781609E+01 | 0.3781609E+01
 DV 2 |  1  | 0.4000429E+00 | 0.4520164E+00 | 0.4520164E+00

  Objective functions :
   - OF 1 : Objective Function #1

 Name  | Active |     Value     |    Grad
 OF 1  |    *   | 0.3713507E+03 |

  Constraints :
   - CF 1 : Constraint on inlet pressure

 Name  |       Value      |  Grad
 CF 1  |  -0.5551383E+05  |
```

After each evaluation of the sensitivities, the values of the optimization functions (constraint and objective) and their sensitivities with respect to the design variables are printed, as shown in the following example. Note that the sensitivities are listed in the column labeled `Grad`.

```
**************************
*                        *
*     OPTIMIZATION       *
*                        *
**************************

 Optimizer current task : Gradient evaluation
  Design Variables: current value
   - DV 1 : End_Parallel_Dis
   - DV 2 : J_F_Displ.

 Name | Cmp |    Previous    |     Current     |      Target
 DV 1 |  1  | 0.3781609E+01 | 0.3781609E+01 | 0.3781609E+01
 DV 2 |  1  | 0.4520164E+00 | 0.4520164E+00 | 0.4520164E+00

  Objective functions :
   - OF 1 : Objective Function #1

  Name  |  Active  |      Value     |       Grad
  OF 1  |    *     | 0.3713507E+03 |
  DV 1  |          |               |  -0.8564837E+01
  DV 2  |          |               |  -0.3527887E+02

  Constraints :
   - CF 1 : Constraint on inlet pressure

  Name  |      Value      |        Grad
  CF 1  |  -0.5551383E+05 |
  DV 1  |                 |  -0.4006829E+06
  DV 2  |                 |   0.4279410E+06
```

Finally, the total number of evaluations of the solution (including those that succeeded and failed) are printed at the end of the file, along with the number of evaluations of the sensitivities (i.e., the gradient evaluations).

```
 RESULT OF THE OPTIMIZER:
 ========================

  _____________________
  Number of function evaluation :  45
  Number of gradient evaluation :  13

 Optimization succeeded.
   GLOBAL CONVERGENCE: End of Optimization
```

33.5.2. The Listing File for Optimization

The information related to the optimization methods (ALM, FR, and LS) appears in a listing file that is dedicated to the optimization, rather than the standard listing file. The base name of this listing file is the same as that of the output sensitivities file (i.e., the `.sen` file), and is followed by the extension `.optslv` (e.g., `MyExample.optslv`). The names of the design variables, objective functions and constraints are unknowns in this file; only their assigned index numbers are known.

The listing file for optimization contains the optimizer parameters and the bounds of the design variables, as shown in the example that follows.

```
**********************
*                    *
*     POLYFLOW       *
*     OPTIMIZER      *
*                    *
**********************


 PARAMETERS:
  - Maximum number of solutions to reach optimum:    100
  - Max iteration to find bracketed minimum:           3
  - Max iteration before reseting search direction:    4
  - Initial penalty coefficient:       +1.000000000e+000
  - Penalty factor multiplier:         +2.000000000e+000
  - Max penalty:                       +5.000000000e+005



INITIAL VARIABLES AND BOUNDS:

Design Variables data:
======================
  1)End_Parallel_Dis
  2)J_F_Displ.

Lower bounds
1) +0.000000000e+000 +0.000000000e+000
Current values
1) +0.000000000e+000 +0.000000000e+000
Upper bounds
1) +4.990000000e+000 +1.000000000e+000
```

Next, an iteration of the ALM method begins, and the initial function values, the penalty coefficient, and the Lagrange Multiplier associate with the violated constraints is printed. The values, the limit value, and the augmented Lagrange limit ($=\frac{-\lambda_{j,p}}{2g_j(S_0)\,r_p}$) are printed for the constraints:

```
BEGIN AUGMENTED LAGRANGE SEQUENCE:     2


Initial Design variable values
------------------------------
Design variables:
  1) +4.528190109e+000 +1.000000000e+000

Initial Function values
-----------------------
Objective function  =   +8.457564090e-001
Normalized objective =  +1.008457564e+002
Normalized cost (A) =   +1.008532804e+002

Constraints:
  - Constraint on inlet pressure is violated (limit + aug. Lagrange limit)
  value = -4.645558861e+004 > +0.000000000e+000 + 3.047173562e+010

Penalty and Lagrange Multipliers for constraints
------------------------------------------------
  Penalty = +2.000000000e+000
  lambda = +7.082372134e-002 for constraint number     0
```

At the beginning of the FR method algorithm, the gradient of the cost function ($A\left(\mathbf{S},\boldsymbol{\lambda}_p,r_p\right)$), the search direction, and the scaled search direction are printed:

```
BEGIN FLETCHER-REEVES STEP   1

 Gradient of objective function df/dDV (Number of gradient evaluation 5)
   1) -4.322516811e-002 -2.005566734e-001
 Search direction:
   1) -3.220570555e-001 +0.000000000e+000
 Search direction (Scaled) :
   1) -1.000000000e+000 +0.000000000e+000
```

At the beginning of the LS method algorithm, the initial and the maximum values of α are printed:

```
Proposed alfa = +2.331935609e-002    alfaMax = +9.074529275e-001

 BEGIN ONE-DIMENSIONAL SEARCH
 Find brackets on minimum
```

Next, the current value of the design variables, the value of the objective function ($\frac{F\ (\mathbf{S})}{F\ (\mathbf{S}_0)}$), the normalized objective ($100 + \frac{F\ (\mathbf{S})}{F\ (\mathbf{S}_0)}$), the normalized cost ($A$), and the status and value of constraints are printed for each function evaluation:

```
1) Alfa = +2.331935609e-002 (Number of function evaluations : 21)

Design variables
   1) +4.411826522e+000 +1.000000000e+000
Objective function    =  +8.515483763e-001
Normalized objective  =  +1.008515484e+002
Normalized cost (A)   =  +1.008515505e+002
Constraints:
    - Constraint on inlet pressure is violated (limit + aug. Lagrange limit)
   value = -3.965115075e+001 > +0.000000000e+000 + 2.322779430e+004
```

The same information as noted previously is printed for a subiteration, except with an appropriate subiteration index number. When the minimum is bracketed, a message is printed with the bracketed bounds:

```
 Brackets, alfaLower = +0.000000000e+000 alfaUpper = +6.105086684e-002
```

At the end of the LS method algorithm, the information that was printed during the LS iteration is printed. Since the end of the LS method is also the end of one iteration of the FR method, the convergence information related to the FR method is printed as well:

```
Result of line search :

    Alfa = +1.507287472e-002 (Number of function evaluations : 24)

Design variables
     1) +4.452976464e+000 +1.000000000e+000
  Objective function    =  +8.493531370e-001
  Normalized objective  =  +1.008493531e+002
  Normalized cost (A)   =  +1.008508696e+002
  Constraints:
   - Constraint on inlet pressure is violated (limit + aug. Lagrange limit)
     value = -1.644541087e+004 > +0.000000000e+000 + 2.322779430e+004

 END FLETCHER-REEVES STEP 1
 CONVERGENCE TEST.

Step check:
  step             = +1.507287472e-002 | precDV*.5 = +4.999999850e-006
  step+stepDirPrev = +1.015072875e+000 | precDV    = +9.999999700e-006
Cost check:
  fCost-fCostprev  = +2.410837052e-003 | CADOE_EPSILON = +1.000000000e-014
  (fCost-fCostprev)/SDelta = +2.390497123e-005 | precLag = +9.999999700e-006
  SDelta = +1.008508661e+002
Non Convergence for design variables and cost.
Slope = +2.173368091e-006
```

When convergence is reached for the FR method algorithm, the information related to the convergence of the ALM method is printed:

```
END OF AUGMENTED LAGRANGE SEQUENCE    2

Design variables
  1) +4.452616373e+000 +1.000000000e+000

Objective and constraint values
Objective function    = +8.493717665e-001
Normalized objective  = +1.008493718e+002
Normalized cost (A)   = +1.008508695e+002
Constraints:
  - Constraint on inlet pressure is violated (limit + aug. Lagrange limit)
    value = -1.630181300e+004 > +0.000000000e+000 + 2.322779430e+004

Gradient of objective function df/dDV (Number of gradient evaluation : 7)
  1) -5.175122201e-002  -1.985748837e-001

CHECK CONVERGENCE OF NON_LINEAR CONSTRAINT
  const[ 0] = +1.242643884e-002 | precision : +9.999999700e-006
Convergence not assumed for non-linear constraints.

CHECK CONVERGENCE OF BOUND CONSTRAINT
 Bound constraints are converged.

CHECK CONVERGENCE OF OBJECTIVE FUNCTION
 Final cost      = +8.493717665e-001
 Previous cost   = +8.457564090e-001
 Cost variation  = +3.615357495e-003 | precision = +9.999999700e-004
 Convergence not assumed for function cost.
GLOBAL CONVERGENCE NOT YET SATISFIED
```

At the end of the optimization, the total number of function evaluations (including those that failed and succeeded) and the sensitivities (gradient) evaluations are printed:

```
GLOBAL CONVERGENCE: End of Optimization



_____________________
Number of function evaluation :  45
Number of gradient evaluation :  13
```

The optimization status is also accessible in ANSYS POLYDIAG, as described in *Exiting ANSYS POLYDIAG* (p. 596).

33.5.3. The Sensitivities Files

The sensitivities file is created or updated after each computation of the sensitivities. This file is not used with the internal optimizer, and is only created when you have specified that only the sensitivities are calculated. This file contains blocks of information related to design variables and optimization functions (objective   and constraint). Only the value and sensitivities are relevant.

The following provides examples of the various blocks of information:

- design variable (block DSGFLD)

  ```
  BEGIN DSGFLD
  #       ___ design variable name
  #      /
   C 16|End_Parallel_Dis
   I  1|  17
   I  1|   4
  #       ___ lower and upper bound
  #      /
   D  2|  0.0000000E+00 0.4990000E+01
  #       ___ current value
  #      /
   D  1|  0.4411740E+01
   ENDOF DSGFLD
  ```

- optimization functions (block RESULTOC) and their sensitivities (block DERIVATIVE)

  ```
  BEGIN RESULTOC
   C 21|Objective Function #1
   I  3|   1   1   0
   D  3| 0.1000000E+01 0.0000000E+00 0.1000000E+01
  ```

```
#        ___ current value
#       /
 D  2| 0.3310040E+03 0.0000000E+00
 BEGIN DERIVATIVE
#        ___ design variable name
#       /
 C 16|End_Parallel_Dis
#        ___ sensitivities value
#       /
 D  1| -0.2131169E+02
 ENDOF DERIVATIVE
 BEGIN DERIVATIVE
 C  10| J_F_Displ.
 D   1| -0.7662757E+02
 ENDOF DERIVATIVE
 ENDOF RESULTOC
```

33.5.4. The Result Files for Successful Evaluations of the Solution

After each solution has been computed, the result is saved as an unformatted file. The name of such a file is `opt<#i>.res`, where <#i> is the index number of the successful evaluation of the solution. The name of the folder in which these files are located is the base name of the sensitivities file with the suffix `_optdir` (e.g., `MyExample_optdir`).

The solutions stored in the result files for successful evaluations are used as the initial solution when the new solution to compute is close to one computed previously. If the new solution to compute has already been computed, the solution is read in the appropriate file instead of the computation of the solution, and the following message is printed in the standard listing file:

```
Solution is read from an old file since this solution has
 already been computed.
```

33.6. Additional Options for Solution Exploration

The internal optimization capabilities described throughout this chapter are fully integrated with the ANSYS POLYFLOW code, and hence provide the most efficient way to optimize your problem in terms of CPU time and memory usage. ANSYS POLYFLOW also allows you to use other means—namely, the ANSYS Workbench tool Design Exploration, and VisualDOC—for exploring solutions and improving your design parameters. These additional means may be useful in some circumstances, and are described in the sections that follow.

33.6.1. Design Exploration

ANSYS POLYFLOW is data-integrated with ANSYS Workbench, which provides you with convenient access to ANSYS applications and utilities that manage the product workflow. Design Exploration is one of the tools available in ANSYS Workbench, and is particularly useful for exploring the designs of complicated parts and assemblies. Design Exploration allows you to easily manipulate parameters defined in any analysis system supported by ANSYS Workbench, ANSYS DesignModeler, and various CAD systems, and it provides unique means for understanding the subsequent analysis responses.

Unlike ANSYS POLYFLOW's internal optimizer, Design Exploration does not perform gradient-based optimization; instead, it uses a deterministic method based on the discipline of design of experiments (DOE). Using a goal-driven optimization technique, Design Exploration can obtain a multiplicity of design points, and then allow you to explore the calculated response surface and generate design points directly from the surface. With minimal effort, responses can be studied, quantified, and graphed.

After setting up your analysis in ANSYS Workbench, you can use Design Exploration to:

- parameterize your solution and view an interpolated response surface for the parameter ranges
- view the parameters associated with the minimum and maximum values of your outputs
- create a correlation matrix that shows you the sensitivity of outputs to changes in your input parameters
- set output objectives and see what input parameters will meet those objectives
- perform a Six Sigma analysis on your model

There are circumstances where it may be beneficial for you to use Design Exploration rather than or in conjunction with ANSYS POLYFLOW's internal optimizer when seeking optimal shape and flow parameters. For example, you may want to investigate changes to the mesh geometry that are not allowed as part of a mesh deformation preprocessor in ANSYS POLYDATA (see *Remarks and Limitations* (p. 636) for details); you may want to take advantage of ANSYS Workbench's ability to easily link and update multiple analysis systems; or perhaps you want to utilize Design Exploration's analysis tools. For more information about ANSYS Workbench, see the separate POLYFLOW in **Workbench** User's Guide. Details about Design Exploration can be found using the ANSYS Workbench's online help, which is available from the **Help** menu or any of the links in the quick help or sidebar help in ANSYS Workbench.

33.6.2. VisualDOC

If you are an experienced user of VisualDOC and VisualScript, then it may be convenient for you to use these software packages to manage the different programs (e.g., GAMBIT, ANSYS POLYDATA, ANSYS POLYFLOW) when seeking to optimize the parameters. See *Appendix B* (p. 701) for details. Note that VisualDOC acts as an external optimizer, and will not be as efficient as ANSYS POLYFLOW's internal optimizer.

Appendix A. Sub-Task Compatibility Charts

The figures in this appendix provide information about the compatibility of sub-tasks:

- For information about which sub-tasks are compatible with various geometries and models, see *Figure A.1* (p. 698).
- For information about which sub-tasks can be created on the same domain, see *Figure A.2* (p. 699).
- For information about which sub-tasks can be created on neighboring domains that share an interface, see *Figure A.3* (p. 700).

Figure A.1 Compatibility of Sub-Tasks with Geometries and Models

	Heat conduction problem	Generalized Newtonian isothermal flow problem	Generalized Newtonian non-isothermal flow problem	Differential viscoelastic isothermal flow problem	Differential viscoelastic non-isothermal flow problem	Integral viscoelastic isothermal flow problem	Integral viscoelastic non-isothermal flow problem	Simplified viscoelastic isothermal flow problem	Simplified viscoelastic non-isothermal flow problem	Darcy isothermal flow problem	Darcy non-isothermal flow problem	Transport of species*	Closure**	Potential problem	Film model: Gen. Newtonian isothermal	Film model: Gen. Newtonian non-isothermal	Film model: Viscoelastic isothermal	Film model: Viscoelastic non-isothermal	Shell model: Gen. Newtonian isothermal	Shell model: Gen. Newtonian non-isothermal	Shell model: Viscoelastic isothermal	Shell model: Viscoelastic non-isothermal	Isothermal crystallisation	Non-isothermal crystallisation	Elasticity (isothermal)	Elasticity (non-isothermal)	Postprocessor of Elasticity (isothermal)	Postprocessor of Elasticity (non-isothermal)	Linear Thermo-Elastic stresses and deformations	Residual stresses and deformations (Narayanaswamy)	Internal radiation
2D	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
2.5D	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N
3D	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y
2D Shell	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	Y	N	N	N	N	N	N	N	N	N
Steady	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	Y
Evolution	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	N	Y
Transient	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y	Y	Y	Y	N	N	N	N	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
MST	N	Y	Y	n	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N
VOF	N	Y	Y	Y	Y	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Key:	
Y	- Compatible
N	- Not compatible
?	- Unknown

* - Transport of species sub-tasks require a preliminary definition of species and flow.

** - Closure sub-tasks require species, chemical reactions, and transport of species.

Figure A.2 Compatibility of Sub-Tasks on the Same Domain

	Heat conduction problem	Generalized Newtonian isothermal flow problem	Generalized Newtonian non-isothermal flow problem	Differential viscoelastic isothermal flow problem	Differential viscoelastic non-isothermal flow problem	Integral viscoelastic isothermal flow problem	Integral viscoelastic non-isothermal flow problem	Simplified viscoelastic isothermal flow problem	Simplified viscoelastic non-isothermal flow problem	Darcy isothermal flow problem	Darcy non-isothermal flow problem	Transport of species*	Closure**	Potential problem	Film model: Gen. Newtonian isothermal	Film model: Gen. Newtonian non-isothermal	Film model: Viscoelastic isothermal	Film model: Viscoelastic non-isothermal	Shell model: Gen. Newtonian isothermal	Shell model: Gen. Newtonian non-isothermal	Shell model: Viscoelastic isothermal	Shell model: Viscoelastic non-isothermal	Isothermal crystallisation	Non-isothermal crystallisation	Elasticity (isothermal)	Elasticity (non-isothermal)	Postprocessor of Elasticity (isothermal)	Postprocessor of Elasticity (non-isothermal)	Linear Thermo-Elastic stresses and deformations	Residual stresses and deformations (Narayanaswamy)	Internal radiation
Heat conduction problem	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y
Generalized Newtonian isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Generalized Newtonian non-isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	Y	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y
Differential viscoelastic isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Differential viscoelastic non-isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	?
Integral viscoelastic isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Integral viscoelastic non-isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	?
Simplified viscoelastic isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Simplified viscoelastic non-isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	?	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	?
Darcy isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Darcy non-isothermal flow problem	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	?
Transport of species*	N	Y	Y	N	N	N	N	?	?	N	N	Y	N	N		N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Closure**	N	Y	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Potential problem	Y	N	Y	N	?	N	?	N	?	N	?	N	N	N	N	?	N	?	N	N	N	N	N	?	N	N	N	N	N	?	N
Film model: Gen. Newtonian isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Film model: Gen. Newtonian non-isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N
Film model: Viscoelastic isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Film model: Viscoelastic non-isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Shell model: Gen. Newtonian isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Shell model: Gen. Newtonian non-isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Shell model: Viscoelastic isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Shell model: Viscoelastic non-isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Isothermal crystallisation	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Non-isothermal crystallisation	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Elasticity (isothermal)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Elasticity (non-isothermal)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Postprocessor of Elasticity (isothermal)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Postprocessor of Elasticity (non-isothermal)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Linear Thermo-Elastic stresses and deformations	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Residual stresses and deformations (Narayanaswamy)	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Internal radiation	Y	N	Y	N	?	N	?	N	?	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Key:

Y	- Compatible
N	- Not compatible
?	- Unknown

* - Transport of species sub-tasks require a preliminary definition of species and flow.

** - Closure sub-tasks require species, chemical reactions, and transport of species.

Figure A.3 Sub-Tasks on Neighboring Domains that Share an Interface

	Heat conduction problem	Generalized Newtonian isothermal flow problem	Generalized Newtonian non-isothermal flow problem	Differential viscoelastic isothermal flow problem	Differential viscoelastic non-isothermal flow problem	Integral viscoelastic isothermal flow problem	Integral viscoelastic non-isothermal flow problem	Simplified viscoelastic isothermal flow problem	Simplified viscoelastic non-isothermal flow problem	Darcy isothermal flow problem	Darcy non-isothermal flow problem	Transport of species*	Closure**	Potential problem	Film model: Gen. Newtonian isothermal	Film model: Gen. Newtonian non-isothermal	Film model: Viscoelastic isothermal	Film model: Viscoelastic non-isothermal	Shell model: Gen. Newtonian isothermal	Shell model: Gen. Newtonian non-isothermal	Shell model: Viscoelastic isothermal	Shell model: Viscoelastic non-isothermal	Isothermal crystallisation	Non-isothermal crystallisation	Elasticity (isothermal)	Elasticity (non-isothermal)	Postprocessor of Elasticity (isothermal)	Postprocessor of Elasticity (non-isothermal)	Linear Thermo-Elastic stresses and deformations	Residual stresses and deformations (Narayanaswamy)	Internal radiation
Heat conduction problem	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	N	N	N	N
Generalized Newtonian isothermal flow problem	N	Y	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	?	N	Y	N	Y	N	N	N	N
Generalized Newtonian non-isothermal flow problem	Y	N	Y	N	Y	N	Y	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	?	N	Y	N	Y	N	N	N
Differential viscoelastic isothermal flow problem	N	Y	N	Y	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N
Differential viscoelastic non-isothermal flow problem	Y	N	Y	N	Y	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
Integral viscoelastic isothermal flow problem	N	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Integral viscoelastic non-isothermal flow problem	Y	N	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Simplified viscoelastic isothermal flow problem	N	Y	N	?	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N
Simplified viscoelastic non-isothermal flow problem	Y	N	Y	N	?	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
Darcy isothermal flow problem	N	Y	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Darcy non-isothermal flow problem	Y	N	Y	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Transport of species*	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Closure**	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Potential problem	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Film model: Gen. Newtonian isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Film model: Gen. Newtonian non-isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N
Film model: Viscoelastic isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Film model: Viscoelastic non-isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N
Shell model: Gen. Newtonian isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N
Shell model: Gen. Newtonian non-isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N
Shell model: Viscoelastic isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N	N
Shell model: Viscoelastic non-isothermal	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	Y	N	N	N	N	N	N	N	N	N
Isothermal crystallisation	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	?	N	N	N	N	N	N
Non-isothermal crystallisation	?	N	?	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	?	N	N	N	N	N
Elasticity (isothermal)	N	Y	N	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	Y	N	N	N	N	N	N
Elasticity (non-isothermal)	N	N	Y	N	Y	N	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	?	N	Y	N	N	N	N	N
Postprocessor of Elasticity (isothermal)	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N
Postprocessor of Elasticity (non-isothermal)	N	N	Y	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N
Linear Thermo-Elastic stresses and deformations	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N
Residual stresses and deformations (Narayanaswamy)	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N
Internal radiation	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y

Key:

Y	- Compatible
N	- Not compatible
?	- Unknown

* - Transport of species sub-tasks require a preliminary definition of species and flow.

** - Closure sub-tasks require species, chemical reactions, and transport of species.

Appendix B. Running ANSYS POLYFLOW with VisualDOC

While *Optimization* (p. 651) provided information about the internal optimization capability of ANSYS POLYFLOW, this appendix will describe how to use an external optimizer to manage the loop of optimization. The external optimizer that can be coupled with ANSYS POLYFLOW is VisualDOC 3.1, a tool developed by Vanderplaats Research and Development Inc. (or more information about VisualDOC, see http://www.vrand.com). VisualDOC calls different software products (e.g., ANSYS POLYDATA, ANSYS POLYFLOW, GAMBIT) in order to get relevant values, to evaluate the sensitivities, to update the mesh and data files, and to improve the solution.

The coupling of ANSYS POLYFLOW with VisualDOC allows for the addition of extrusion optimization capabilities that are required for applications involving flow balancing at die exits (see *Quantification of Die Balancing* (p. 569) for details).

This appendix assumes that you are an experienced user of VisualDOC and VisualScript software. It contains the following sections:

B.1. Introduction

VisualDOC 3.1 is a graphics-based design optimization software package, that provides convenient tools for adding optimization capabilities to almost any design task. It uses a powerful graphical user interface along with state-of-the-art optimization algorithms to set up, solve, and postprocess design problems.

VisualDOC uses both gradient-based and response-surface approximate optimization algorithms. Design variables, derived from die geometry, material properties, and operating conditions such as flow rate or boundary conditions, are defined in VisualDOC and are provided to ANSYS POLYFLOW as inputs for flow analysis. VisualDOC uses the results returned by ANSYS POLYFLOW to compute new values for design variables through the application of optimization algorithms.

Control of data flow and communication between VisualDOC and ANSYS POLYFLOW (or an external source) is managed by a script that you generate using VisualScript. Optimization of design problems is handled exclusively by VisualDOC through the iterative coupling of VisualDOC and ANSYS POLYFLOW.

Quantities computed by the VisualDOC Optimizer and used by ANSYS POLYFLOW are called inputs. There are design variables (e.g., geometry, material properties, flow rate, or boundary condition variables) and the variables linked to the design variables by algebraic relations (e.g., initial values, lower and upper bounds). Inputs are determined by the Optimizer and are replaced in the data file.

Quantities computed by ANSYS POLYFLOW and used by the VisualDOC Optimizer are called responses. They are objective functions and optimization constraints. Responses are computed by ANSYS POLYFLOW and are extracted from the results file. The Optimizer uses responses to compute new values for the inputs.

The extraction of responses and replacements of inputs is controlled by a script that you generate using VisualScript. VisualScript allows you to create and edit a Perl script file containing replace and extract operations as well as analysis program calls. The Perl script file, which is output from VisualScript, is executed by VisualDOC as the control mechanism for optimization. ANSYS POLYDATA generates a file named `optimization.help` to help guide you in VisualDOC and VisualScript usage.

B.2. Constraints and Limitations

The optimization task in VisualDOC is based on the sensitivities of the responses with respect to the design variables. The Optimizer uses a finite difference scheme to obtain the sensitivities. It varies the design variables by the finite-difference step size (ε). The sensitivity of responses (R), with respect to the design variables (x) is evaluated as

$$Sensitivity = \frac{R(x+\varepsilon) - R(x)}{\varepsilon} \qquad \textbf{(B–1)}$$

The design variables that are close to zero are perturbed with the absolute value of ε, while design variables that are not close to zero are perturbed with the relative value of ε. In VisualDOC, the default relative and absolute values for ε are 0.001 and 0.0001, respectively.

The finite difference scheme that is used for the sensitivity evaluation requires an accurate solution of the flow problem. To meet this constraint, the convergence criteria for the ANSYS POLYFLOW solver must be smaller than the finite-difference step size of the optimizer. To satisfy this, it is recommended that you change the convergence criteria for an optimization task in ANSYS POLYDATA. Typical values of the convergence criteria for the optimization should be in the range of 10^{-6} to 10^{-7}. By default, when an optimization task is defined in ANSYS POLYDATA, the convergence criteria is set to a maximum value of 10^{-6}. In the Picard iterative scheme, the number of iterations must be adapted to the convergence criterion.

At each optimization step, a new mesh is generated. Therefore, it is important that meshes always maintain the same topology (i.e., the same number of nodes and elements). This is especially true for the sensitivity computation based on the finite difference scheme. To meet this constraint, it is recommended that GAMBIT generate a mesh with the same topology. If the mesh topology is not maintained, ANSYS POLYFLOW will restart from an old results file. See Example 79 in the ANSYS POLYFLOW Examples Manual for details.

To reduce the computation time for the die design optimization, assume that the optimization task starts from a solution close to the optimized one. The starting solution is named “initial task" and the associated files are preceded by the prefix “initial" (e.g., initial data file, initial mesh file, etc.). As an optimization task starts from a solution close to the optimized one, and at each optimization step, ANSYS POLYFLOW restarts the calculation from the result file obtained at the previous step. Since an evolution scheme is not required, the optimization is not compatible with evolution and transient tasks. This is not a limitation of the software because VisualDOC receives a return error if ANSYS POLYFLOW does not converge. In this case, VisualDOC restarts the current optimization step with new values for the inputs.

B.3. Parameterization Types in ANSYS POLYDATA

ANSYS POLYDATA distinguishes between three types of parameterization:

- ANSYS POLYDATA parameterization

With ANSYS POLYDATA parameterization, the design variables are related to ANSYS POLYDATA material property data and operating conditions (flow rate or boundary conditions).

- geometrical parameterization

  With geometrical parameterization, the design variables are related to the geometry. Note that for this kind of parameterization, you must include a mesh tool such as GAMBIT in the optimization loop.

- external parameterization

  With external parameterization, the design variables are related to an external input data file or an external analysis code. This occurs if a design variable is a parameter of a user-defined function (UDF), if an external program first computes a parameter, or if the result of ANSYS POLYFLOW is subsequently treated by an external program to obtain the responses.

These three types of parameterization can be combined in an ANSYS POLYDATA session. ANSYS POLYDATA determines what types of input and response files are needed. It also determines the major steps that are required in VisualDOC and VisualScript.

B.3.1. Optimization Files in ANSYS POLYDATA

By default, VisualDOC and VisualScript use two files for the transfer of inputs to, and responses from, analysis codes (e.g., ANSYS POLYFLOW):

- `dvar.vef` file for input values
- `resp.vef` file for response values

Both of these files use the VEF format and are fully compatible with VisualDOC. For each input, the `dvar.vef` file contains the input values (one per line) in the same order as defined in VisualDOC. A sample `dvar.vef` file is shown below:

```
dvar.vef
987.654321  <   Value of input 1
12.3456789  <   Value of input 2
```

For each response, the `resp.vef` file contains the response values (one per line) in the same order as defined in VisualDOC. A sample `resp.vef` file is shown below:

```
resp.vef
0.23456582154  <  Value of response 1
125.562151215  <  Value of response 2
-1             <  Value of response 3
```

Typically, VisualScript is used to replace input values in the data file, and to extract response values from the results file of the analysis code. However, for certain types of parameterization, ANSYS POLYDATA reads input values directly from the `dvar.vef` file, and ANSYS POLYFLOW writes the responses directly to the `resp.vef` file, avoiding replacement and extraction operations in VisualScript.

If ANSYS POLYDATA is not able to read inputs and write responses directly, intermediate PFL-formatted files are generated, which simplify input and response searches in the ANSYS POLYFLOW data and results files. The PFL format includes comments and is easy to read. The translation to PFL format is handled by VisualScript. Inputs are replaced in `dvar.pfl` and responses are extracted from `resp.pfl`. For each input, the `dvar.pfl` file contains a line starting with DV, followed by the index of the input, a line with the comment specified during the ANSYS POLYDATA session, and a line containing the value of the input.

A sample `dvar.pfl` is shown below:

```
dvar.pfl
 DV #1
 Flow rate along BS1
 987.654321     <  Value of input 1
 DV #2
 Flow rate along BS2
 12.3456789     <  Value of input 2
```

For each response, the `resp.pfl` file contains a line starting with RESP, followed by the index of the response, a line with the comment specified during the ANSYS POLYFLOW session, and a line with the value of the response. A sample `resp.pfl` file is shown below:

```
resp.pfl
RESP #1
Velocity at point A
0.23456582154  <   Value of response 1
RESP #2
Temperature at point B
125.562151215  <   Value of response 2
RESP #3
Pass/Fail POLYFLOW status
-1             <   Value of response 3
```

You can change the name of the input and response files, if desired, but to avoid naming conflicts within VisualDOC, it is recommended that you use the filenames suggested by ANSYS POLYDATA. Different scenarios for the data transfer between the Optimizer and analysis codes are illustrated in *Figure B.1* (p. 705) and *Figure B.2* (p. 706).

B.3.1.1. ANSYS POLYDATA Parameterization

If design variables are related to ANSYS POLYDATA and if ANSYS POLYFLOW computes the responses, the VEF format is used for both `dvar.vef` and `resp.vef` files. For ANSYS POLYDATA parameterization, VisualDOC writes the `dvar.vef` file and ANSYS POLYDATA reads inputs directly from `dvar.vef` file (Direct read from ANSYS POLYDATA scenario in *Figure B.1* (p. 705)). ANSYS POLYFLOW writes responses directly to the `resp.vef` file, and VisualDOC reads it (Direct write from ANSYS POLYFLOW scenario in *Figure B.2* (p. 706)).

Figure B.1 Replacement Operations for Data Files

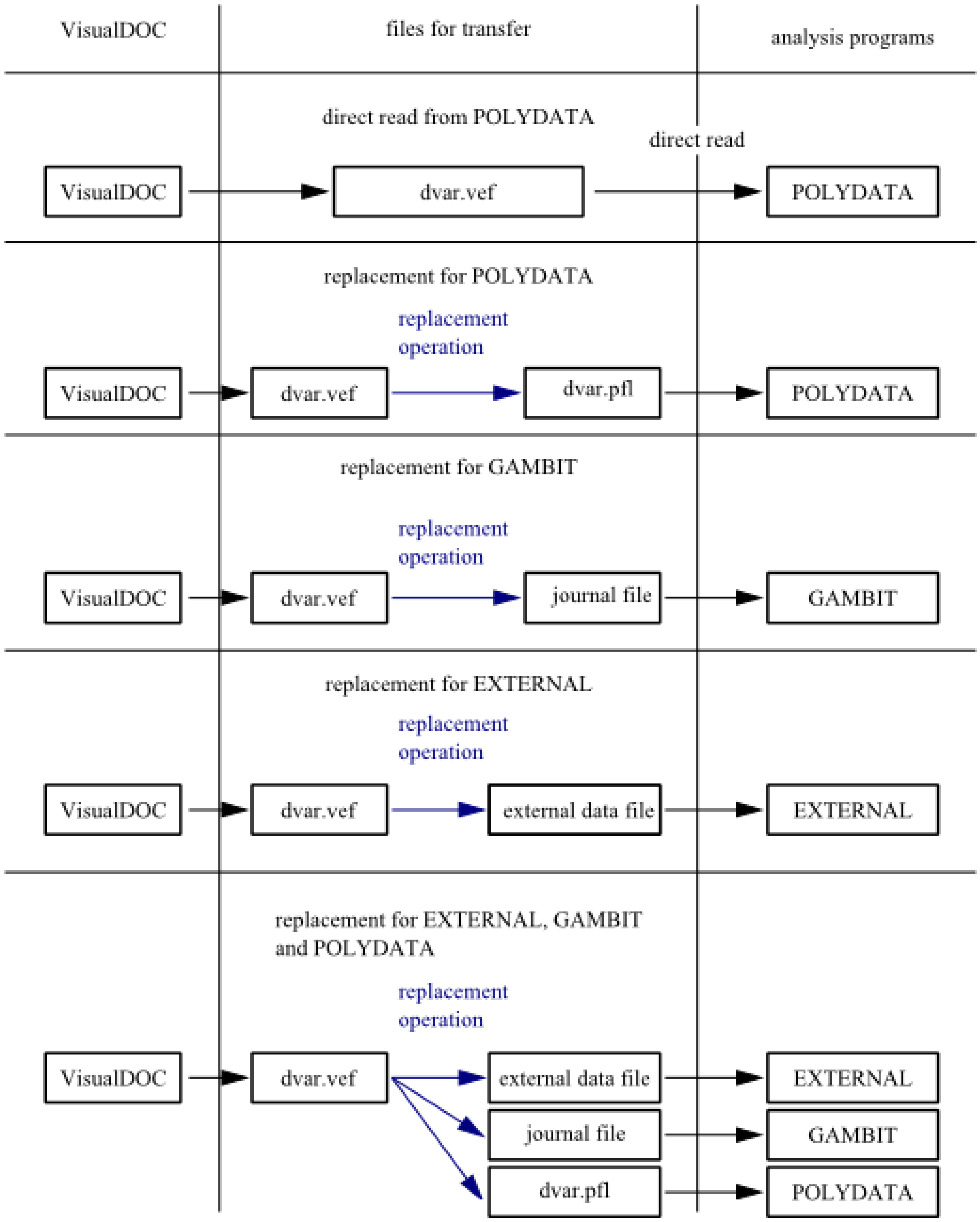
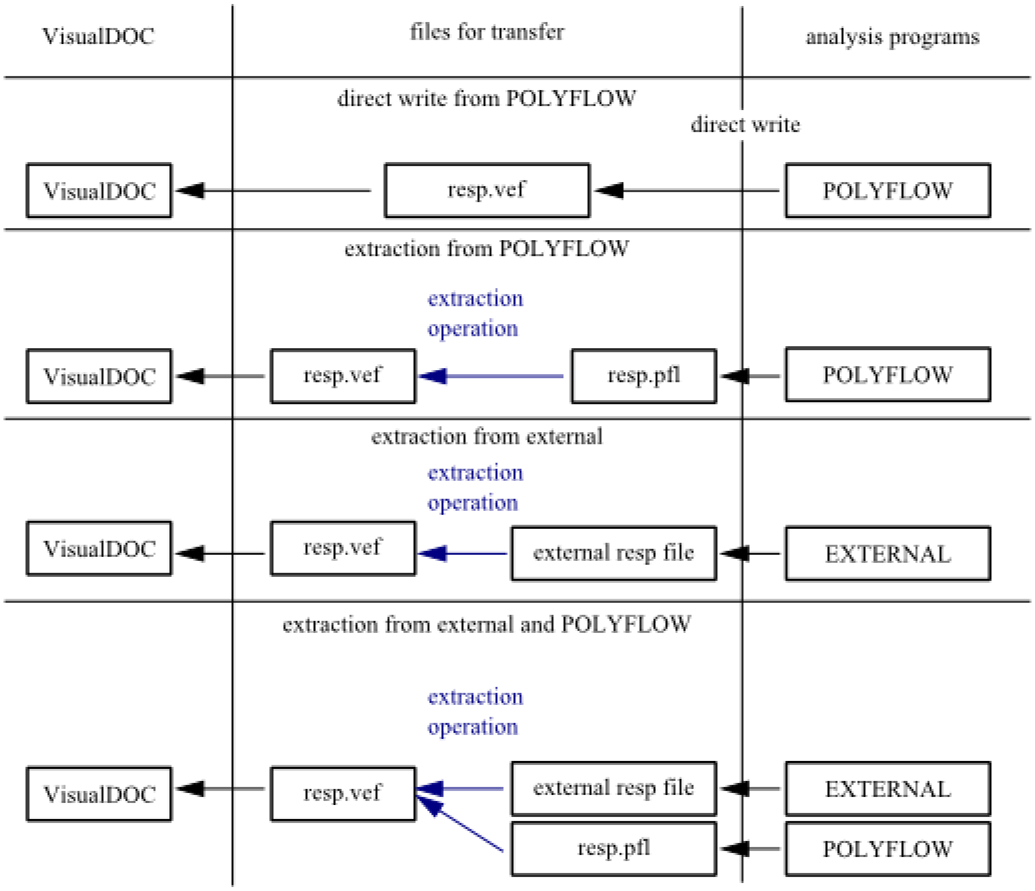


Figure B.2 Extraction Operations from Result Files



B.3.1.2. Geometrical Parameterization

If design variables are related to the geometry and if ANSYS POLYFLOW computes the responses, the VEF format is used for both `dvar.vef` and `resp.vef` files. VisualDOC writes the `dvar.vef` file, but since GAMBIT cannot read the VEF format, VisualScript replaces the design variables in the GAMBIT journal file and GAMBIT reads it (Replacement for GAMBIT scenario in *Figure B.1* (p. 705)). ANSYS POLYFLOW writes responses directly to the `resp.vef` file and VisualDOC reads it (Direct write from ANSYS POLYFLOW scenario in *Figure B.2* (p. 706)).

B.3.1.3. External Parameterization

The PFL format is used for both input and response files if the design variables are related to an external data file or an external analysis code and if the responses are computed by ANSYS POLYFLOW.

If design variables are related to an external data file (e.g., design variables are defined in a UDF or computed by an external program), VisualScript replaces the design variables in the data file for the external code and the external code reads it (Replacement for External scenario in *Figure B.1* (p. 705)). ANSYS POLYFLOW writes responses directly in the `resp.vef`, which is read by VisualDOC (Direct write from ANSYS POLYFLOW scenario, *Figure B.2* (p. 706)).

If design variables are related to an external analysis code (e.g., the result of ANSYS POLYFLOW is subsequently used by an external analysis code), then responses are written by ANSYS POLYFLOW to the `resp.pfl` file and by the external code to its own response file. VisualScript extracts responses related to ANSYS POLYFLOW from the `resp.pfl` file and writes them to the `resp.vef` file (Extraction from ANSYS POLYFLOW scenario in *Figure B.2* (p. 706)).

The external code writes an external response file, which is used by VisualScript to extract responses and write them back to the same `resp.vef` file (Extraction from External scenario in *Figure B.2* (p. 706)).

B.3.1.4. Combining Types of Optimization

You can combine parameterization types in an ANSYS POLYDATA session, but the order of design variable declaration is important. You cannot mix design variables related to different analysis codes; instead, you must first define all design variables of ANSYS POLYDATA type, then define all design variables of geometrical tool type, and finally define all design variables of external tool type.

B.3.2. Tagging of Inputs

The tagging of inputs for ANSYS POLYDATA and GAMBIT is discussed below. The tagging of the inputs for an external program is left up to you.

B.3.2.1. Tagging of Inputs in ANSYS POLYDATA

In ANSYS POLYDATA, the inputs are tagged using the **UPDT** button. When you click **UPDT**, a menu is displayed that allows you to create an entry for a template for a particular parameter. During optimization, ANSYS POLYDATA writes a tag and its value to the input file for the parameter. The input file is used by VisualDOC for the replacement of inputs.

B.3.2.2. Tagging of Inputs in GAMBIT

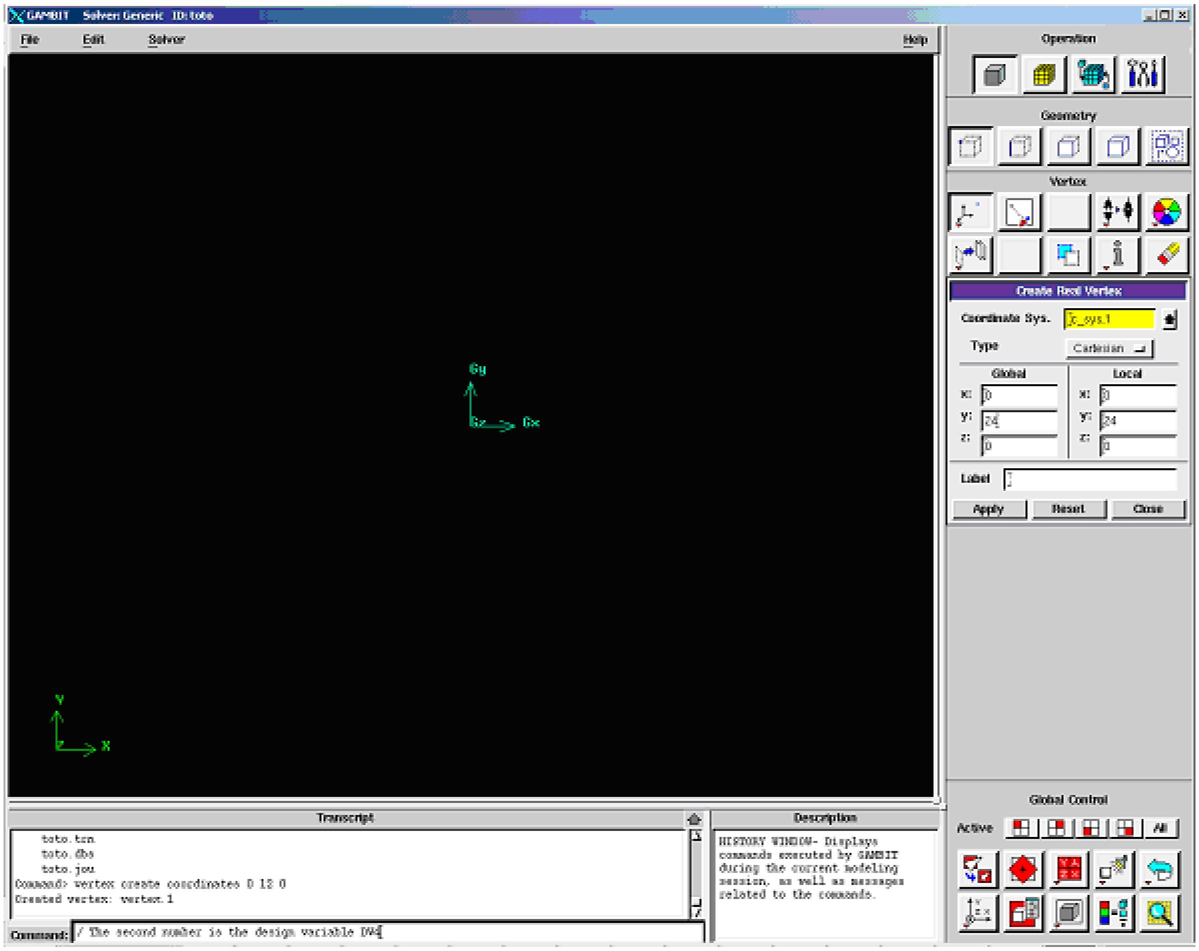
The journal file that is produced by GAMBIT is used for geometrical optimization. You will need to use VisualScript to replace inputs in the journal file. To identify inputs in the GAMBIT journal file, you can enter a comment in the command line before executing an action. This comment is subsequently written to the journal file.

For example, for a coordinate of (0, 24, 0) where the y component is a design variable, add the following comment (see *Figure B.3* (p. 708)) just before creation of the coordinate:

```
/ The second number is the design variable DV4
```

Press **RETURN** to create the coordinate.

Figure B.3 Tagging Design Variable in GAMBIT



Create Real Vertex

Coordinate Sys. c_sys.1

Type Cartesian

	Global		Local
x:	0	x:	0
y:	24	y:	24
z:	0	z:	0

Label

Apply Reset Close

Transcript

```
    toto.trn
    toto.dbs
    toto.jou
Command> vertex create coordinates 0 12 0
Created vertex: vertex.1
```

Command: / The second number is the design variable DV4

Description

HISTORY WINDOW- Displays commands executed by GAMBIT during the current modeling session, as well as messages related to the commands.

The extraction of the GAMBIT journal file for this coordinate is given by

```
vertex create coordinates 0 12 0
 / The second number is the design variable DV4
 vertex create coordinates 0 24 0
```

The line that begins with the “/" character is a *user-comment* line. If the value you enter does not have enough digits for the numerical value (two in this case), there will not be enough digits to replace the value. In this case, you will need to edit the journal file to add zeros after the value of the design variable.

The extraction of the GAMBIT journal file with enough digits for the replacement operation is as follows:

```
vertex create coordinates 0 12 0
 / The second number is the design variable DV4
 vertex create coordinates 0 24.00000000000000000 0
```

To clean the journal file using GAMBIT, remove the SAVE and UNDO commands, but not the comments. To avoid printing all of the commands during the GAMBIT process, add the following line to the beginning of the journal file as the first command:

```
default set "GUI.GENERAL.TRANSCRIPT" numeric -1
```

B.3.3. Defining a Response in ANSYS POLYDATA

You can define the responses in the **Optimization** menu in ANSYS POLYDATA and select among the primary quantities (velocity, temperature, pressure, or coordinates) or postprocessor quantities (local shear-rate, viscosity, dissipated power, flow balance, etc.). For the quantities varying over the domain, you will need to specify the location of the point where the response will be evaluated. (The response will be evaluated at the closest node of the given point.) This node can change in a moving mesh.

ANSYS POLYDATA adds a Pass/Fail response to the data file automatically. Pass/Fail are special responses that are used to form Boolean (True/False) types of constraints and are used to indicate whether or not ANSYS POLYFLOW was successfully completed.

Pass/Fail responses force the Optimizer not to consider points where ANSYS POLYFLOW cannot be completed successfully. If ANSYS POLYFLOW obtains a solution, Pass/Fail responses are assigned the value of -1. Otherwise, they take the value 1. To take into account a Pass/Fail response in VisualDOC, you will need to define a constraint based on the Pass/Fail response with an upper value of 0.

- If ANSYS POLYFLOW is successfully completed, the Pass/Fail response is set to -1 and the constraint is not violated.
- If ANSYS POLYFLOW is not successfully completed, the Pass/Fail response is set to 1 and the constraint is violated.

B.3.4. File Management for Optimization

A Perl script is used to move, copy, or delete certain files created by ANSYS POLYFLOW. This script creates one or more back-up directories. At each optimization step, it also saves files in those directories. The script file named `manag_ files.pl` is created if it does not exist in the working directory. All instructions are contained within the script file under the USER AREA section.

By default you can move, copy, or remove files for the sake of file management. The files that are to be removed are in the working directory. They are removed from the working directory and are not saved.

The files that are to be moved are saved into a back-up directory. The files that can be moved are:

- graphical files (CFD-Post, FieldView, FLUENT/Post)
- listing files
- results files if they are not read by ANSYS POLYFLOW to restart the computation

The files that are to be copied are saved to a back-up directory and remain in the working directory because they will be used again. The files that can be copied are:

- results files if they are read by ANSYS POLYFLOW to restart the computation
- input files (`dvar.vef`)
- response files (`resp.vef`)
- journal files (`*.jou`)

If you do not want to follow the existing file naming convention or if you want to save other files, you can modify the following script file. The USER AREA section of the script file is shown below:

```
#  USER AREA
#  =========
#  In this user area, please respect the following syntax:
```

```
#  For the list of files (to remove, to move or to copy), the file name is
#  given between double quotes ("), separated by a comma (,), parentheses
#  all around file names, and semi-column (;) at the end of the lines.
#  Example : ("file1", "file3", "file.dat");
#  You can use the (*) character as usually to specify a set of files
#  i.e. : *.lst, fv* or *flu*

#  Files will be moved (or copied) into a directory. The directory name or
#  file names can be indexed with respect to the optimization step index.
#  If the directory name is indexed, at each step a new directory is created
#  and files are moved (or copied):
#      Opt_res_1
#          fv.uns
#          res
#      Opt_res_2
#          fv.uns
#          res
#      ...
#  If file names are indexed, a new directory is created only once at the
#  beginning, and the files are moved (or copied) with a new name:
#      Opt_res
#          fv_1.uns
#          fv_2.uns
#          res_1
#          res_2
#       ...
#
#  Give the name of the directory to create to save the files.

$Opt_Dir = "Opt_res";
# Give the list of files to move

@FILE_TO_MOVE  = ("fv*", "*.flum", "*.flur", "*.lst");

# Give the list of files to remove

@FILE_TO_REMOVE = ("*.cnvg");

# Give the list of files to copy

@FILE_TO_COPY = ("*res", "dvar.*", "resp.*" );

# If you want index the directory, set 0 for $DirOrFile
# If you want index the files, set 1 for $DirOrFile

$DirOrFile = 1;

#  END OF USER AREA
```

B.3.5. Files Generated by an ANSYS POLYDATA Session

Some files are created at the end of an ANSYS POLYDATA session for use during the optimization task. The files that are created depend upon the type of parameterization:

- Help file, `optimization.help`:

  A help file describes how to run an optimization task with VisualDOC. The default name for the help file is `optimization.help`, though you can change the filename if desired. This file gives a checklist of the major points for the optimization and describes the procedure into VisualDOC and VisualScript.

- Console file, `optimization.cons`:

  If an optimization includes inputs related to ANSYS POLYDATA, a console file is created with the default name `optimization.cons`, though you can change the filename if desired. This console file is used to update parameters into the ANSYS POLYFLOW data file from the inputs file. The use of this file is described in `optimization.help`.

- Mesh conversion file, `NeuToMsh.conv`:

  If an optimization includes inputs for geometry, a file for the conversion of the mesh format to the ANSYS POLYFLOW mesh format is created. The default name of the conversion file is `NeuToMsh.conv`, though you can change the filename if desired. During the ANSYS POLYDATA session, you can specify the type and input file needed for conversion.

  > **Important**
  >
  > The conversion does not take place when the conversion file is created. It will be done during the optimization loop. The mesh conversion file must exist for the conversion process to take place.

  The use of the mesh conversion file is described in `optimization.help`.

B.4. Problem Setup

The basic steps for setting up an external optimization task are explained below.

1. Read the mesh.

   Read a mesh file

   The mesh that will be used during optimization should already exist. If the optimization task includes modifications to the geometry, you must either copy the initial mesh or generate the mesh for optimization before running ANSYS POLYDATA.

2. Read an old data file.

   Read an old data file

   Since an optimization task starts from an initial solution, you will need to use the initial data file. Read and adapt it for the optimization.

3. Redefine global parameters of a task to switch to optimization.

   Redefine global parameters of a task

   The **Redefine global parameters of a task** menu will open, allowing you to select **Optimization problem(s)**. Then click **Accept the current setup** to return to the previous menu.

   Note that an optimization task is not compatible with evolution or transient tasks. This is not a limitation, because optimization restarts from an initial solution and VisualDOC gets a return error message if ANSYS POLYFLOW does not converge. In this case, VisualDOC restarts with new values for the design variables.

4. Tag the inputs.

   a. In the submenu of a task (material data, flow boundary condition, thermal boundary conditions), click the **UPDT** button.

   b. When you set a parameter, the **Create template entry** menu is displayed and you can create a template entry by selecting the **Create a new template entry** menu.

   Create a new template entry

   c. A template entry is created with a default comment of the parameter name. You can change the name by selecting the **Modify comment** menu item:

      Modify comment

      You can delete a template entry be selecting the **Delete the current template entry** item.

      Delete the current template entry

5. Define the postprocessor sub-tasks for design variables that will be used as responses.
6. Define the optimization features by selecting the **Optimization** menu in the **Task** menu.

   Optimization

   a. Specify that the optimization is external by clicking **Switch to external optimization (VisualDOC)**.

      Switch to external optimization (VisualDOC)

      The comments at the top of the menu will display your selection: **External optimization (VisualDOC)**.

   b. Specify the parameterization features. The first three menu items are related to the parameterization type. The default is no parameterization.

      - Click **Inputs for POLYDATA: change to NEEDED** if some inputs related to ANSYS POLYDATA exist.

        Inputs for POLYDATA: change to NEEDED

        Otherwise, click **Inputs for POLYDATA: change to NOT NEEDED** to specify that no inputs related to ANSYS POLYDATA exist.

        Inputs for POLYDATA: change to NOT NEEDED

        The comments at the top of the menu will reflect your specification (i.e., either **Inputs for POLYDATA needed**or **Inputs for POLYDATA not needed**).

      - Click **Inputs for mesh generator: change to NEEDED** if some inputs related to the mesh generator (i.e., GAMBIT or another software product) exist.

        Inputs for mesh generator: change to NEEDED

        Otherwise, click **Inputs for POLYDATA: change to NOT NEEDED** to specify that no inputs related to the mesh generator (i.e., GAMBIT or another software product) exist.

        Inputs for mesh generator: change to NOT NEEDED

        The comments at the top of the menu will reflect your specification (i.e., either **Inputs for mesh generator needed** or **Inputs for mesh generator not needed**).

- Click **Other inputs (e.g., UDF files): change to NEEDED** if some inputs related to an external program (e.g., a UDF) exist.

  Other inputs (e.g., UDFfiles): change to NEEDED

  Otherwise, click **Other inputs (e.g., UDF files): change to NOT NEEDED** to specify that no inputs related to an external program (e.g., a UDF) exist.

  Other inputs (e.g., UDF files): change to NOT NEEDED

  The comments at the top of the menu will reflect your specification (i.e., either **Other inputs needed** or **Other inputs not needed**).

You can use any combination of **Inputs for POLYDATA**, **Inputs for mesh generator**, and **Other inputs (e.g., UDF files)**.

c. Select the **Define responses** menu item in the **Optimization** panel to define the response.

   Define responses

   i. Select a field to create a new response among the list of fields suggested.

   - To create a new response associated to the selected field, select:

     Create a new response

   - To modify the parameter of a response, select the **response # 1** menu item.

     response # 1

     For a field varying over the domain, the origin of the coordinate is the default position of the point where the response is evaluated. To modify it, select the **Define position** menu item and specify the coordinates. For a field constant over the domain, the position where the field is evaluated must not be specified and the **Define position** menu item is not accessible. For a vector field, you must specify the component of the vector for the response by selecting either the **Select component 1**, the **Select component 2**, or the **Select component 3** menu item.

     Select component 1

     Select component 2

     Select component 3

   - To deselect a component, select either the **Unselect component 1**, the **Unselect component 2**, or the **Unselect component 3** menu item. These are accessible only if the corresponding components have been selected. More than one component can be selected and in this case, each selected component is a response.

   - To delete a response, select the **Delete a response** menu item and select the response to delete.

   ii. Repeat the above steps for each field used by a response.

d. For the response file, the VEF format is used by default, which means that ANSYS POLYFLOW writes a response file readable by VisualDOC directly. If VisualDOC cannot find all responses in this file, you can switch to the PFL format by selecting the menu item **Response file format: change to pfl format**.

   Response file format: change to pfl format

   This menu item will then become **Response file format: change to vef format**, which you can select if you want to switch back to the VEF format. If you use the PFL format, you will need to extract the responses from this file during the VisualScript session. Change the name of the response file by selecting **Enter response file name**.

   Enter response file name

   If some inputs are related to ANSYS POLYDATA, it is recommended that you do not name the input file for ANSYS POLYDATA `dvar.vef`. To change the name of the input file for ANSYS POLYDATA, select **Enter inputs file name for POLYDATA**.

   Enter inputs file name for POLYDATA

e. If some inputs are related to a mesh generator, specify the type of conversion and the input for the conversion by selecting the **Enter neutral file to convert** menu item.

   Enter neutral files to convert

   Then, specify the conversion type and the name of the input file for the conversion. The filename of the new mesh will be the same one used by ANSYS POLYDATA.

7. Continue the ANSYS POLYDATA session. Change the convergence criteria and the number of iterations.

8. Save and exit the ANSYS POLYDATA session. Specify the name of the console file for optimization, the mesh conversion file, and the help file, if it is required.

Following the ANSYS POLYDATA session, launch VisualDOC and follow the instructions in the help file for optimization. An example of the help file is fully described in Example 79 of the ANSYS POLYFLOW Examples Manual.

Bibliography

[1] H. A. Barnes, J. F. Hutton, and K. Walters.*An Introduction to Rheology*. Elsevier. 1989.

[2] R. B. Bird, R. C. Armstrong, and O. Hassager.*Dynamics of Polymeric Liquids*. John Wiley. 1987.

[3] R. B. Bird, P. J. Dotson, and N. L. Johnson.*J Non-Newtonian Fluid Mech*. 7:213. 1980.

[4] R. B. Bird, W. E. Stewart, and E. N. Lightfoot.*Transport Phenomena*. John Wiley & Sons. 1960.

[5] R. S. Chambers. *Numerical Integration of the Hereditary Integrals in a Viscoelastic Model for Glass*. *J Am Ceram Soc*. 75(8). 2213-2218. 1992.

[6] N. Clemeur, R. P. G. Rutgers, and B. Debbaut.*On the evaluation of some differential formulations for the pompom constitutive model*. *Rheol Acta*. 42(1). 217-231. 2003.

[7] P. Coussot, A. I. Leonov, and J. M. Piau.*Rheology of concentrated dispersed systems in a low molecular weight matrix*. *J Non-Newtonian Fluid Mech*. 46. 179-217. 1993.

[8] B. Debbaut and M. J. Crochet.*"Extensional effects in complex flows"*. *J Non-Newtonian Fluid Mech*. 30. 169-184. 1988.

[9] A. K. Doufas, A. J. McHugh, and C. Miller. *Simulation of melt spinning including flow-induced crystallization Part I: model development and predictions*. *J Non-Newtonian Fluid Mech*. 92. 27-66. 2000.

[10] M. Fortin. *Old and New Finite Elements for Incompressible Flows*. *Int Numerical Methods Fluids*. 1. 347-364. 1987.

[11] G. S. Fulcher. *J Am Ceram Soc*. 8(6). 339-355. 1925.

[12] M. Goldstein and J. R. Howell.*Boundary Conditions for the Diffusion Solution of Coupled Conduction-Radiation Problems*. *Technical Report NASA-TN-D-4618*. NASA Lewis Research Center. 1968.

[13] A. Goublomme, B. Draily, and M. J. Crochet.*Numerical Prediction of Extrudate Swell of a High-Density Polyethylene*. *J Non-Newtonian Fluid Mech,*. 44. 171-195. 1992.

[14] R. Guénette and M. Fortin.*A new mixed finite element method for computing viscoelastic flow*. *J Non-Newtonian Fluid Mech*. 60(1). 27-52. 1995.

[15] J. R. Howell and M. Goldstein.*Effective Slip Coefficients for Conduction-Radiation Problems*. *J Heat Transfer*. 91(1). 165-169. 1969.

[16] N. J. Inkson, T. C. B. McLeish, O. G. Harlen, and D. J. Groves. *Predicting low density polyethylene melt rheology in elongational and shear flows with “pom-pom" constitutive equations*. *J Rheol*. 43(4). 873-89. 1999.

[17] C. Johnson.*Numerical Solutions of Partial Differential Equations by the Finite Element Method*. Cambridge Univ Press. 1987.

[18] A. I. Leonov.*Analysis of simple constitutive equations for viscoelastic liquids*. *J Non-Newtonian Fluid Mech*. 42. 323-350. 1992.

[19] A. Markovsky and T. F. Soules.*An Efficient and Stable Algorithm for Calculating Fictive Temperatures*. *Comm Am Ceram Soc*. 67. C-56-C-57. 1984.

[20] T. C. B. McLeish and R. C. Larson. *Molecular constitutive equations for a class of branched polymers: The pom-pom polymer*. *J Rheol*. 42(1). 82-112. 1998.

[21] O. S. Narayanaswamy. *A Model of Structural Relaxation in Glass*. *J Am Ceram Soc*. 54(10). 491-498. 1971.

[22] D. Rajagopalan, R. C. Armstrong, and R. A. Brown*J Non-Newtonian Fluid Mech*. 36. 159-192. 1990.

[23] Y. Saad and M. H. Schultz. *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*. *SIAM J Sci Stat Comput*. 7. 856-869. 1986.

[24] R. Siegel and J. R. Howell. *Thermal Radiation Heat Transfer*. McGraw-Hill, New York. 1981.

[25] M. Simhambhatla and A. I. Leonov.*On the rheological modeling of filled polymers with particle-matrix interactions*. *Rheol Acta*. 34. 329-338. 1995.

[26] M. V. Simhambhatla. *The rheological modeling of simple flows of unfilled and filled polymers*. *PhD thesis*. University of Akron, Akron, Ohio. 1994.

[27] K. Stueben. *Algebraic Multigrid (AMG): An Introduction with Applications*. *GMD Report*. 53. 1999.

[28] J. F. Thompson et. al..*Numerical Grid Generation: Foundations and Applications*. Elsevier. 1985.

[29] A. Ungan and R. Viskanta.*Effects of Air Bubbling on Circulation and Heat Transfer in a Glass Melting Tank*. *J Am Ceram Soc*. 69. 382-391.

[30] H. A. van der Vorst. *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems*. *SIAM J Sci Stat Comput*. 13. 631-644. 1992.

[31] J. J. Van Schaftingen and M. J. Crochet.*Int J Numerical Methods Fluids*. 4. 1065-1081. 1984.

[32] M. H. Wagner. *Constitutive Analysis of Uniaxial Elongational Flow Data of Low-Density Polyethylene Melt*. *J Non-Newtonian Fluid Mech*. 4. 39-55. 1978.

[33] H-H. Yang and I. Manas-Zloczower. *Analysis of mixing*. *Int Polymer processing IX*. 1994.

Index