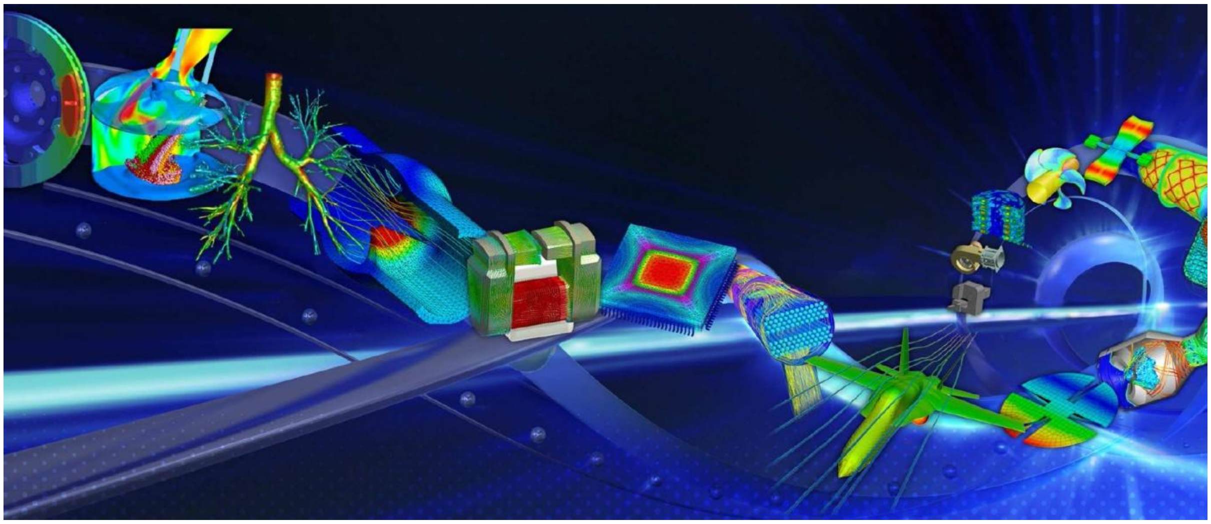


AUTODYN® Release 14.0
User Subroutines Tutorial

Release 14.0 P2
November 2011

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

Copyright and Trademark Information

Disclaimer Notice

U.S. Government Rights

Third Party Software

Published in the U.S.A.

TABLE OF CONTENTS

Published: 2011-10-05

Published: 2011-10-05

Preface

AUTODYN Tutorial Manuals

AUTODYN tutorial manuals provide detailed tuition on particular features available in the program. The manuals assume that you proficient in setting up, reviewing, executing, and post processing data for simple problems such as the one presented in the AUTODYN demonstration manual. If you have worked through the problem in the demonstration manual, you should have no problem following the tutorials.

Most tutorials are interactive and you are expected to have access to AUTODYN while working through the tutorial. Some tutorials have associated files that contain sample calculations used in the tutorial.

Existing manuals are continuously updated and new manuals are created as the need arises, so you should contact ANSYS if you cannot find the information that you need.

Published: 2011-10-05

This page left intentionally blank

Published: 2011-10-05

1. Introduction

This manual shows you how to create and use your own user subroutines in AUTODYN. Topics covered include:

1. How to invoke the User Subroutines from Input.
2. Compiling and linking user subroutines
3. Writing your own user subroutines
4. Description of AUTODYN module variables

AUTODYN provides you with a number of standard alternatives for options such as Equations of State, Yield Models, Boundary conditions, etc. However, you may wish to use your own custom models for these options. AUTODYN allows you to do this by including your own subroutines written in Fortran. This tutorial shows you how to include these subroutines in your calculations and offers guidelines on writing user subroutines. Currently, user subroutines may be included for the following.

AUTODYN User Supplied Extra Routines

Material Modeling User Subroutines	
MDEOS_USER_1	Custom equation of state (Previously EXEOS)
MDSTR_USER_1	Custom yield and/or shear model (Previously EXYLD)
MDFAI_USER_1	Custom failure criteria (Previously EXFAIL / EXFAILS)
MDERO_USER_1	Custom erosion criteria (Previously EXEROD)
EXBULK	Custom bulk modulus for a linear EOS
EXCOMP	Custom porous compaction curve, P-α equation of state
EXCRCK	Custom tensile crack softening rate
EXDAM	Custom damage parameter
EXPLRN	Custom plastic flow return algorithm
EXSHR	Custom shear modulus
EXSTIF	Custom stiffness matrix, orthotropic-elastic with failure
EXTAB	Custom tabulated saturation curve for two-phase EOS

Additional User Subroutines	
EXACC	Apply user defined acceleration to a Lagrangian node
EXALE	Custom ALE (Arbitrary Lagrange Euler) grid motions
EXEDIT	Custom edits
EXFLOW	Custom Euler flow boundary
EXFRICTION	User defined friction
EXLOAD	Loading additional, non-standard data from "SAVE" files
EXPOR	Custom variable polygon porosity
EXSAVE	Saving additional, non-standard data to "SAVE" files
EXSIE	Custom energy deposition
EXSTR	Custom stress boundary condition
EXVAL	Custom initial conditions
EXVEL	Custom velocity boundary condition
EXZONE	Custom nodal coordinates
EXORTHO_AXES	Custom define initial material axes for orthotropic materials

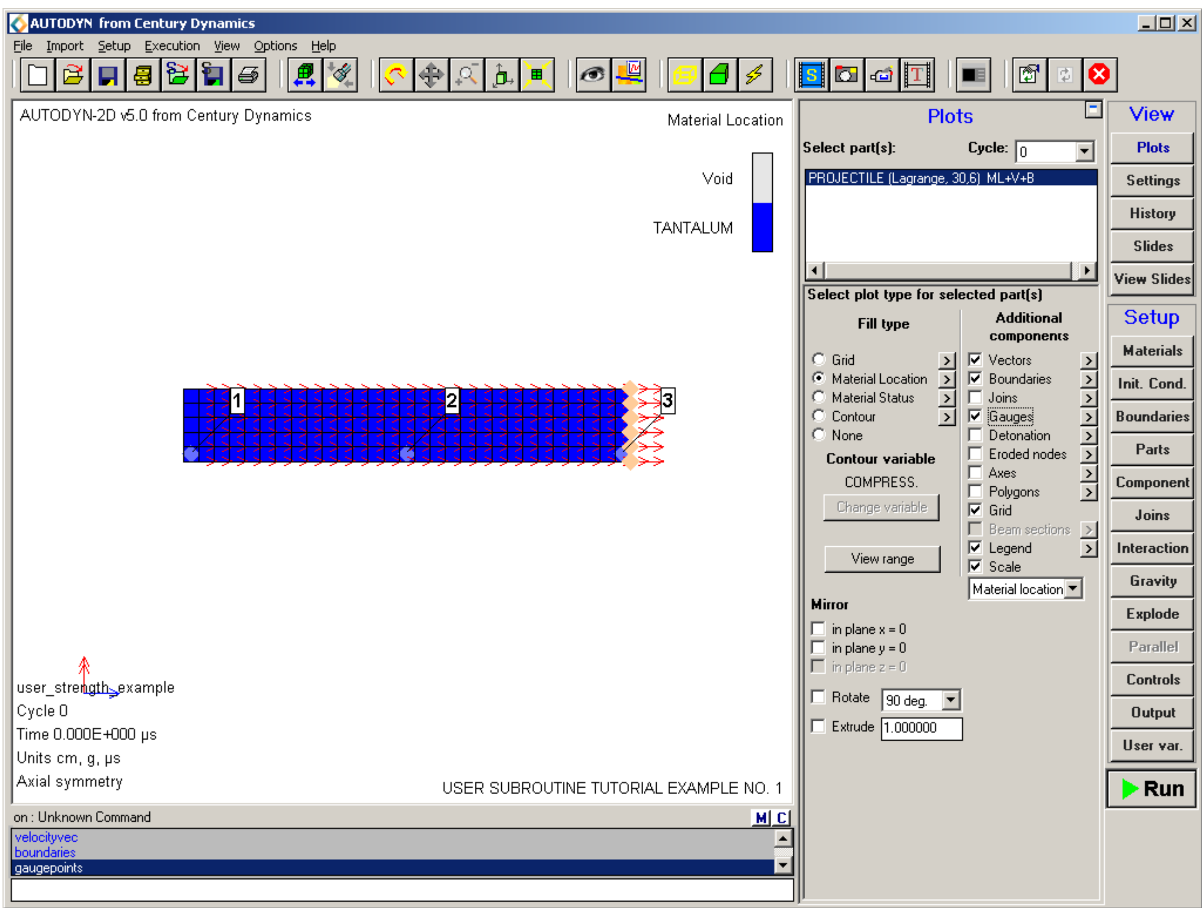
2. How to Invoke User Subroutines

To explain how user subroutines are invoked in AUTODYN we will look at a specific example. Most user subroutines require a specification of “user” for a particular input specification. However, some user subroutines are always called as discussed in chapter 3.8 **Timing of calls to user subroutines**.

2.1. Specific Example

Activate AUTODYN on your computer and from the main menu load cycle zero of the problem with ident "user_strength_example.ad".

Use the options on the Plots menu to see the material locations and boundary conditions for the problem. You will see that the problem consists of a tantalum cylinder impacting a rigid wall:



Published: 2011-10-05

Now select the Materials menu, and "Review" the material data for "TANTALUM". You will notice that a "Von Mises" yield model has been specified for this material. This model allows you to define a constant yield stress.

CENTURY DYNAMICS
Solutions through Software

ANSYS AUTODYN

Materials in model (click for shortcut)

TANTALUM	-	-	-	-	-

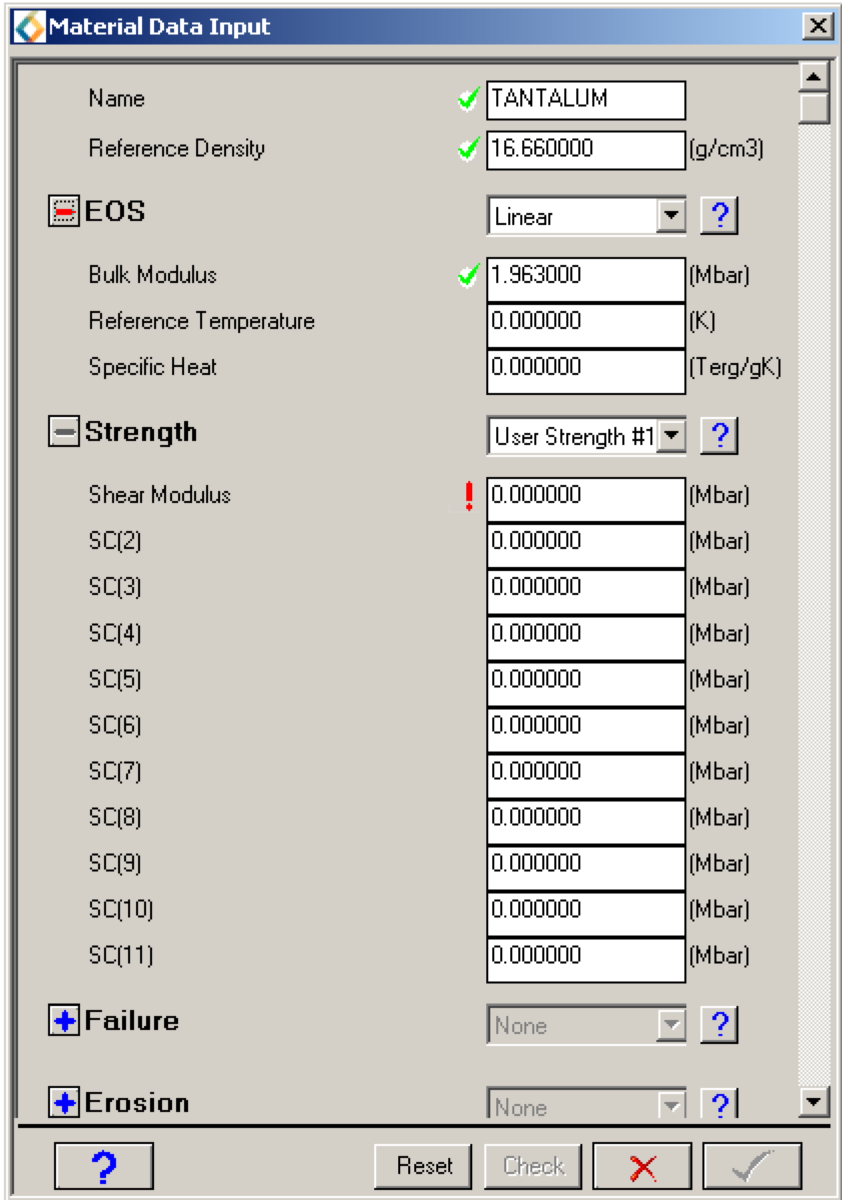
Material Name - TANTALUM

Equation of State	**Linear**
Reference density	1.66600E+01 (g/cm3)
Bulk Modulus	1.96300E+00 (Mbar)
Reference Temperature	0.00000E+00 (K)
Specific Heat	0.00000E+00 (Terg/gK)
Thermal Conductivity	0.00000E+00 (Terg/cmKus)
Strength	**von Mises**
Shear Modulus	6.92000E-01 (Mbar)
Yield Stress	9.00000E-03 (Mbar)
Failure	**None**
Erosion	**None**
Material Cutoffs	-
Maximum Expansion	1.00000E-01 (none)
Minimum Density Factor (Euler)	1.00000E-04 (none)

Suppose that instead of keeping it constant (Von Mises), we wish to make the yield stress a function of the effective plastic strain.

We first select the Modify button. In the resulting dialog, press the "+" next to the Strength option and inspect the available options for yield models, we will see the option, Piecewise, exists which could be used to describe such a relationship. However, in the interests of illustrating user subroutines we will create our own subroutine to effect the same thing. Therefore we will supply our own custom yield model. To do this, change the selected yield option from "Von Mises" to "User Strength #1":

Published: 2011-10-05

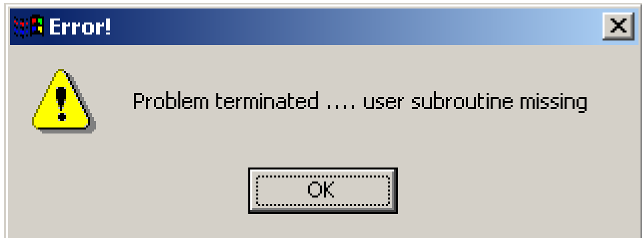


The window above appears which allows for the input of variables, Shear Modulus (SC(1)), and SC(2) through SC(11) to be used in the MD_STR_USER_1 module. Note that these pre-defined parameters are only present for demonstration and can be tailored in terms of number of parameters, names, units. It is also possible to add your own option list and parameters from many of the existing standard AUTODYN strength models.

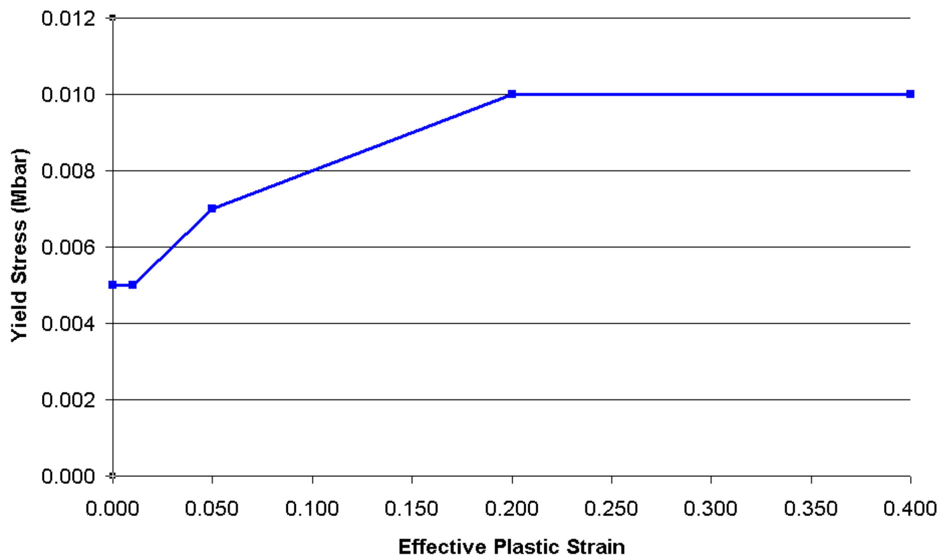
The use of the input parameters allows you to code and link your user subroutine once and then simply change variable input values through AUTODYN input. The user specifies the usage of these parameters in the MD_STR_USER_1 module.

For the moment, should provide a shear modulus, of 0.692 as before with the Von Mises model to enable us to close and save the data for the Tantalum material.

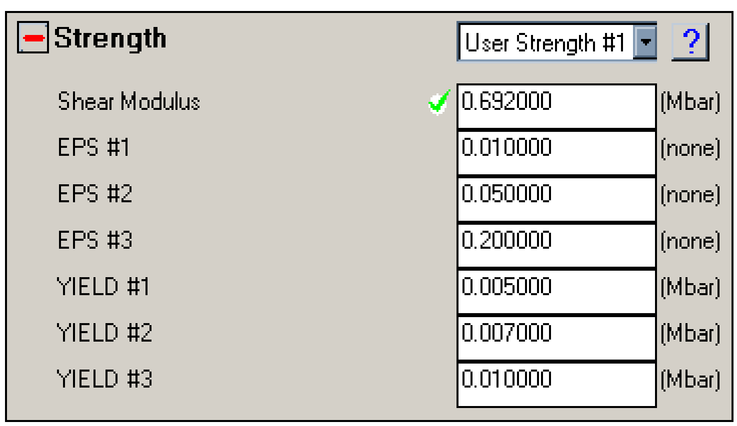
If we save the example database as “example_usersub_1” and press Run. We are presented with the error dialog:



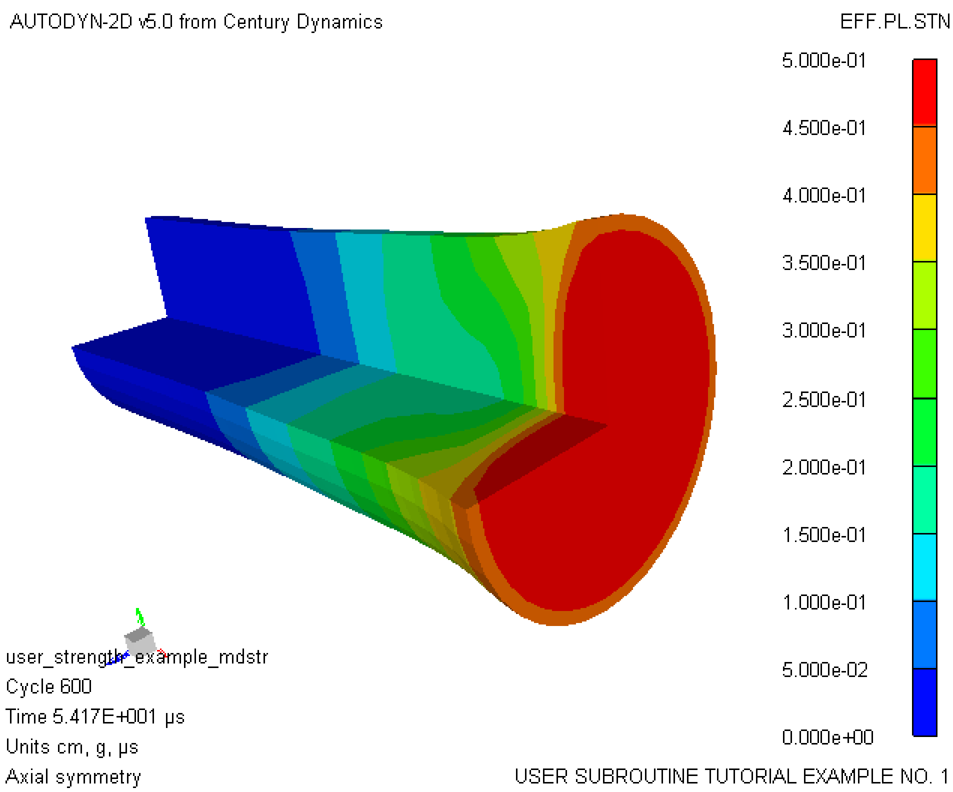
The sections that follow describe how to write user subroutines to allow execution of the example above and others. Appendix C contains an example user subroutine MDSTR_USER_1.f90 that implements the simple piecewise linear variation of yield stress against effective plastic strain as shown below.

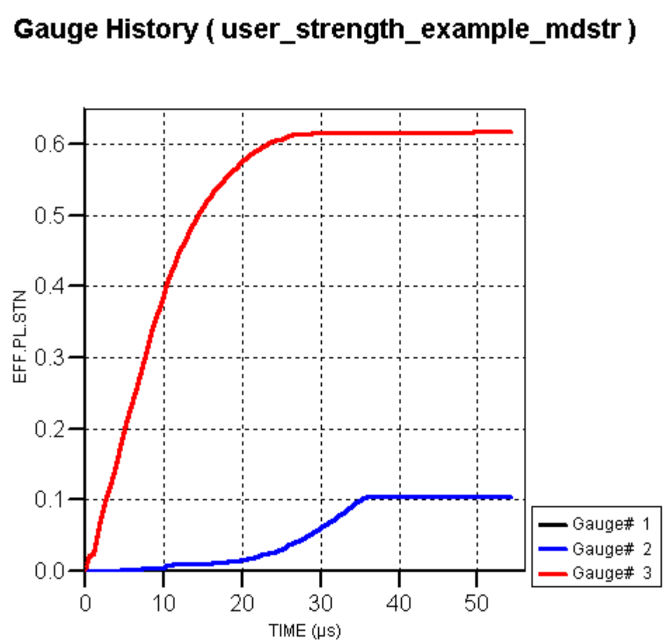


Following implementation of the subroutine shown in Appendix C we can again execute the user AUTODYN software and modify the material "TANTALUM". The material menu for the Strength model data will then appear as follows. Note that the user defined parameters set in the user subroutine now appear in the material parameter list.



Enter the material data shown above and run the analysis. A contour plot of effective plastic strain at the wrapup cycle of 600 and a gauge point history plot of effective plastic strain are shown below.





Published: 2011-10-05

3. Writing Your Own User Subroutines

There are no absolute rules for writing your own user subroutines. What we will do here is to outline some of the techniques that may be used and the tools that are available to help create user subroutines. If we look at the examples given in Appendices C-H, we will see some of the common techniques and tools used to create user subroutines. These include:

- Formal parameters passed to and from the user subroutine
- Accessing global variables in the AUTODYN modules (e.g. time, material data, cycle number, etc.)
- Accessing Part grid variables (e.g. pressure, density, velocity, etc.)
- Calling AUTODYN utility functions/subroutines
- Terminating execution from a user subroutine

We will look at each of these items in more detail.

3.1. Compilers required to work with User Subroutines

The following compilers are required to work with user subroutines at release 12.1.

Operating System	C++	Fortran
Windows XP Pro, 32 bit Windows Vista, 32 bit	Microsoft Visual Studio C++.NET 2008	Intel Fortran 11.1
Windows XP Pro, 64 bit Windows Vista, 64 bit	Microsoft Visual Studio C++.NET 2008	Intel Fortran 11.1
RedHat EL 4 (Update5)		Intel Fortran 11.1
SUSE 10		Intel Fortran 11.1

3.2. User subroutine files

The following files are included in your distribution to facilitate the development and compilation of user subroutines:

…..\AUTODYN user_subroutines

\fsrc	
\2d\usrsub2.f90	AUTODYN-2D specific user subroutines
\3d\usrsub3.f90	AUTODYN-3D specific user subroutines
\materials\mdeos_user_1.f90	User equation of state
mdstr_user_1.f90	User strength model
mdfai_user_1.f90	User failure model
mdero_user_1.f90	User erosion model

Published: 2011-10-05

Additionally, the module file fsrc\materials\mdusersub_call.f90 has been supplied. The module in this files should not be modified.

On Windows operating systems, the following ntel/.NET project files should be used;

…..\ AUTODYN user_subroutines

\ad_usrsub.sln	Intel/.NET Solution
\ad_usrsub.vfproj	Intel/.NET Fortran project

3.3. Formal parameters

Most of the user subroutines have formal parameters. There are comments at the head of the subroutines defining each parameter, stating whether the parameter is an input parameter, an output parameter, or both. In the simplest case, the writing of a user subroutine might consist of computing the output parameter(s) from the input parameter(s) as in our MD_STR_USER_1 example (2.1). If the user subroutine requires the use of other variables that are not formal parameters then we must obtain it in some other manner. If the variable is global in nature (e.g. cycle number, material data, time, etc.) this information is obtained through the USE statement, referencing the appropriate AUTODYN module. If the desired variable is associated with a Part (e.g. cell pressures, velocities, etc.), these are obtained, most readily, by using AUTODYN supplied functions to retrieve Part data.

3.4. AUTODYN modules (non-parameter data)

If you need additional data to that supplied as formal parameters to a user subroutine, you can usually get this data directly from the AUTODYN modules: “ *.mod ”. To aid you in this task, the AUTODYN modules for AUTODYN-2D and AUTODYN-3D are described in Appendix A. To use a particular module in a user subroutine insert the statement:

USE ‘xxxxxx’

where ‘xxxxxx’ is the name of the appropriate module.

The dummy subroutines have "USE" statements already defined for the most often required global data. Also the comments indicate which variables in these modules are likely to be of use.

Note: ***Be extremely careful*** *if you choose to modify variables other than the user variables (VAR01 through VAR20) and that the modified values are consistent with the use of the variables. Global variables are used by other routines and assigning bad values to them could cause AUTODYN to crash!*

3.5. AUTODYN Variables

AUTODYN utilizes the dynamic array allocation features available through Fortran 90/95. As such, there are no fixed limits on the size of the model that you can generate. However, your computing time may be greatly limited by such factors as your machine memory.

Published: 2011-10-05

There are two basic types of data storage and access used in AUTODYN depending on whether you are using the IJK based or Unstructured solvers. These are described below.

3.5.1. IJK Solvers

For the IJK based solvers (Lagrange, ALE, Shell, Beam, Euler, Euler-FCT, Multi-Material Euler), variables in the model are accessed by pointers to dynamically allocated arrays. To assist in the retrieving and storing of grid data a number of AUTODYN functions are provided. Examples are also given in Appendix F and G for AUTODYN-2D and 3D respectively.

Grid variable arrays and pointers are defined in module ‘mdgrid’ for 2D and ‘mdgrid3’ for 3D. The listings for AUTODYN-2D and AUTODYN-3D give all grid variables and their associated pointer names. Appendix B. provides a definition of these variables. The grid arrays and pointer names are used to reference all grid variables either a.) by direct reference to arrays or b.) through use of the supplied AUTODYN functions. These functions will be described in detail in the next section. By way of illustration, if we want to access the x-velocity at a particular (I,J,K) in the currently processed Part in AUTODYN-3D, we would include the “mdgrid3” module in our subroutine:

```
USE mdgrid3
```

For node (I,J,K) of the current Part, we can obtain the index, IJK, for this node by including the statement:

```
IJK = IJKSET3(I,J,K)
```

(Note: for a shell Part, set I=1 ; for a beam Part set I=1 and J=1; for an SPH Part set I=1 and K=1; for 2D IJK=IJSET(I,J) would be used instead)

Then we can obtain the required velocity, XVEL, either directly through the array reference or by using a function:

Direct: XVEL = UXN(IJK)

Function: XVEL = GV3(NUXN,IJK)

Where NUXN is the pointer index for the X-velocity array. GV3 is an AUTODYN-3D function which retrieves the specified variable value from the current Part (the equivalent AUTODYN-2D function is simply GV. The various AUTODYN functions such as “IJKSET3” and “GV3” are described in detail later.

Published: 2011-10-05

Normally, the direct reference approach is recommended. This works best when there is a "current" Part (NSUB), see "Timing of calls" below. For references to Parts other than "current", use of the functions is recommended.

In the Euler Godunov processor in 3D, the access to the dynamic memory management is slightly different to the other processors. While, calling the function "GV3" can be done as described before, the direct access method uses Fortran90 pointers. Thus, to reference the x velocity for a 3D Euler Godunov cell:

Direct: XVEL = MTSUB(IJK)%V(NNUXN)

The variable "NNUXN" is contained in the module "mdgrid3". Any specific Euler Godunov variable index is defined by NN*name*, where *name* is the AUTODYN-3D internal variable name (see Appendix B. section B.2)

3.5.2. Unstructured Solvers/Elements

For the unstructured solvers, variables associated with nodes and elements can be accessed and stored using specific functions developed for the user:

CALL GET_ELEM_VAR(*index_elem,index_layer*)
CALL GET_NODE_VAR(*index_node,index_layer*)

> Where *index_elem* is the internal index for a given element
> *index_node* is the internal index for a given node
> *index_layer* is the layer (integration point) number for each element/node

These functions copy data from the underlying data structures to local arrays. These arrays contain the values of all possible variables. Only the variables used by the element/node identified in the above calls will actually be set. To access these local arrays you will need to include

USE mdvar_all
RVL(*index_var)* will contain all real variables for the element/node
IVL(*index_var)* will contain all integer variables for the element/node

The values of *index_var* required to access a particular variable can be can be obtained through the *Output, Save, Review variables* option in the interface.

For example, to retrieve the pressure for an element, you could use the statement

PRESSURE = RVL(IVR_PRES)

Published: 2011-10-05

The values of user variables VAR01 to VAR20 can be updated/stored by first setting the updated value in the RVL array e.g.

RVL(IVR_VAR01) = VAR01VAL

Then use the functions

CALL PUT_ELEM_VAR(*index_elem,index_layer*)
CALL PUT_NODE_VAR(*index_node,index_layer*)

This will place all the data set for user variables 1 to 20 in the array RVL and IVL back into permanent main storage for this element or node respectively.

Notes:

Do not change variables other than user variables 1 through to 20 because these changes will not be stored to underlying data structures when PUT_ELEM_VAR/PUT_NODE_VAR are called. Variables passed into the material modeling subroutines as arguments can be changed.

Calls to GET_ELEM_VAR/GET_NODE_VAR and PUT_ELEM_VAR/PUT_NODE_VAR should not be made in the subroutines contained in usersub33.f90 and usrsub2.f90 since the values retrieved will be those from the end of the previous computational cycle.

Calls to GET_ELEM_VAR/GET_NODE_VAR and PUT_ELEM_VAR/PUT_NODE_VAR should not be used in the material user subroutines for materials used to fill ANP and NBS tetrahedra, with the exception of the user erosion subroutine in mdero_user_1.f90. An example of using these calls in mdero_user_1.f90 for NBS tetrahedra is given in Appendix H.5.

If GET_ELEM_VAL is called for an element other than the one currently being processed in the solver (or for a shell sublayer that differs from the current one being processed), it will not be possible to tell whether the values in RVL for that element are from the current or the previous computational cycle.

The above methods for retrieving and updating unstructured element or node data require the global *index_elem* or *index_node* as input. For material modeling user subroutines, the index of the current element will be available via

USE mdstring
Index_elem = ELEM_NOW

For other cases, there are a number of ways in which one can obtain the internal index for an element; direct access, looping over Parts, looping over a Group. Examples of various types of element/node access are given by example in Appendix H.

3.6. Accessing multiple material data – all IJK solvers

Depending upon the AUTODYN processor (solver) being used, it is possible for a single cell (element) to contain more than one type of material. This is primarily applicable to Euler and Euler Godunov processors. In AUTODYN-2D, for multiple material cells, the standard cell values of compression, internal energy, temperature, and alpha are mass-weighted averages of multiple material values. In AUTODYN-3D, plotting options allow you to plot multiple material values for a specific material or for mass-weighted values as in 2D. Normally, if you wish to access the individual cell values of these variables for each material in the cell, you must make the following subroutine call:

CALL GETMLT(IJK,0) or GETMLT3(IJK,0)

Where IJK is the index of the cell. After the call is made, the cell variables may be accessed according to the following table:

Multi-material variable	Internal array	ML Index
Material volume fraction	CVF(1,matno)	NCVF*
Material mass	CMS(1,matno)	NCMS*
Material compression	CMU(1,matno)	NCMU*
Material internal energy	CEN(1,matno)	NCEN*
Material temperature	CTP(1,matno)	NCTP*
Material alpha	CAL(1,matno)	NCAL*
Material burn fraction	CBF(1,matno)*	NCBF*
Material damage	CDM(1,matno)*	NCDM*
Material plastic strain	CPS(1,matno)*	NCPS*

*AUTODYN-3D only

,where matno is the material number for a given material. These material numbers are assigned in sequence when defining your problem, starting with 1.

You can determine what the material number is for a given material name by testing against the MATERIALS(matno)%NAME array (see the example user subroutine listings).

In AUTODYN-3D, a function is provided, GETV3, that provides a functional method for accessing multi-material variables without use of GETMLT3. This is discussed in section 3.9 below.

Note that to save updated values of multi-material variables, a call PUTMLT3(IJK,0) must be made after setting the data in the internal arrays listed above.

Published: 2011-10-05

For Lagrange, ALE, Shell, SPH and Beam solvers of AUTODYN-3D it is also possible to access and set data in the material arrays directly using the following procedure: after setting IJK for each cell, use the call

```
ML => MTSUB(IJK)%V(1:NUMMLV)
```

This sets-up the pointer ML to look at the material data for the current cell. This data can be accessed and set directly by addressing the appropriate index of the ML pointer array. For example, to set material damage to one and internal energy to zero, we could now use

```
ML(NCDM) = 1.0
ML(NCEN) = 0.0
```

This direct method of access is significantly more efficient than using the GET/PUTMLT3 calls in AUTODYN-3D.

3.7. Material Modeling User Subroutines

The main material modeling user subroutines (MD_EOS_USER_1.F90, MD_STR_USER_1.F90, MD_FAI_USER_1.F90, MD_ERO_USER_1.F90) have been updated and modularized in AUTODYN v6.0 to allow more flexibility. Each of the above routines contains four basic components:

Routine	Description
*Nam*_USER_1	A module to allow the user to define variables that can be defined and accessed in any of the routines below (or anywhere with the USE *nam*_USER_1 statement)
Init_*nam*_USER_1	A subroutine that allows the user to define the input parameters for the material modeling option.
check_*nam*_USER_1	A subroutine that allows the user to perform checks on the input data for the user model, during input of the data.
Set_*nam*_USER_1	A subroutine to get the material input data from the internal AUTODYN data structures and assign local variables as required.
solve_*nam*_USER_1	A subroutine in which the user writes his material modeling algorithm

Further details on how to implement material modeling user subroutines are given in the example user subroutine in Appendix C.

3.8. Timing of calls to user subroutines

Depending upon the user subroutine, the routine may be called once per problem (e.g. EXLOAD), once per specified cycle (e.g. EXEDIT), or many times for each cycle and each cell. It is important to understand this calling sequence if you are to be successful in implementing your user subroutines. The calling sequence of the user subroutines may be classified according to type, as outlined in the table below:

Table 1. User Subroutine Calling Sequence Types

Type	Timing of calls
1	Called once, each time a Load or Save is requested
2	Called at each user specified cycle
3	Called for user specified material, each cell, each cycle
4	Called for particular boundary conditions, each cell on boundary, each cycle
5	Called for each cell, each cycle
6	Called for each "fill" (initialization) region, during problem set-up
7	Called for each EXZONE menu selection: Part/Zoning/Import

For Types 3-7, a "current" Part (NSUB) is defined, such that variables may be directly referenced through their array name and index (e.g. UXN(IJK)). For other types, the user is advised to use the AUTODYN utility functions to obtain variables.

The table below provides a description of when the major user subroutines are called.

Table 2. User Subroutine Calling Sequence Descriptions

Name	Description	Calling sequence type
SOLVE_EOS_USER_1	Custom equation of state	3 (called when EOS specified as "user", each cell, each cycle.)
SOLVE_STR_USER_1	Custom strength model	5 (called for strength models specified as "USER", each cell, each cycle. Also, called for all other strength models (except None), after standard calculation of yield stress, to allow for further modification of yield stress. NB. For shell elements, called once for each sublayer, for each cell, for each cycle.)
SOLVE_FAI_USER_1	Custom failure criteria	5 (called when failure model specified as "USER", each cell, each cycle Also, called for all other failure models (except None), after standard checks for failure, to allow for further modification of failure criteria. NB. For shell elements, called once for each sublayer, for each cell, for each cycle.)
SOLVE_ERO_USER_1	Custom erosion criteria	3 (called when erosion model specified as "USER", each cell,

Published: 2011-10-05

Name	Description	Calling sequence type
		each cycle.)
EXBULK	Custom bulk modulus for a linear, polynomial, and P-α EOS	5 (always called for linear, polynomial, and P-α EOS, each cell, each cycle. No other user specification required. NB. For shell elements, called once for each sublayer, for each cell, for each cycle.)
EXCOMP	Custom porous compaction curve, P-α equation of state	3 (called when compaction curve specified with P-α EOS is “USER”, each cell, each cycle)
EXCRCK	Custom tensile crack softening rate	3 (always called when crack softening specified ($G_f \neq 0.0$), each cell, each cycle. No other user specification required.)
EXDAM	Custom damage parameter	3 (always called for brittle damage and Johnson-Holmquist failure models, each cell, each cycle. No other user specification required.)
EXEDIT	Custom edits	2 (called at the end of cycle, as specified under Global /Edit/User)
EXFLOW	Custom Euler flow boundary	4 (always called for “flow in”, and “flow out with reverse” boundary conditions, for each associated boundary cell, each cycle. No other user specification required.)
EXLOAD	Loading additional, non-standard data from "SAVE" files	1 (always called when Load is selected, after standard loading completed. No other user specification required.)
EXORTHO_AXES	Custom initial orthotropic material axes	6 (always called when performing initialization. No other user specification required.)
EXPLRN	Custom plastic return algorithm	5 (each cell, each cycle, No other user specification required)
EXPOR	Custom variable polygon porosity	4 (always called for each Euler-Lagrange polygon, each cycle. No other user specification required.)
EXSAVE	Saving additional, non-standard data to "SAVE" files	2 (called for every Save, through execution of Save command, or specification of Save Edits. No other user specification required.)
EXSHR	Custom shear modulus	5 (always called, each cycle, for every “non-hydro” strength model cell, all materials. No other user specification required. NB. For

Published: 2011-10-05

Name	Description	Calling sequence type
		shell elements, called once for each sublayer, for each cell, for each cycle.)
EXSIE	Custom energy deposition	5 (always called, each cycle, for every cell, all materials. No other user specification required.)
EXSTIF	Custom stiffness matrix, orthotropic-elastic with failure	3 (called when orthotropic EOS and a strength model specified, each cell, each cycle. NB. For shell elements, called once for each sublayer, for each cell, for each cycle.)
EXSTR	Custom stress boundary condition	4 (called when stress boundary condition specified as “user”, for each associated boundary cell, each cycle)
EXTAB	Custom tabulated saturation curve for two-phase EOS	3 (called when two-phase EOS model is specified as “user”, each cell, each cycle)
EXVAL	Custom initial conditions	6 (always called when performing a “fill” initialization, during problem Create/Modify. No other user specification required.)
EXVEL	Custom velocity boundary condition	4 (called when velocity boundary condition specified as “USER”, for each associated boundary cell, each cycle)
EXZONE	Custom nodal coordinates	7 (called each time EXZONE menu selection is made.)

(Note: in situations where the extra subroutine is always called, with no “user” specification required, the default “dummy” subroutines are programmed to have no effect.)

Published: 2011-10-05

3.9. AUTODYN utility functions/subroutines

AUTODYN utility routines perform a variety of functions that can be used in user subroutines. We have already seen one in the previous section where the function "IJKSET3" was used to determine the index of the node/element (I,J,K) in the current Part. Below are further routines that may be of use:

FUNCTION IJKSET (3D only)

Usage:	IJKSET3(I,J,K)
Purpose:	Gets the IJK index of node/zone (I,J,K) relative to the current Part Note: Function IJKSETL(I,J,K) can also be used with the same effect.

FUNCTION IJSET (2D only)

Usage:	IJSET(I,J)
Purpose:	Gets the IJ index of node/zone (I,J) relative to the current Part

FUNCTION IJKSETS (3D only)

Usage:	IJKSETS3(I,J,K,N)
Purpose:	Gets the IJK index of node/zone (I,J,K) relative to all Parts, where N is the Part number (in order of definition, starting with 1). See below for example of how to obtain Part *number* from a Part *name*.

FUNCTION IJSETS (2D only)

Usage:	IJSETS(N,I,J)
Purpose:	Gets the IJK index of node/zone (I,J,K) relative to all Parts, where N is the Part number (in order of definition, starting with 1). See below for example of how to obtain Part *number* from a Part *name*.

SUBROUTINE IJANDK (inverse of IJKSET) – (3D only)

Usage:	CALL IJANDK3 (IJKIN,I,J,K) ; IJKIN is input, I,J, and K are output
Purpose:	Gets the I,J, and K indices for IJK relative to the current Part

SUBROUTINE IANDJ (inverse of IJSET) – (2D only)

Usage:	CALL IANDJ (IJIN,I,J) ; IJIN is input, I and J are output
Purpose:	Gets the I and J indices for IJ relative to the current Part

SUBROUTINE IJANDKS (inverse of IJKSETS) – (3D only)

Usage:	CALL IJANDKS3 (IJKSIN,I,J,K) ; IJKSIN is input, I,J, and K are output
Purpose:	Gets the I,J, and K indices for IJK relative to the all Parts

Published: 2011-10-05

SUBROUTINE IANDJS (inverse of IJSETS) – (2D only)

Usage:	CALL IANDJS (IJSIN,I,J,M,IJKL);IJSIN is input, I,J,M and IJKL are output
Purpose:	Gets the I and J indices for IJK relative to the all Parts. Also output is the Part number M and the local IJK for that Part.

FUNCTION GV

Usage:	GV (NV,IJK) / GV3(NV,IJK)
Purpose:	Gets the value of Part variable "NV" for the node IJK relative to current Part. See module mdgrid/mdgrid3 for a list of values for NV.

FUNCTION GVS

Usage:	GVS (NV,IJKS) / GVS3(NV,IJKS)
Purpose:	Gets the value of Part variable "NV" for the node IJKS relative to all Parts. See module mdgrid for a list of values for NV.

SUBROUTINE PUTGVS

Usage:	CALL PUTGVS(NV,IJKS,VALUE) / CALL PUTGVS3(NV,IJKS,VALUE) ; all values are input
Purpose:	Puts (stores) the "VALUE" of Part variable "NV" for the node/zone IJKS(relative to all Parts) in the dynamic storage arrays.

FUNCTION GETV (3D only)

Usage:	GETV3 (NV,IJK,MODE)
Purpose:	This is a general function that gets the variable NV for zone IJK for the current Part. Depending on the value of MODE the following actions are taken: MODE=0 : gets zonal variable NV (calls GV) MODE>0 : gets multimaterial variable NV for MAT=MODE MODE<0 : gets volume weighted average over all materials for multimaterial variable NV.

FUNCTION NPK

Usage:	NPK (NV,IJK) / NPK3 (NV,IJK)
Purpose:	Gets the value of *integer* Part variable "NV" for the node IJK relative to current Part. See module mdgrid for list of index values for NV. To be used instead of GV when the variable is an integer and not a real number.

Published: 2011-10-05

FUNCTION NPKS

Usage:	NPKS (NV,IJKS) / NPKS3 (NV,IJK)
Purpose:	Gets the value of *integer* Part variable "NV" for the node IJKS relative to all Parts. See module mdgrid for list of values for NV. To be used instead of GVS when the variable is an integer and not a real number.

SUBROUTINE PUTNPKS

Usage:	CALL PUTNPKS(NV,IJKS,NVALUE) / CALL PUTNPKS3(NV,IJKS,NVALUE) ; all values are input
Purpose:	Puts (stores) the integer "NVALUE" of Part variable "NV" for the node/zone IJKS(relative to all Parts) in the dynamic storage arrays. To be used instead of PUTGVS when the variable is an integer and not a real number.

FUNCTION GET_ELEM_ID

Usage:	GET_ELEM_ID(ID_USER)
Purpose:	Returns the internal index of user element number, ID_USER

FUNCTION GET_ELEM_VAR

Usage:	GET_ELEM_VAR(TYPE_ELEM,ELEM_NOW)
Purpose:	Copies the data from main storage into local vector RVL (real data), IVL (integer data) accessible via "USE mdvar_all". To find the index of the variable in these vectors see Appendix B.

FUNCTION PUT_ELEM_VAR

Usage:	PUT_ELEM_VAR(TYPE_ELEM,ELEM_NOW)
Purpose:	Copies the data from main storage into local vector RVL (real data), IVL (integer data) accessible via "USE mdvar_all". To find the index of the variable in these vectors see Appendix B.

SUBROUTINE ADQUIT

Usage:	CALL ADQUIT ('Message to be displayed') CALL ADQUIT(TEXT)
Purpose:	Terminates AUTODYN EXECUTION immediately.
Example:	CALL ADQUIT ('Error #1 in routine EXEOS')

SUBROUTINE GETYON

Usage:	CALL GETYON (YON, 'Question'); CALL GETYON (YON, TEXT)
Purpose:	Presents a question in the message area and awaits a yes/no answer. "YON" is the answer ("Y" or "N", no other input is accepted). Maximum text length is 80 characters.

Example:	CHARACTER*1 YON CALL GETYON (YON, 'Stop run - are you sure?') IF (YON=='Y') STOP

SUBROUTINE USR_MESSAG

Usage:	CALL USR_MESSAG ('message to be displayed'); CALL USR_MESSAG (TEXT)
Purpose:	Displays a message in the message window

SUBROUTINE USR_ERROR

Usage:	CALL USR_ERROR (‘title’,'message to be displayed'); CALL USR_ERROR (TITLE,TEXT)
Purpose:	Displays a message in the message window Eg CALL USR_ERROR(‘Warning !’,’Inconsistent strength model parameters’)

3.10. Terminating execution from a user subroutine

Sometimes you may wish to terminate execution of a calculation if an error is detected in a user subroutine. The easiest way to do this is to simply put a STOP statement in the user subroutine. This method will immediately terminate the program and return you to the operating system. However, if you do this, you may lose information contained in output buffers.

A better way to terminate execution is to call subroutine ADQUIT described in the previous section. This will return you to the operating system in an orderly manner.

If want to stop executing a problem without quitting AUTODYN you can do this by setting the wrapup switch, “NSWRAP” equal to 99. A non-zero value of NSWRAP (found in module “WRAPUP”) will cause AUTODYN to stop execution at the end of the current cycle and return the user to the main menu. If NSWRAP is set equal to 99, the message “Problem terminated by user subroutine” is displayed upon wrapup. Since the calculation will continue to the end of the cycle, it is necessary to set the return parameters of your user subroutine to reasonable values so that they can be used, if necessary, without consequence for the current cycle.

3.11. How to determine the Part number from the Part name

Sometimes it may be desirable to know the Part number for a particular Part. Part numbers are assigned in their order of definition, starting with one. Structured and Unstructured Parts are contained and stored in different constructs;

Structured Parts:

If you want the Part number associated with a specific Part name, the following coding will obtain that number (NSB) :

```
USE SUBDEF

INTEGER (INT4) :: NS, NSB
NSB = 0
DO NS = 1, NUMSUB
   IF (NAMSUB(NS) /= 'Part name') CYCLE
   NSB = NS
   EXIT
END DO
 !        ERROR, PART NOT FOUND
IF (NSB= = 0) CALL USR_ERROR ('ERROR !', 'PART NOT FOUND')
```

If you are writing a user subroutine called within the computational cycle (types 3 and 5), the current Part number (NSUB) will already be set (module LOCSUB). You can then simply test the Part names, e.g. NAMSUB(NSUB), to determine if the Part is the one you wish to perform some action on.

Unstructured Parts:

```
USE mdpart
INTEGER(INT4) :: NPART, NPART_WANTED

NPART_WANTED = 0
DO NPART = 1, NUM_PARTS
  IF (PARTS(NPART)%P%NAME/='Part name') CYCLE
  NPART_WANTED  = NPART
END DO

!        ERROR, PART NOT FOUND
IF (NPART_WANTED= = 0) CALL USR_ERROR ('ERROR !', 'PART NOT FOUND')
```

3.12. Variables available through F90 modules

The primary modules of interest for writing user subroutines are provided (*.mod files) for 2D and 3D. The descriptions of the variables in the modules most likely to be utilized in user subroutines are given in Appendix A for AUTODYN-2D and AUTODYN-3D. Appendix B. provides a further description of the grid variables.

4. Compiling and Running Your User Subroutines

The procedure for linking your own user subroutines into AUTODYN varies according to the system on which you are running. The general procedure is to edit the existing dummy user subroutines, modifying the appropriate subroutine to implement your modifications. By following the instructions that follow for your platform, a customized AUTODYN version can be created.

4.1. Compiling, debugging and running your customized AUTODYN version on Microsoft Windows

We recommend that you use the supplied Microsoft .NET development environment solution file (ad_usrsub.sln) for compiling, debugging and linking user subroutines with AUTODYN. This solution file and other user subroutine files will be contained in the directory that would have been setup when AUTODYN was first run on your system. By default, this directory is located in the 'My Documents' folder, eg:

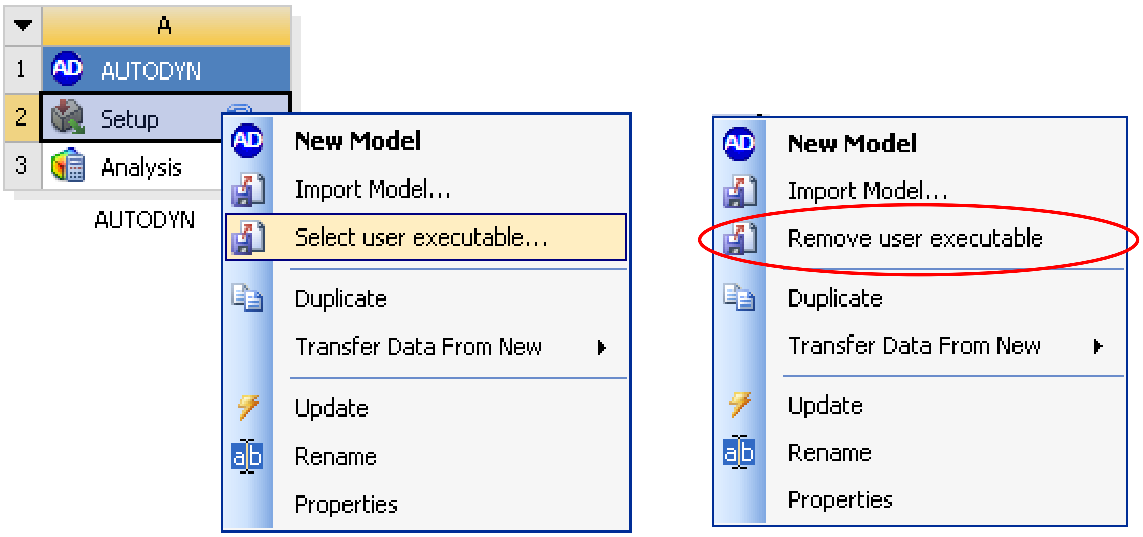
C:\Documents and Settings\A User\My Documents\AUTODYN user_subroutines

Once the solution is loaded into the development environment, changes to the user subroutine files can be made and the release and debug customized AUTODYN versions can be compiled in both single and double precision configurations.

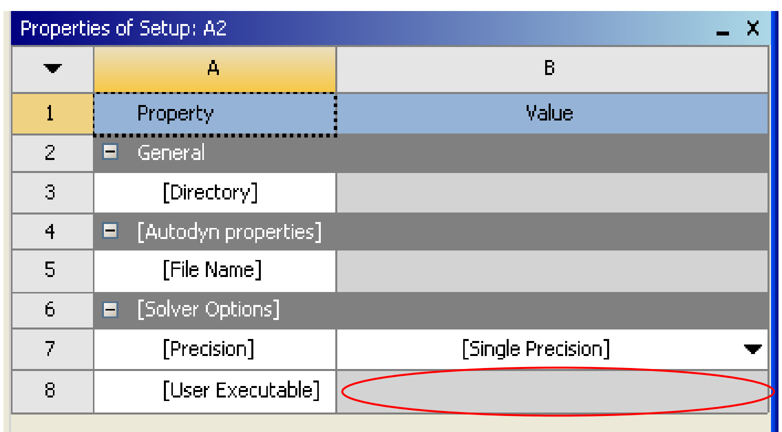
In order to run your customized AUTODYN version from within Workbench, you should set the preferences within Workbench as follows:

- Right click on AUTODYN system's **Setup** cell and from the drop down menu that appears choose **'Select User Executable'**. Navigate to the user compiled executable you wish to use for this AUTODYN system.
- To deselect your own user executable and to run the standard AUTODYN release version again, right click the **Setup** cell of the AUTODYN system and from the drop down menu that appears choose **'Remove user executable'**.

Published: 2011-10-05



You can confirm which executable will be used for a given AUTODYN system by right click on the system **Setup** cell and select **Properties.**



If you require debugging of your customized AUTODYN version, you must run the program from outside of the Workbench environment. The development environment solution is setup to allow debugging of the executables, ensure that the solution configuration is set to Debug and select Start debugging from the Debug drop down window.

4.2. Linking your own user subroutines on Linux platforms

The user-subroutine directory of the Linux installation package can be found under the platform specific directory in the autodyn/usrsub directory. For example, if autodyn has been installed into the default directory, then the files required to create a customized version would be found in:

/ansys_inc/v130/autodyn/usrsub/<platform>

Where the platform directory is either: linia32 (32bit Linux), linia64 (64bit Itanium II Linux), linop64 (64bit AMD64 Linux) or linem64t (64bit Intel EM64T Linux)

We recommend that this whole directory and it's sub-directories are copied to your home directory to avoid any permissions conflicts when editing and compiling the customized versions on Linux.

The files for editing are contained within this directory, and the libraries needed for compilation contained in the Module sub-directory. A script is supplied for compilation of customized AUTODYN executables, to execute this type ./autolnk within the user subroutine directory. This will compile each user subroutine FORTRAN file in turn, and then link with the AUTODYN library to produce a customized AUTODYN executable (and slave executable). Before running the *autolnk* script, the environment variable MPI_ROOT should be set, or the path to the HPMPI directory set in the script. By default, HP-MPI is installed into the directory:

/ansys_inc/v130/ansys/hpmpi/<platform>

In order to run your customized AUTODYN executable, you should set the environment variable CUSTOMIZED_AUTODYN to be the full path to the location of the customized AUTODYN executables.

For example, for a customized AUTODYN executable created in a user's directory /home/autodyn_user/autodyn/customize, using the C shell, the environment variable is defined by:

setenv CUSTOMIZED_AUTODYN /home/autodyn_user/autodyn/customize

Once this environment variable is set, run the standard AUTODYN script (located by default in */ansys_inc/v130/autodyn/bin*) and the customized executable will be run from the path defined by the variable.

In order to return to the standard AUTODYN executable, you should be sure to unset the variable by typing:

unsetenv CUSTOMIZED_AUTODYN

To run a customized parallel AUTODYN simulation, the parallel configuration file (parallel.cfg) should contain the paths for the customized AUTODYN executables. Located in the same directory as the customized AUTODYN executables is a script called ADSLAVE_EXE. This script sets up the environment variables at runtime required for the slave to execute. The parallel.cfg file for customized parallel calculation should contain the

Published: 2011-10-05

path to this script. For further information regarding the parallel configuration files, please see the Parallel Documentation.

Published: 2011-10-05

Appendix A. AUTODYN Modules

AUTODYN modules contain most of the problem variables that a user might require in order to write a user subroutine. The sections below provide a listing and description of the primary variables of interest. Note that the listing below is not inclusive of all the AUTODYN modules in the program. Also, note that the modules are delivered in compiled form and therefore cannot be read as text.

Warning: The user should be very careful about changing the values contained in the standard AUTODYN modules. Such actions may cause unpredictable results.

A.1. BNDDEF, Boundary Definitions

```
MODULE bnddef
USE kindef
IMPLICIT NONE
SAVE

INTEGER, PARAMETER :: LIMBDY=200
INTEGER, PARAMETER :: LIMBDC=20
INTEGER (INT4) :: NUMBDY
INTEGER (INT4) ::  IFLIMX,  IFLIMY,  IFLIMZ
INTEGER (INT4), DIMENSION(LIMBDY), TARGET :: NBDTYP, IVB
REAL (REAL8) ::  XMIND,  XMAXD,  YMIND,  YMAXD,  ZMIND,  ZMAXD
REAL (REAL8), DIMENSION(LIMBDC,LIMBDY), TARGET :: RVB
CHARACTER (LEN=10), DIMENSION(LIMBDY), TARGET :: NAMBDY

END MODULE bnddef
```

LIMBDY	Limit on number of boundary conditions
LIMBDC	Limit on number of parameters stored for each boundary condition
NUMBDY	Number of boundary conditions
NBDTYP	Boundary condition types
IVB	Material number for Euler Flow and Transmit boundaries
RVB	Boundary condition parameters
NAMBDY	Boundary condition names

A.2. CYCVAR, Cycle Variables

```
MODULE cycvar
USE kindef
IMPLICIT NONE
SAVE

INTEGER (INT4) :: NCYCLE, IDTCAL, MTSTEP, ITSTEP, JTSTEP, KTSTEP
INTEGER (INT4) :: NCYBEG, NRSCYC,   MDELS,   JDELS
REAL (REAL8) :: TIMB, TIME, DLTB, DLTH, DLTE, DLTMIN, DLTMAX
REAL (REAL8) :: DTFRAC, SSSTEP, SSSTAB, DRSTAB, VLSTAB, DVSTAB
REAL (REAL8) :: DLTHOL, CSSTEP,  DTMIN
INTEGER (INT4) :: FCTSTEP
INTEGER (INT4) :: NCYCLEEUL, IDTCALEUL, MTSTEPEUL, ITSTEPEUL, JTSTEPEUL,
KTSTEPEUL
INTEGER (INT4) :: NCYBEGEUL, NRSCYCEUL,   MDELSEUL,   JDELSEUL
REAL (REAL8) :: TIMBEUL, TIMEEUL, DLTBEUL, DLTHEUL, DLTEEUL, DLTMINEUL, DLTMAXEUL
REAL (REAL8) :: DTFRACEUL, SSSTEPEUL, SSSTABEUL, DRSTABEUL, VLSTABEUL,
DVSTABEUL
REAL (REAL8) :: DLTHOLEUL, CSSTEPEUL,  DTMINEUL

END MODULE cycvar
```

NCYCLE	Current cycle number
IDTCAL	Not available
MTSTEP	Part number controlling timestep
ITSTEP	I-index controlling timestep
JTSTEP	J-index controlling timestep
KTSTEP	Not used in 2D
NCYBEG	Starting cycle for current segment of calculation
NRSCYC	Cycle number for which "SAVE" file is to be loaded (passed to "GETRST")
TIMB	Time at beginning of cycle: t(n)
TIME	Time at end of cycle: t(n+1)
DLTB	Timestep from t(n-1/2) to t(n+1/2)
DLTH	Timestep from t(n) to t(n+1)
DLTE	Timestep from t(n+1/2) to t(n+3/2)
DLTMIN	Minimum timestep
DLTMAX	Maximum timestep
DTFRAC	Timestep stability factor
SSSTEP	Stability timestep
SSSTAB	Soundspeed in cell controlling timestep
DRSTAB	Cell dimension in cell controlling timestep
VLSTAB	Cell velocity in cell controlling timestep
DVSTAB	Cell divergence in cell controlling timestep
DLTHOL	Previous timestep, DLTH

Published: 2011-10-05

A.3. FILDEF, File Definitions

Module fildef defines the various file unit numbers. Typically, the only information important for the user is that NUT8 is the AUTODYN log file, in case the user wishes to write something to the log file.

```
MODULE fildef
USE kindef
IMPLICIT NONE
SAVE

INTEGER, PARAMETER :: LIMUNT=3
INTEGER, PARAMETER :: NUT1=31, NUT2=32, NUT3=33, NUT4=34, NUT5=35
INTEGER, PARAMETER :: NUT6=36, NUT7=37, NUT8=38, NUT9=39,NUT10=40
INTEGER, PARAMETER ::NUT11=41,NUT12=42,NUT13=43,NUT14=44,NUT15=45
INTEGER, PARAMETER ::NUT16=46,NUT17=47,NUT18=48,NUT19=49,NUT20=50
INTEGER, PARAMETER ::NUT21=51, NUT22=52
INTEGER, PARAMETER :: NOLD=0,NNEW=1,NAPP=2,NUNF=0
!         UNFORMATTED FILE FORMATS
!                        NUNF =  0  DEFAULTS TO 'BIG ENDIAN'
!                       NULND = -1 'LITTLE_ENDIAN'
!                       NUCRY = -2 'CRAY'
!                       NUFDX = -3 'FDX'
!                       NUFGX = -4 'FGX'
!                       NUIBM = -5 'IBM'
!                       NUVXD = -6 'VAXD'
!                       NUVXG = -7 'VAXG'
!                       NUNAT = -8 'NATIVE'
INTEGER, PARAMETER :: NULND=-1,NUCRY=-2,NUFDX=-3,NUFGX=-4
INTEGER, PARAMETER :: NUIBM=-5,NUVXD=-6,NUVXG=-7,NUNAT=-8
INTEGER, PARAMETER :: NFOR=1,NUNK=2,NSEQ=0,NDIR=1
INTEGER (INT4) :: IFBINI, IFBINO, LOGFILE, NUNIT, IRDOLY
CHARACTER (LEN=4), PARAMETER :: ADHLP1 = 'AD21', ADHLP2 = 'AD22'
CHARACTER (LEN=9), PARAMETER :: FNEW='unknown',FOLD='old',FAPP='append'
CHARACTER (LEN=1)  :: SLASH
CHARACTER (LEN=3)  :: FEXT
CHARACTER (LEN=256):: FNID
CHARACTER (LEN=6)  :: IOUNIT
CHARACTER (LEN=10) :: FNREST, FNHIST, FNPRNT = 'PRT', WRITESW
CHARACTER (LEN=80) :: FDPLOT, FDREST, FDHIST, FDPRNT, FDSLID
CHARACTER (LEN=80) :: FDHELP, FDMTRL, FDMCRO, FDTEMP, FDBIN
CHARACTER (LEN=256) :: FNAME, FDUMMY,FDADI, FNUNIT

END MODULE fildef
```


Published: 2011-10-05

A.4. GLOOPT, Global Options

```
MODULE gloopt
USE kindef
IMPLICIT NONE
SAVE

INTEGER (INT4) ::  NETTYP,   NTALG,  IFMULT,  NELPMX,  NELOVF,  NHRVER
INTEGER (INT4) ::  IFSMLS,  IFDIMS,   IFCUT,   IFDIV,   IFFOR,   IFDEV
INTEGER (INT4) ::   IFDEN,  IFSPHA,  IFSPHD,  IFVISS,  IFALLQ,  IFSMLD
INTEGER (INT4) ::   IFNOD,  IFSPHK,  NTMSTP,  NLQEXP,  IDENUP,   IFSUB, IFREADSUB
REAL (REAL8)   ::   GRAVX,   GRAVY,   GRAVZ,   QQUAD,    QLIN,   CHOUR,   CTANG
REAL (REAL8)   ::  RADCUT,  VELCUT,  SSPCUT,   FVCUT,  RHOCUT, PRESCUT,  VELLIM
REAL (REAL8)   ::   RELAX,    VTSF,   RHOMN,  SPHDLT,  VELCOR,  SSPMAX
REAL (REAL8)   ::  QQUADS,   QLINS,  QCORRS,  SMLFAC,  RHOMAX,  TEMLIM,  JOITOL

END MODULE gloopt
```

NETTYP	Type of Energy Transport (ALE/Euler)
NTALG	Type of Mass Transport algorithm (Euler)
GRAVX	X-component of gravity
GRAVY	Y-component of gravity
QQUAD	Quadratic viscosity coefficient
QLIN	Linear viscosity coefficient
CHOUR	Hourglass viscosity coefficient
CTANG	Anti-tangle constant
RADCUT	Radius cutoff (axial symmetry only)
VELCUT	Velocity minimum cutoff
SSPCUT	Soundspeed minimum cutoff
FVCUT	Covered volume fraction cutoff(rezone)
VELLIM	Velocity maximum limit
RELAX	Relaxation parameter (quasi-static damping)

Published: 2011-10-05

A.5. IJKNOW, Cell Indices

Module ijknow provides the (I,J) index and Part number for the current cell being processed. Applicable to type 3 and type 5 subroutines.

```
MODULE ijknow
USE kindef
IMPLICIT NONE

SAVE

INTEGER (INT4) :: INOW,JNOW,KNOW,LYNOW,MNOW

END MODULE ijknow
```

INOW	Current I-index
JNOW	Current J-index
KNOW	Not used in 2D
MNOW	Current Part number

A.6. JETDEF, Jetting Variables

Module jetdef includes variables associated with the jetting option. (AUTODYN-2D only)

```
MODULE jetdef
USE kindef
IMPLICIT NONE

SAVE

INTEGER, PARAMETER :: LIMJET = 100
INTEGER, PARAMETER :: MAXJVR = 21
INTEGER (INT4) :: NUMJET, NXTJET, JETSUB, JETRAP
INTEGER (INT4), DIMENSION(LIMJET), TARGET :: NPJET, JPJET
REAL (REAL8) :: VSLBAR, PMSLUG, XMOMSL
REAL (REAL8), DIMENSION(LIMJET), TARGET :: TIMJET, PMJET, XZJET
REAL (REAL8), DIMENSION(LIMJET), TARGET :: YZJET
REAL (REAL8), DIMENSION(LIMJET), TARGET :: XJET, YJET, UXJET, UYJET
REAL (REAL8), DIMENSION(LIMJET), TARGET :: DXJET, DYJET, VLJET
REAL (REAL8), DIMENSION(LIMJET), TARGET :: THKJET

END MODULE jetdef
```

LIMJET	Limit on number of jetting points
NUMJET	Number of jetting points
NXTJET	Next point to jet
JETSUB	Part containing jetting points
JETRAP	Wrapup indicator
NPJET	Array of jetted points
JPJET	Jetting point index
VSLBAR	Mean slug velocity
PMSLUG	Total slug mass
XMOMSL	Total slug momentum
TIMJET	Time of jetting
PMJET	Jet mass
XZJET	Initial X-coordinate
YZJET	Initial Y-coordinate
XJET	X coordinate at jet formation
YJET	Y coordinate at jet formation
UXJET	X component of collapse velocity
UYJET	Y component of collapse velocity
DXJET	DX of segment at jet formation
DYJET	DY of segment at jet formation
VLJET	Initial volume
THKJET	Initial thickness

Published: 2011-10-05

A.7. KINDEF, Constant Variable Definitions

Module kindef includes a number of commonly used constants (e.g. PI) that can be used in user subroutines. The variable descriptions are self-explanatory.

```
MODULE kindef
USE precision

SAVE

INTEGER, PARAMETER :: INT1  = SELECTED_INT_KIND (2)
INTEGER, PARAMETER :: INT2  = SELECTED_INT_KIND (4)
INTEGER, PARAMETER :: REAL4 = SELECTED_REAL_KIND (6,30)
INTEGER, PARAMETER :: REAL8H= SELECTED_REAL_KIND (12,300)

INTEGER (INT4), PARAMETER :: LARGE  = 999999
INTEGER (INT4), PARAMETER :: IUNDEF = 11111
INTEGER (INT4), PARAMETER :: MAXINT = 30000
INTEGER (INT4), PARAMETER :: MAXEXP = 20

REAL (REAL8), PARAMETER :: EPSLN1  = 1.0E-1_real8
REAL (REAL8), PARAMETER :: EPSLN2  = 1.0E-2_real8
REAL (REAL8), PARAMETER :: EPSLN3  = 1.0E-3_real8
REAL (REAL8), PARAMETER :: EPSLN4  = 1.0E-4_real8
REAL (REAL8), PARAMETER :: EPSLN5  = 1.0E-5_real8
REAL (REAL8), PARAMETER :: EPSLN6  = 1.0E-6_real8
REAL (REAL8), PARAMETER :: EPSLN7  = 1.0E-7_real8
REAL (REAL8), PARAMETER :: EPSLN8  = 1.0E-8_real8
REAL (REAL8), PARAMETER :: EPSLN9  = 1.0E-9_real8
REAL (REAL8), PARAMETER :: EPSLN10 = 1.0E-10_real8
REAL (REAL8), PARAMETER :: EPSLN11 = 1.0E-11_real8
REAL (REAL8), PARAMETER :: EPSLN12 = 1.0E-12_real8
REAL (REAL8), PARAMETER :: EPSLN13 = 1.0E-13_real8
REAL (REAL8), PARAMETER :: EPSLN14 = 1.0E-14_real8
REAL (REAL8), PARAMETER :: EPSLN15 = 1.0E-15_real8
REAL (REAL8), PARAMETER :: EPSLN16 = 1.0E-16_real8
REAL (REAL8), PARAMETER :: EPSLN17 = 1.0E-17_real8
REAL (REAL8), PARAMETER :: EPSLN18 = 1.0E-18_real8
REAL (REAL8), PARAMETER :: EPSLN19 = 1.0E-19_real8
REAL (REAL8), PARAMETER :: EPSLN20 = 1.0E-20_real8

REAL (REAL8), PARAMETER :: ZERO     = 0.0_real8
REAL (REAL8), PARAMETER :: ONE      = 1.0_real8
REAL (REAL8), PARAMETER :: TWO      = 2.0_real8
REAL (REAL8), PARAMETER :: THREE    = 3.0_real8
REAL (REAL8), PARAMETER :: FOUR     = 4.0_real8
REAL (REAL8), PARAMETER :: FIVE     = 5.0_real8
REAL (REAL8), PARAMETER :: SIX      = 6.0_real8
REAL (REAL8), PARAMETER :: SEVEN    = 7.0_real8
REAL (REAL8), PARAMETER :: EIGHT    = 8.0_real8
REAL (REAL8), PARAMETER :: NINE     = 9.0_real8
REAL (REAL8), PARAMETER :: TEN      = 10.0_real8
REAL (REAL8), PARAMETER :: TWELVE   = 12.0_real8
REAL (REAL8), PARAMETER :: SIXTEEN  = 16.0_real8
REAL (REAL8), PARAMETER :: NINETY   = NINE * TEN
REAL (REAL8), PARAMETER :: HUNDRED  = TEN * TEN
REAL (REAL8), PARAMETER :: ONE80    = TWO * NINE * TEN
REAL (REAL8), PARAMETER :: TWO70    = THREE * NINE * TEN
REAL (REAL8), PARAMETER :: THREE60  = TWO * ONE80
REAL (REAL8), PARAMETER :: THRHUN   = THREE * HUNDRED
REAL (REAL8), PARAMETER :: THOUSAND = HUNDRED * TEN
REAL (REAL8), PARAMETER :: TENTHOUS = HUNDRED * HUNDRED

REAL (REAL8), PARAMETER :: PI     = 3.14159265358979_real8
REAL (REAL8), PARAMETER :: SMALL  = 1.0E-20_real8
```


Published: 2011-10-05

```
REAL (REAL8), PARAMETER :: BIG    = 1.01E20_real8
REAL (REAL8), PARAMETER :: BIG2   = TWO*BIG
REAL (REAL8), PARAMETER :: UNDEF  = 1.01E11_real8
REAL (REAL8), PARAMETER :: UNDEFP = TWO*UNDEF
REAL (REAL8), PARAMETER :: CUTOFF = 1.0E-10_real8

REAL (REAL8), PARAMETER :: COMP_MAX = 3.0_real8
REAL (REAL8), PARAMETER :: K_VOID=1.E-5_real8

REAL (REAL8), PARAMETER :: HALF   = ONE/TWO
REAL (REAL8), PARAMETER :: THIRD  = ONE/THREE
REAL (REAL8), PARAMETER :: TWTHRD = TWO/THREE
REAL (REAL8), PARAMETER :: QUART  = ONE/FOUR
REAL (REAL8), PARAMETER :: FIFTH  = ONE/FIVE
REAL (REAL8), PARAMETER :: FRTHRD = FOUR/THREE
REAL (REAL8), PARAMETER :: SIXTH  = ONE/SIX
REAL (REAL8), PARAMETER :: OVER7  = ONE/SEVEN
REAL (REAL8), PARAMETER :: OVER8  = ONE/EIGHT
REAL (REAL8), PARAMETER :: OVER9  = ONE/NINE
REAL (REAL8), PARAMETER :: TENTH  = ONE/TEN
REAL (REAL8), PARAMETER :: ONEME  = ONE - SMALL
REAL (REAL8), PARAMETER :: EPSCNV = EPSPP2

END MODULE kindef
```

A.8. LOCELM, Element Quantities

Module locelm includes variables for the currently calculated cell. Applicable to user subroutines types 3 and 5.

```
MODULE locelm
USE kindef
IMPLICIT NONE
SAVE
INTEGER (INT4) ::    IJK, IMJMKM,  IJMKM,  IJKM,  IMJKM,  IMJMK
INTEGER (INT4) ::   IJMK,   IIJK,   IMJK,   IMJ,   IMJM, ISTATE
INTEGER (INT4), DIMENSION (8)   :: LELM
INTEGER (INT4), DIMENSION (4) ::IJKA
REAL (REAL8)   ::  XE1, XE2, XE3, XE4, YE1, YE2, YE3, YE4
REAL (REAL8)  ::   UX1, UX2, UX3, UX4, UY1, UY2, UY3, UY4
REAL (REAL8)  ::  YBAR,   STHETA
REAL (REAL8)  ::  CELMAS,  CDIAG, UXB, UYB
REAL (REAL8)  :: TAE1,TAE2,TAE3,TAE4, AREAE
REAL (REAL8)   ::   EDIM,   DVOV,  WXROT, WYROT,  WZROT,   QOLD
REAL (REAL8)   ::   VOLH,  URMAX,  UXBEG, UYBEG,  UZBEG
REAL (REAL8)   ::   VDOV,   DSDE,   PSAV,  ESAV
REAL (REAL8), DIMENSION (4),  TARGET ::  XEA,  YEA
REAL (REAL8), DIMENSION (8)   :: XELM, YELM, ZELM
REAL (REAL8), DIMENSION (8,3) :: AELM, BELM, CELM, UELM
REAL (REAL8), DIMENSION (4,3) :: WELM
END MODULE locelm
```

IJK	IJ index of (I,J)
IMJ	IJ index of (I-1,J)
IMJM	IJ index of (I-1,J-1)
IJKM	IJ index of (I,J-1)
XEi YEi	Coordinates of four corners of cell I,J at t(n+1) (anti-clockwise from IJ)
UXi UYi	Velocity components of four corners of cell I,J at t(n+1/2) (anti-clockwise from IJ)
YBAR	Average Y of four corners
DVOV	DV/V for cell
VDOV	(DV/DT)/V for cell
STHETA	Angle of rotation of cell
QOLD	Artificial viscosity
VOLH	Cell volume at t(n+1/2)
CELMAS	Cell mass
CDIAG	Longest diagonal of cell
EDIM	Cell dimension used to calculate timestep
UXB	X-velocity of (I,J) at start of cycle
UYB	Y-velocity of (I,J) at start of cycle
DSDE	Increment of distortional energy
TAEi	Cell area components at end of cycle
AREAE	Total cell area at end of cycle

A.9. MATDEF, Material Definitions

Module matdef includes global material data variables.

```
MODULE matdef
USE kindef

! ************************************************************

! THIS MODULE DEFINES ALL MATERIAL MODELING FLAGS/OPTIONS.

! MAIN/BASIC MATERIAL FLAGS AND OPTIONS:
! - THESE DEFINE THE TOP LEVEL STRUCTURE OF A MATERIAL MODEL
!   AND MUST ALWAYS BE SET FOR A GIVEN MATERIAL
!
! ADDITIONAL MATERIAL FLAGS/OPTIONS:
! - THESE DEFINE MATERIAL MODELING OPTIONS THAT CAN BE USED
!   TO RECURSIVELY ACCESS MATERIAL MODELING OPTION INPUT
!   AND EQUATIONS. IN PARTICULAR, THEY ARE USED IN THE GENERIC
!   MATERIAL MODEL BUILDER

! ************************************************************

IMPLICIT NONE
SAVE

INTEGER, PARAMETER :: LIMMAT = 100, LIMMAP = LIMMAT+1
INTEGER (INT4) :: IFUPDATE=0  ! FLAG TO INDICATE IF MATERIAL IS BEING UPDATED IN GET_VIS
INTEGER (INT4), PARAMETER :: LIMSOL = 8
INTEGER (INT4) :: IFSPH
INTEGER (INT4) :: NUMMAT, NUMMAP, MATNO, MATOLD,  NEOS,  NSTR,   NFAI
INTEGER (INT4) :: MUNTYP, KUNTYP, IFOUT, MATNOP
INTEGER (INT4) :: NUMMAT_TMP
LOGICAL,DIMENSION(LIMMAT) :: LSMTPL
REAL (REAL8), DIMENSION(LIMMAT) :: RMATIE, RMATKE, RMATDE, RMATVL
REAL (REAL8), DIMENSION(LIMMAT) :: RMATXM, RMATYM, RMATZM, RMATMS

! MODEL PARAMETERS COMMON TO SEVERAL FLAGS (OPTIONS)
INTEGER(INT4) :: NSBSLD, IFSTOCH
REAL (REAL8) :: RHOREF, A1, C1
REAL (REAL8) ::  TPREF, SHCV
REAL (REAL8) :: EY1, EY2, EY3, V12, V23, V31, G12, G23, G31
REAL (REAL8) :: OAN, OXC, OYC, OZC
REAL (REAL8) :: C11, C22, C33, C12, C23, C31, KEFF
REAL (REAL8) :: SHRMDZ, YLDSTZ, EROMOD, EROSON, PMIN, EPSLIM
REAL (REAL8) ::  DERIV1, DERIV2
REAL (REAL8) :: FT11, FT22, FT33, FT12, FE11, FE22, FE33, FE12, FT31, FT23
REAL (REAL8) :: FE23, FE31, X11M, Y11M, Z11M
REAL (REAL8) :: OMTY, OMAN, OMXC, OMYC, OMZC, FTYPE
REAL (REAL8) ::   GF, CCDIAG, CCOUP, CSHR
REAL (REAL8) :: CC, SS

! DEFINE PROCESSOR FLAGS - THESE SHOULD GO IN COMMON GRID MODULE WHEN INTEGRATED
INTEGER(INT4), PARAMETER  :: ISLV_LAG       = 1
INTEGER(INT4), PARAMETER  :: ISLV_EULER     = 2
INTEGER(INT4), PARAMETER  :: ISLV_ALE       = 3
INTEGER(INT4), PARAMETER  :: ISLV_SHELL     = 4
INTEGER(INT4), PARAMETER  :: ISLV_EULER_GOD = 5
INTEGER(INT4), PARAMETER  :: ISLV_FCT       = 6
INTEGER(INT4), PARAMETER  :: ISLV_SPH       = 7
INTEGER(INT4), PARAMETER  :: ISLV_BEAM      = 8

! DEFINE PARAMETERS OF MATERIAL TYPES
INTEGER(INT4), PARAMETER  :: MATTYP_ISO   = 1
INTEGER(INT4), PARAMETER  :: MATTYP_ORTHO = 2
INTEGER(INT4), PARAMETER  :: MATTYP_GAS   = 3
```


Published: 2011-10-05

```
! COMPLETE LIST OF MODEL FLAGS INCLUDING FUTURE FLAGS
INTEGER(INT4), PARAMETER  :: NUMFLAGS     = 1000
INTEGER(INT4), PARAMETER  :: NFLAGS_MAIN  = 100

INTEGER (INT4), PARAMETER::IMF_EQUATION      =1
INTEGER (INT4), PARAMETER::IMF_EOS           =2
INTEGER (INT4), PARAMETER::IMF_STR           =3
INTEGER (INT4), PARAMETER::IMF_FAI           =4
INTEGER (INT4), PARAMETER::IMF_POR           =5
INTEGER (INT4), PARAMETER::IMF_ERO           =6
INTEGER (INT4), PARAMETER::IMF_CUTOFFS       =7
INTEGER (INT4), PARAMETER::IMF_OPTIONS       =8
INTEGER (INT4), PARAMETER::IMF_USER_MAT_1    =90
INTEGER (INT4), PARAMETER::IMF_USER_MAT_2    =91
INTEGER (INT4), PARAMETER::IMF_USER_MAT_3    =92
INTEGER (INT4), PARAMETER::IMF_USER_MAT_4    =93
INTEGER (INT4), PARAMETER::IMF_USER_MAT_5    =94

INTEGER (INT4), PARAMETER::IMF_EOS_LINEAR    =101
INTEGER (INT4), PARAMETER::IMF_EOS_POLYNOMIAL=102
INTEGER (INT4), PARAMETER::IMF_EOS_IDEALGAS  =103
INTEGER (INT4), PARAMETER::IMF_EOS_SHOCK     =104
INTEGER (INT4), PARAMETER::IMF_EOS_JWL       =105
INTEGER (INT4), PARAMETER::IMF_EOS_TILLOTSON =106
INTEGER (INT4), PARAMETER::IMF_EOS_PUFF      =107
INTEGER (INT4), PARAMETER::IMF_EOS_POROUS    =108
INTEGER (INT4), PARAMETER::IMF_EOS_ORTHO     =109
INTEGER (INT4), PARAMETER::IMF_EOS_TWOPHASE  =110
INTEGER (INT4), PARAMETER::IMF_EOS_LEETARVER =111
INTEGER (INT4), PARAMETER::IMF_EOS_SESAME    =112
INTEGER (INT4), PARAMETER::IMF_EOS_COMPACTION=113
INTEGER (INT4), PARAMETER::IMF_EOS_PALPHA    =114
INTEGER (INT4), PARAMETER::IMF_EOS_GRUN      =115
INTEGER (INT4), PARAMETER::IMF_EOS_GEN       =116
INTEGER (INT4), PARAMETER::IMF_EOS_HJC       =117
INTEGER (INT4), PARAMETER::IMF_EOS_SLOWBURN  =118
INTEGER (INT4), PARAMETER::IMF_EOS_USER_1    =190
INTEGER (INT4), PARAMETER::IMF_EOS_USER_2    =191
INTEGER (INT4), PARAMETER::IMF_EOS_USER_3    =192
INTEGER (INT4), PARAMETER::IMF_EOS_USER_4    =193
INTEGER (INT4), PARAMETER::IMF_EOS_USER_5    =194
INTEGER (INT4), PARAMETER::IMF_LIMEOS        = IMF_EOS_LINEAR - 123

INTEGER (INT4), PARAMETER::IMF_STR_HYDRO     =201
INTEGER (INT4), PARAMETER::IMF_STR_ELASTIC   =202
INTEGER (INT4), PARAMETER::IMF_STR_VONMISES  =203
INTEGER (INT4), PARAMETER::IMF_STR_DRUCKERP  =204
INTEGER (INT4), PARAMETER::IMF_STR_JNCOOK    =205
INTEGER (INT4), PARAMETER::IMF_STR_ZERARM    =206
INTEGER (INT4), PARAMETER::IMF_STR_STEINB    =207
INTEGER (INT4), PARAMETER::IMF_STR_PCWISE    =208
INTEGER (INT4), PARAMETER::IMF_STR_JH2       =209
INTEGER (INT4), PARAMETER::IMF_STR_RHT       =210
INTEGER (INT4), PARAMETER::IMF_STR_GRANULAR  =211
INTEGER (INT4), PARAMETER::IMF_STR_GENERIC   =212
INTEGER (INT4), PARAMETER::IMF_STR_VISCOEL   =213
INTEGER (INT4), PARAMETER::IMF_STR_RJC       =214
INTEGER (INT4), PARAMETER::IMF_STR_HJC       =215
INTEGER (INT4), PARAMETER::IMF_STR_USER_1    =290
INTEGER (INT4), PARAMETER::IMF_STR_USER_2    =291
INTEGER (INT4), PARAMETER::IMF_STR_USER_3    =292
INTEGER (INT4), PARAMETER::IMF_STR_USER_4    =293
INTEGER (INT4), PARAMETER::IMF_STR_USER_5    =294
INTEGER (INT4), PARAMETER::IMF_STR_BEAMRESIST=295
INTEGER (INT4), PARAMETER::IMF_LIMSTR        = IMF_STR_HYDRO - 220

INTEGER (INT4), PARAMETER ::IMF_FAI_NONE     =301
INTEGER (INT4), PARAMETER ::IMF_FAI_HYDRO    =302
INTEGER (INT4), PARAMETER ::IMF_FAI_PLSTN    =303
```

```
INTEGER (INT4), PARAMETER ::IMF_FAI_PSTRESS  =304
INTEGER (INT4), PARAMETER ::IMF_FAI_PSTRAIN  =305
INTEGER (INT4), PARAMETER ::IMF_FAI_PSS      =306
INTEGER (INT4), PARAMETER ::IMF_FAI_MSTRESS  =307
INTEGER (INT4), PARAMETER ::IMF_FAI_MSTRAIN  =308
INTEGER (INT4), PARAMETER ::IMF_FAI_MSS      =309
INTEGER (INT4), PARAMETER ::IMF_FAI_CUMDAM   =310
INTEGER (INT4), PARAMETER ::IMF_FAI_JH2      =311
INTEGER (INT4), PARAMETER ::IMF_FAI_RHT      =312
INTEGER (INT4), PARAMETER ::IMF_FAI_TSHOFF   =313
INTEGER (INT4), PARAMETER ::IMF_FAI_GRADY    =314
INTEGER (INT4), PARAMETER ::IMF_FAI_JNCOOK   =315
INTEGER (INT4), PARAMETER ::IMF_FAI_USER_1   =390
INTEGER (INT4), PARAMETER ::IMF_FAI_USER_2   =391
INTEGER (INT4), PARAMETER ::IMF_FAI_USER_3   =392
INTEGER (INT4), PARAMETER ::IMF_FAI_USER_4   =393
INTEGER (INT4), PARAMETER ::IMF_FAI_USER_5   =394

INTEGER (INT4), PARAMETER ::IMF_LIMFAI       =IMF_FAI_NONE - 301

INTEGER (INT4), PARAMETER ::IMF_POR_NONE     =401
INTEGER (INT4), PARAMETER ::IMF_POR_SIMPLE   =402
INTEGER (INT4), PARAMETER ::IMF_POR_GENERIC  =403
INTEGER (INT4), PARAMETER ::IMF_POR_USER_1   =490
INTEGER (INT4), PARAMETER ::IMF_POR_USER_2   =491
INTEGER (INT4), PARAMETER ::IMF_POR_USER_3   =492
INTEGER (INT4), PARAMETER ::IMF_POR_USER_4   =493
INTEGER (INT4), PARAMETER ::IMF_POR_USER_5   =494
INTEGER (INT4), PARAMETER ::IMF_LIMPOR = IMF_POR_NONE - 404

INTEGER (INT4), PARAMETER ::IMF_ERO_NONE     =501
INTEGER (INT4), PARAMETER ::IMF_ERO_GEOMETRIC=502
INTEGER (INT4), PARAMETER ::IMF_ERO_PLASTIC  =503
INTEGER (INT4), PARAMETER ::IMF_ERO_USER_1   =590
INTEGER (INT4), PARAMETER ::IMF_ERO_USER_2   =591
INTEGER (INT4), PARAMETER ::IMF_ERO_USER_3   =592
INTEGER (INT4), PARAMETER ::IMF_ERO_USER_4   =593
INTEGER (INT4), PARAMETER ::IMF_ERO_USER_5   =594
INTEGER (INT4), PARAMETER ::IMF_LIMERO = IMF_ERO_NONE - 508

INTEGER (INT4) :: LIMPARAM  ! DEFINED IN GET_EQ_PARAM

! END OF MAIN/BASIC FLAGS

! EOS DEPENDENT FLAGS
INTEGER (INT4), PARAMETER ::IMF_TEMPERATURE          =600
INTEGER (INT4), PARAMETER ::IMF_ORTHO_MODULI         =601
INTEGER (INT4), PARAMETER ::IMF_ORTHO_STIFFMAT       =602
INTEGER (INT4), PARAMETER ::IMF_ORTHO_IJKSPACE       =603
INTEGER (INT4), PARAMETER ::IMF_ORTHO_XYZSPACE       =604

! STRENGTH DEPENDENT FLAGS
INTEGER (INT4), PARAMETER ::IMF_YP_PCWISE            =701
INTEGER (INT4), PARAMETER ::IMF_YP_LINEAR            =702
INTEGER (INT4), PARAMETER ::IMF_YP_STASSI            =703
INTEGER (INT4), PARAMETER ::IMF_YD_PCWISE            =704
INTEGER (INT4), PARAMETER ::IMF_GD_PCWISE            =705

! FAILURE DEPENDENT FLAGS
INTEGER (INT4), PARAMETER ::IMF_FAI_CRACKSOFT        =801
INTEGER (INT4), PARAMETER ::IMF_FAI_ORTHODAM         =802
INTEGER (INT4), PARAMETER ::IMF_FAI_STOCHASTIC       =803


END MODULE matdef
```

NUMMAT	Number of materials for problem
MATNO	Current material number

MATOLD	*not available*
NEOS	Current equation of state number
NSTR	Current strength model number
NFAI	Current failure model

For each cell, the following variables are defined, according to the material in that cell:

For all materials:

RHOREF	Reference density
TPREF	Reference temperature
SHCV	Specific heat (constant volume)
SHRMDZ	Shear modulus (initial)
YLDSTZ	Yield stress (initial)
EROMOD	Erosion model type
EROSON	Erosion model parameter
PMIN	Hydrodynamic tensile limit (pmin)
EPSLIM	Effective plastic strain limit

Published: 2011-10-05

A.10. MATERIAL, Local Material Data

Contains all data for each defined material. The material data is stored in the structure

```
TYPE (MAT), DIMENSION (:), POINTER            :: MATERIALS, MATERIALS_TMP
TYPE (MAT), POINTER                           :: MTL, MTL_TMP
```

The array MATERIALS is allocated when a new model is loaded into the application and is dimensioned by LIMMAT.

The data for each material is stored in a type MAT contained in MODULE material. This contains the following data:

```
TYPE MAT
        ! ************************************************************
        ! DESCRIBES A SET OF EQUATIONS/FLAGS FOR ONE MATERIAL
        !   NAME             - MATERIAL NAME
        !   REFERENCE        - A REFERENCE FOR THE MATERIAL
        !   NOTES           - ADDITIONAL NOTES ON THE MATERIAL
        !   TYP             - TYPE CLASSIFICATION OF MATERIAL (ISOTROPIC, ORTH, GAS ETC)
        !   RHOREF          - SOLID REFERENCE DENSITY FOR MATERIAL
        !   STIFFMAT        - THE MATERIAL STIFFNESS MATRIX (ISOTROPIC MATERIALS ONLY)
        !   FLAGS           - LIST OF POSSIBLE FLAGS (MATERIAL MODELLIGN OPTIONS) ASSOCIATED
                                     WITH A MATERIAL
        !   MAIN            - PROPERTIES FOR MAIN MODELING OPTIONS, USED TO ASSIST IN UI
                                     GENERATION
        !   IFSOLVER        - FLAG TO INDICATE WHICH SOLVERS A MATERIAL CAN BE USED WITH
                              THIS IS GENERATED AS A SUPERSET OF ALL THE SELECTED MATERIAL
                              MODELING OPTION FLAGS
        ! ************************************************************

        CHARACTER(LEN=30)          :: NAME
        CHARACTER(LEN=256)         :: REFERENCE, NOTES
        INTEGER(INT4)              :: TYP
        REAL(REAL8)                :: RHOREF
        REAL(REAL8), DIMENSION(3)  :: STIFFMAT
        TYPE(PARAMLIST), DIMENSION(NUMFLAGS) :: FLAGS
        TYPE(MAINFLAG), DIMENSION(NFLAGS_MAIN) :: MAIN
        INTEGER (INT4), DIMENSION(LIMSOL)               :: IFSOLVER
END TYPE MAT
```

Within each material definition, the array FLAGS contains data for all material modeling options available, NUMFLAGS. The index of all material modeling options (flags) is specified in matdef. Each option (flag) has a module associated with it which defines/provides

- input parameters
- variables
- checks
- equation solution

for that option. For example, IMF_EOS_LINEAR is the flag (index in the FLAGS array) for a linear equation of state. The input parameters , variables, checks and equation solution for the linear equation of state are contained within the module mdeos_linear.f90.

The data for each flag is stored in TYPE PARAMLIST:

```
TYPE PARAMLIST
        ! TYPE DEFINITION FOR A LIST OF PARAMETERS
        ! NAME - NAME ASSOCIETD WITH LIST
        ! IACTIVE - INDICTES IF LIST (FLAG) IS ACTIVE
        ! VISIBLE - INDICATES IF LIST IS VISIBLE IN UI
        ! IFSOLVER - INDICATES SOLVER TYPES FOR WHICH FLAG IS AVAILABLE
        ! EQTYPE - FLAG INDEX
        ! NPAR - NUMBER OF REAL PARAMETERS IN LIST
        ! NUMOPT - NUMBER OF OPTIONS IN LIST
        ! NDEPFLG - NUMBER OF DEPENDANT FLAGS THAT ARE ALWAYS USED WITH THIS FLAG (CHILDREN)
        ! IPOS - ARRAY INDICATING POSITION OF REAL PARAMETERS TO BE DISPLAYED IN UI
        ! DEPFLG - LIST OF DEPENDANT (CHILD) OPTIONS (FLAGS)
        ! PAR - REAL PARAMETER DEFINITIONS
        ! OPTION - OPTION LIST(S) DEFINITIONS

        CHARACTER (LEN=30)                          :: NAME
        INTEGER (INT4)                              :: IACTIVE
        INTEGER (INT4)                              :: VISIBLE
        INTEGER (INT4), DIMENSION(LIMSOL)           :: IFSOLVER
        INTEGER (INT4)                              :: EQTYPE  !(FLAG)
        INTEGER (INT4)                              :: NPAR, NUMOPT, NDEPFLG
        INTEGER (INT4), DIMENSION(:), POINTER       :: IPOS
        INTEGER (INT4), DIMENSION(:), POINTER       :: DEPFLG
        TYPE (PRMT), DIMENSION(:), POINTER          :: PAR
        TYPE (OPTION_LIST), DIMENSION(:), POINTER   :: OPTION
 END TYPE PARAMLIST
```

Within a PARAMLIST, the real parameters and options are defined through the types given below:

```
TYPE PRMT
        ! TYPE DEFINITION FOR A SINGLE MATERIAL INPUT PARAMETER
        ! NAME - NAME OF PARAMETER AS DISPLAYED IN UI
        ! D_L - POWER OF LENGTH UNIT
        ! D_T - POWER OF TIME UNIT
        ! D_M - POWER OF MASS UNIT
        ! D_H - POWER OF TEMEPRATURE UNIT
        ! VAL - CURRENT VALUE
        ! MIN - MINIMUM ALLOWABLE VALUE
        ! MAX - MAXIMUM ALLOWABLE VALUE
        ! DEFAULT - DEFAULT VALUE
        ! VISIBLE - VISIBILITY OF PARAMETER SWITCH
        ! REQUIRED - REQUIRED PARAMETER SWITCH

        CHARACTER (LEN=50)   :: NAME
        INTEGER (INT4)       :: D_L
        INTEGER (INT4)       :: D_T
        INTEGER (INT4)       :: D_M
        INTEGER (INT4)       :: D_H
        REAL (REAL8)         :: VAL
        REAL (REAL8)         :: MIN
        REAL (REAL8)         :: MAX
        REAL (REAL8)         :: DEFAULT
        INTEGER (INT4)       :: VISIBLE
        INTEGER (INT4)       :: REQUIRED
END TYPE PRMT

 TYPE OPTION
        ! TYPE DEFINITION FOR A SINGLE MATERIAL INPUT OPTION
        !   NAME - OPTIONS NAME
        !   AUTH - AUTHORIZATION CODE
        !   REF  - NAME  OF THE REFERENCE FILE ABOUT THIS OPTION
        !   ID   - INTEGER ID (USED FOR DIFFERENT PURPOSES, FOR EXAMPLE,
        !          IT CAN BE THE ID NUMBER FOR A DEPENDANT (CHILD) FLAG
        CHARACTER(LEN=80) :: NAME
        CHARACTER(LEN=10) :: AUTH
        CHARACTER(LEN=10) :: REF
        INTEGER(INT4) :: ID
```

```
END TYPE OPTION

TYPE OPTION_LIST
        ! TYPE DEFINITION FOR AN OPTION LIST
        ! NAME - OPTION LIST NAME
        ! NUMOPT - NUMBER OF OPTIONS IN THE LIST
        ! OPTS - DETAILS OF EACH OPTION
        ! DEFAULT - DEFAULT OPTION IN THE LIST
        ! SELETCED - CURRENT SELECTED OPTION
        ! IPOS - POSITION OF OPTION LIST WITHIN PARAMLIST
        ! REQUIRED - INDICATES WHETHER AN OPTION MUST BE SPECIFIED OR NOT
        CHARACTER (LEN=30)                          :: NAME
        INTEGER (INT4)                              :: NUMOPT
        TYPE (OPTION), DIMENSION(:), POINTER        :: OPTS
        INTEGER (INT4)                              :: DEFAULT, SELECTED
        INTEGER (INT4)                              :: IPOS, VISIBLE, REQUIRED
END TYPE
```

Local pointers used extensively throughout the code to create temporary shortcuts to the material data. The most common are

```
MTL => MATERIALS(MATNO)
POINTER TO CURRENT MATERIAL
EQ => MTL%FLAGS(IMF_****)
POINTER TO PARAMETER LIST (FLAG) *** OF MATERIAL MATNO
```

Both these pointers are referenced in module material.

Subroutine GETMAT sets up the pointer MTL to the current material MATNO. The name of a material would subsequently be available as MTL%NAME.

A.11. Equation of State (EOS) Variables

A.11.1. Linear EOS:

To access local data

```
USE matdef
```

A1	A1 parameter in linear EOS (bulk modulus)

A.11.2. Polynomial EOS :

To access local data

```
USE matdef
USE eos_polynomial
```

A1, A2, A3	A_i parameters in polynomial EOS
B0, B1	B_i parameters
T1, T2	T_i parameters

A.11.3. Ideal Gas EOS:

To access local data

```
USE eos_idealgas
```

GAMMA	Ideal gas constant, gamma
GMCON	Adiabatic constant
PSHIFT	Pressure shift

A.11.4. Shock EOS:

To access local data

```
USE matdef
USE eos_shock
```

C1	C1 parameter in shock EOS
S1	S1 parameter
GRUG	Gruneisen gamma
VE	VE relative volume
VB	VB relative volume
C2	C2 parameter
S2	S2 parameter

A.11.5. JWL EOS:

To access local data

```
USE eos_jwl
```

DETE	Chapman-Jouget(C-J) energy / unit volume in JWL EOS
A	A parameter
B	B parameter
RR1	R1 parameter
RR2	R2 parameter
W	W parameter
DETV	C-J detonation velocity
CJP	C-J pressure
BCJ	Burn on compression fraction
PREBK	Pre-burn bulk modulus
ADCON	Adiabatic constant

Published: 2011-10-05

A.11.6. Tillotson EOS:

To access local data

```
USE eos_tillotson
```

AU	Parameter A in Tillotson EOS
BU	Parameter B
AL	Parameter a
BL	Parameter b
ALP	Parameter alpha
BETA	Parameter beta
EZERO	Parameter e0
ES	Parameter es
ESD	Parameter esd

A.11.7. PUFF EOS:

To access local data

```
USE eos_puff
```

PA1, PA2, PA3	Parameters A_i in Puff EOS
GRU	Gruneisen coefficient
EXC	Expansion coefficient
SUB	Sublimation energy
PT1	Parameter T1
PT2	Parameter T2

A.11.8. Porous EOS:

To access local data

```
USE matdef
USE eos_porous
```

C1	Solid sound speed
CPOR	Porous sound speed
RTBL(1) to RTBL(10)	Tabular density values
PTBL(1) to PTBL(10)	Tabular pressure values

A.11.9. Orthotropic EOS:

To access local data

```
USE matdef
```

EY1	Youngs modulus 1
EY2	Youngs modulus 2
EY3	Youngs modulus 3
V12	Poissons ratio 12
V23	Poissons ratio 23
V31	Poissons ratio 31
OTY	Material axes option
OAN	Rotation angle
OXC	X-origin
OYC	Y-origin

Published: 2011-10-05

A.11.10. Two-Phase EOS:

To access local data

```
USE matdef
USE eos_twophase
```

CMPEOS	Compression EOS switch (linear,polynomial,shock,user)
XNUMP	Number of points in table
XNT	Pointer to model data

A.11.11. Lee-Tarver EOS:

To access local data

```
USE matdef
USE eos_leetarver
```

EZIG	Chapman-Jouget(C-J) energy / unit volume in JWL EOS
A	A parameter
B	B parameter
RR1	R1 parameter
RR2	R2 parameter
W	W parameter
DETV	C-J detonation velocity
PCJ	C-J pressure
WREAC	Reaction zone width
DFMAX	Maximum change in reaction ratio
RRI	Ignition parameter I
RRB	Ignition reaction ratio exponent
RRA	Ignition critical compression
RRX	Ignition compression exponent
RRG1	Growth parameter G1
RRC	Growth reaction ratio exponent c
RRD	Growth reaction ratio exponent d
RRY	Growth pressure exponent y
RRG2	Growth parameter G2
RRE	Growth reaction ratio exponent e

Published: 2011-10-05

RRG	Growth reaction ratio exponent g
RRZ	Growth pressure exponent z
FIGMAX	Maximum reaction ratio: ignition
FG1MAX	Maximum reaction ratio: growth G1
FG2MIN	Minimum reaction ratio: growth G2
LTUEOS	Unreacted Lee-Tarver EOS switch (shock, JWL)
VUMAX	Maximum relative volume in tension
SHKC0	Parameter C1, unreacted shock EOS
SHKS	Parameter S1, unreacted shock EOS
SHKGAM	Gruneisen coefficient, unreacted shock EOS
AUR	Unreacted JWL, coefficient A
BUR	Unreacted JWL, coefficient B
R1U	Unreacted JWL, coefficient R1
R2U	Unreacted JWL, coefficient R2
WU	Unreacted JWL, coefficient W
EZIU	Unreacted JWL, internal energy / unit volume
VVNS	Unreacted JWL, Von Neumann spike volume

A.11.12. P-α EOS:

To access local data

```
USE matdef
USE eos_palpha
```

PARHO0	Porous density
PAEL	Initial compaction pressure
PACP	Solid compaction pressure
PAC0	Porous soundspeed
PAEN	Compaction exponent
PAEOS	Switch for solid EOS (linear, polynomial, shock,
A1	Solid EOS: A1 parameter for linear, polynomial (bulk modulus)
A2, A3, B0, B1, T1, T2	Solid EOS: Parameters for polynomial EOS
C1, S1, GRUG, VE, VB, C2, S2	Solid EOS: Parameters for shock EOS

A.11.13. Rigid EOS:

To access local data

```
USE matdef
USE eos_rigid
```

C_OF_MASS_X	Initial X position of centre of mass
C_OF_MASS_Y	Initial Y position of centre of mass
C_OF_MASS_Z	Initial Z position of centre of mass
RIGID_MASS	Mass of rigid body
RIGID_IXX	Initial moment of inertia about XX
RIGID_IYY	Initial moment of inertia about YY
RIGID_IZZ	Initial moment of inertia about ZZ
RIGID_IXY	Initial moment of inertia about XY
RIGID_IYZ	Initial moment of inertia about YZ
RIGID_IZX	Initial moment of inertia about ZX
IF_RIGID_CONSTRAINT	Rigid body constraint switch
UX_RB	Constrained X-velocity
UY_RB	Constrained Y-velocity
UZ_RB	Constrained Z-velocity
URX_RB	Constrained URX-velocity
URY_RB	Constrained URY-velocity
URZ_RB	Constrained URZ-velocity

A.12. Strength Model Variables

A.12.1. Drucker-Prager Strength Model:

To access local data

```
USE matdef
USE yp_linear
USE yp_pcwise
USE yp_stassi
```

Piecewise Linear	
PRETAB(n)	Tabular pressure values, 1 to 10
YLDTAB(n)	Tabular yield stress values, 1 to 10
Linear Hardening	
SLOPEZ	Hardening slope
Stassi Hardening	
YLDSTC	Yield in uniaxial strain - compressive
YLDSTT	Yield in uniaxial strain - tension
KFACT	Ratio YLDSTC/YLDSTT

A.12.2. Johnson-Cook Strength Model:

To access local data

```
USE matdef
USE str_jncook
```

CJHCON	Hardening constant
CJHEXP	Hardening exponent
CJRATE	Strain rate constant
CJSOFT	Thermal softening exponent
CJMELT	Melting temperature

A.12.3. Zerilli-Armstrong Strength Model:

To access local data

```
USE matdef
USE str_zerarm
```

ZAHCNi	Hardening constants, I = 1, 6

A.12.4. Steinberg-Guinan Strength Model:

To access local data

```
USE matdef
USE str_steinb
```

SGYMAX	Maximum yield stress
SGHCON	Hardening constant
SGHEXP	Hardening exponent
SGHGDP	Derivative, dG/dP
SGHGDT	Derivative, dG/dT
SGHYDP	Derivative, dY/dP

A.12.5. Cowper Symonds Strength Model:

To access local data

```
USE matdef
USE str_cowper
```

YIELD0	Initial yield stress
CS_BCONST	Strain hardening constant
CS_NCONST	Strain hardening exponent
CS_DCONST	Strain rate hardening constant
CS_QCONST	Strain rate hardening exponent

Published: 2011-10-05

A.12.6. Piecewise Linear Strength Model:

To access local data

```
USE matdef
USE str_pcwise
```

EPSTAB(n)	Effective plastic strain tabular values (1-4)
YLDTAB(n)	Yield stress tabular values (1-4)
CJRATE	Strain rate constant
CJSOFT	Thermal softening exponent
CJMELT	Melting temperature

A.12.7. Johnson-Holmquist Strength Model:

To access local data

```
USE matdef
USE str_jh2
```

CJHHEL	Hugoniot Elastic Limit
CJHISA	Intact strength constant A
CJHISN	Intact strength exponent N
CJHRAT	Strain rate constant C
CJHFSB	Fracture strength constant B
CJHFSM	Fracture strength exponent M
CJHSMX	Maximum fracture strength ratio

A.12.8. RHT Concrete strength model

To access local data

```
USE matdef
USE str_rht
```

SFC	Compressive Strength (fc)
STOVERC	Tensile Strength (ft/fc)
SSOVERC	Shear Strength (fs/fc)

SBFAIL	Intact Failure Surface Constant A
SNFAIL	Intact Failure Surface Exponent N
SQ2N	Tens./Comp. Meridian Ratio (Q)
SBQ	Brittle to Ductile Transition
SPREFACT	G (elas)/(elas-plas)
STENSRAT	Elastic Strength / ft
SCOMPRAT	Elastic Strength / fc
SBFRIC	Fracture Strength Constant B
SNFRIC	Fracture Strength Exponent M
SRALPHA	Compressive Strain rate Exp. Alpha
SRDELTA	Tensile Strain rate Exp. Delta
SFMAXX	Max. Fracture Strength Ratio

A.12.9. Orthotropic Yield Strength Model

To access local data

```
USE matdef
USE str_orthyld
```

OYA11	Yield surface constant a11
OYA22	Yield surface constant a22
OYA33	Yield surface constant a33
OYA12	Yield surface constant a12
OYA23	Yield surface constant a23
OYA13	Yield surface constant a13
OYA44	Yield surface constant a44
OYA55	Yield surface constant a55
OYA66	Yield surface constant a66
EFFTAB	Table of maximum stress hardening

A.13. Crushable Foam (iso)

To access local data

```
USE matdef
USE str_isocrush

      INTEGER(INT4) :: IF_LOAD_FROM_FILE, NUM_POINTS, IF_LOADED=0
      REAL(REAL8) :: TTMAX, YIELD_CRUSH
      REAL(REAL8), DIMENSION(:), POINTER :: R_LNVOL, R_STRESS
```

IF_LOAD_FROM_FILE	Flag to get compaction data from file
NUM_POINTS	Number of compaction data points
IF_LOADED	Flag to indicate if compaction data is in memory
YIELD_CRUSH	Current crush strength
TTMAX	Tension cutoff stress
R_LNVOL	Compaction curve volumetric strain
R_STRESS	Compaction curve stress

Published: 2011-10-05

A.14. Failure Model Variables

A.14.1. Hydro (PMIN) Failure Model

To access local data

```
USE matdef
USE fai_hydro
```

PMIN	Hydro Tensile limit
REHEAL	Flag to indicate if reheal is on
GF	Crack softening G_f

A.14.2. Directional Failure Models:

To access local data

```
USE matdef
USE fai_orthodam
```

FT_{ij}	Failure stress components
FE_{ij}	Failure strain components
OMTY	Material axis option switch
OMAN	Rotation angle
OMXC	X-origin
OMYX	Y-origin
GF	Crack softening G_f

A.14.3. Cumulative Failure Models:

To access local data

```
USE matdef
USE fai_cumdam
```

EPSZDA	
EPSMDA	
DAMMAX	

Published: 2011-10-05

A.14.4. Johnson-Holmquist Damage Model:

To access local data

```
USE matdef
USE fai_jh2
```

CJHD1	Damage constant D1
CJHD2	Damage exponent D2
CJHBET	Bulking constant BETA
CJHFAI	Failure Type

A.14.5. RHT Damage Model:

To access local data

```
USE matdef
USE str_rht
```

SDAMI	Damage constant 1
SDAMII	Damage constant 2
SEFMIN	Minimum Strain to Failure
SHRATD	Residual Shear Modulus Fraction
RHTFAI	Failure Type

A.14.6. Orthotriopic Softening Model:

To access local data

```
USE matdef
USE fai_orthodam
```

GF11	Fracture energy 11
GF22	Fracture energy 22
GF33	Fracture energy 33
GF23	Fracture energy 23
GF31	Fracture energy 31
GF12	Fracture energy 12

Published: 2011-10-05

A.15. MDGRID, AUTODYN-2D Grid Variable Definitions

Module mdgrid contains the definitions for the grid variables. The various pointers and arrays to be used in accessing the grid variables are provided. For a description of the available grid variables see Appendix B.

```
MODULE mdgrid

USE cycvar
USE gloopt
USE subdef
USE material
USE memory
USE microz
USE mltmat
USE prodef
USE kindef

IMPLICIT NONE

SAVE

CHARACTER(LEN=1), DIMENSION(LIMSUB,LIMVAR), TARGET :: IDRVAR

CHARACTER(LEN=1), DIMENSION(LIMSUB,LIMIVR), TARGET :: IDIVAR

CHARACTER(LEN=1), DIMENSION(LIMSUB), TARGET :: IZONE_TMP

TYPE MULTI_MATERIAL_POINTERS
  TYPE (MULTI_ARRAY), DIMENSION(:), POINTER :: MTS
END TYPE MULTI_MATERIAL_POINTERS
TYPE (MULTI_MATERIAL_POINTERS), DIMENSION(LIMSUB) :: MTGRID
TYPE (MULTI_ARRAY), DIMENSION(:), POINTER :: MTSUB

TYPE (MICROZONE_POINTERS), DIMENSION(LIMSUB) :: MCGRID

TYPE (MICRO_ARRAY), POINTER :: MCCEL

! POINTER ARRAY TO ALL STANDARD AND TEMPORARY SUBGRID SMALL INTEGER
! VARIABLES
TYPE(SMALL_INTEGER_ARRAY_POINTER),DIMENSION(LIMSUB,LIMIVR+3)::NPACK
INTEGER, PARAMETER :: KMT =  1, KMN =  2, KMS =  3, KBX =  4, KBY =  5
INTEGER, PARAMETER :: KBI =  6, KBJ =  7, KAL =  8, KJN =  9, KOV = 10
INTEGER, PARAMETER :: KIF = 11, KNW = 12, KIC = 13, K01 = 14, K02 = 15
INTEGER, PARAMETER :: K03 = 16, K04 = 17, K05 = 18, K06 = 19, K07 = 20
INTEGER, PARAMETER :: K08 = 21, K09 = 22, K10 = 23
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKMT, NPKMN, NPKMS, NPKBX
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKBY, NPKBI, NPKBJ, NPKAL
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKJN, NPKOV, NPKIF, NPKNW
INTEGER (INT1), DIMENSION(:), POINTER ::  NPK01, NPK02, NPK03, NPK04
INTEGER (INT1), DIMENSION(:), POINTER ::  NPK05, NPK06, NPK07, NPK08
INTEGER (INT1), DIMENSION(:), POINTER ::  NPK09, NPK10, NPKIC

INTEGER (INT1), DIMENSION(:), POINTER ::  NVAR

! POINTER ARRAY TO ALL TEMPORARY SUBGRID INTEGER VARIABLES
INTEGER, PARAMETER :: LIMILT  = 10
TYPE (INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB,LIMILT) :: IGRID
INTEGER, PARAMETER :: NITMP01 = 1
INTEGER, PARAMETER :: NITMP02 = 2
INTEGER, PARAMETER :: NITMP03 = 3
INTEGER, PARAMETER :: NITMP04 = 4
INTEGER, PARAMETER :: NITMP05 = 5
INTEGER, PARAMETER :: NITMP06 = 6
INTEGER, PARAMETER :: NITMP07 = 7
```

```
INTEGER, PARAMETER :: NITMP08 = 8
INTEGER, PARAMETER :: NITMP09 = 9
INTEGER, PARAMETER :: NITMP10 = 10
INTEGER (INT4), DIMENSION(:), POINTER :: ITMP01,ITMP02,ITMP03
INTEGER (INT4), DIMENSION(:), POINTER :: ITMP04,ITMP05,ITMP06
INTEGER (INT4), DIMENSION(:), POINTER :: ITMP07,ITMP08,ITMP09
INTEGER (INT4), DIMENSION(:), POINTER :: ITMP10,ITMP11,ITMP12

! POINTER ARRAY TO ALL TEMPORARY SUBGRID COORDINATES FOR SAVE/RESTORE FACILITY
TYPE (REAL_ARRAY_POINTER),  DIMENSION(LIMSUB,3) :: GRID_TMP
TYPE (SMALL_INTEGER_ARRAY_POINTER),DIMENSION(LIMSUB,2)::NPACK_TMP

INTEGER (INT1), DIMENSION(:), POINTER ::  NPKMT_TMP, NPKMN_TMP
REAL (REAL8), DIMENSION(:), POINTER ::    XPP_TMP,    YPP_TMP

! POINTER ARRAY TO ALL STANDARD AND TEMPORARY SUBGRID REAL VARIABLES
TYPE (REAL_ARRAY_POINTER), DIMENSION(LIMSUB,LIMVAR) :: GRID

INTEGER, PARAMETER ::     NXN =  1,     NYN =  2,    NUXN =  3
INTEGER, PARAMETER ::    NUYN =  4,     NFX =  5,     NFY =  6
INTEGER, PARAMETER ::  NPMASS =  7, NRINTER =  8,   NVOLN =  9
INTEGER, PARAMETER ::  NCMASS = 10,    NXMU = 11,     NEN = 12
INTEGER, PARAMETER ::     NPN = 13,      NQ = 14,   NPLWK = 15
INTEGER, PARAMETER ::    NDEN = 16,   NTEMP = 17,    NEPS = 18
INTEGER, PARAMETER :: NEPSDOT = 19,    NEFS = 20,   NSSPD = 21
INTEGER, PARAMETER ::    NDAM = 22,    NDIV = 23,  NALPHA = 24
INTEGER, PARAMETER ::    NT11 = 25,    NT22 = 26,    NT12 = 27
INTEGER, PARAMETER ::    NTXX = 28,    NTYY = 29,    NTXY = 30
INTEGER, PARAMETER ::    NTTT = 31,    NTVM = 32,  NYIELD = 33
INTEGER, PARAMETER ::   NEXXD = 34,   NEYYD = 35,   NEXYD = 36
INTEGER, PARAMETER ::  NSTN11 = 37,  NSTN22 = 38,  NSTN33 = 39
INTEGER, PARAMETER ::  NSTN12 = 40,  NPSANG = 41,   NVOID = 42
INTEGER, PARAMETER :: NFCOVRV = 43, NFCOVRI = 44, NFCOVRJ = 45
INTEGER, PARAMETER ::   NDPDX = 46,   NDPDY = 47,   NDRDX = 48
INTEGER, PARAMETER ::   NDRDY = 49,  NDUXDX = 50,  NDUXDY = 51
INTEGER, PARAMETER ::  NDUYDX = 52,  NDUYDY = 53,   NSTN1 = 54
INTEGER, PARAMETER ::   NSTN2 = 55,   NSTR1 = 56,   NSTR2 = 57
INTEGER, PARAMETER ::  NSRES1 = 58,  NSRES2 = 59,  NBMOM1 = 60
INTEGER, PARAMETER ::  NBMOM2 = 61, NTSHEAR = 62,  NTHICK = 63
INTEGER, PARAMETER ::    NSML = 64,    NVOR = 65, NDENNM1 = 66
INTEGER, PARAMETER ::   NRCUT = 67,   NRNON = 68, NABSVEL = 69
INTEGER, PARAMETER ::  NHNORM = 70,  NTFAIL = 71,   NPDIL = 72
INTEGER, PARAMETER ::   NEDIL = 73,  NVAR01 = 74,  NVAR02 = 75
INTEGER, PARAMETER ::  NVAR03 = 76,  NVAR04 = 77,  NVAR05 = 78
INTEGER, PARAMETER ::  NVAR06 = 79,  NVAR07 = 80,  NVAR08 = 81
INTEGER, PARAMETER ::  NVAR09 = 82,  NVAR10 = 83,  NVAR11 = 84
INTEGER, PARAMETER ::  NVAR12 = 85,  NVAR13 = 86,  NVAR14 = 87
INTEGER, PARAMETER ::  NVAR15 = 88,  NVAR16 = 89,  NVAR17 = 90
INTEGER, PARAMETER ::  NVAR18 = 91,  NVAR19 = 92,  NVAR20 = 93
INTEGER, PARAMETER :: NEPSPRE = 94,  NFRATE = 95, NRTHIRD = 96
INTEGER, PARAMETER ::   NFCAP = 97,  NEPSDO = 98, NPCOR11 = 99
INTEGER, PARAMETER :: NPCOR22 =100, NPCOR33 =101, NMASFAC =102
INTEGER, PARAMETER ::   NVTXX =103,   NVTYY =104,   NVTXY =105
INTEGER, PARAMETER :: NIGTIME =106, NSBRCRT =107,    NXN0 =108
INTEGER, PARAMETER ::    NYN0 =109,   NPGAS =110, NFILDEN =111
INTEGER, PARAMETER ::   NMOTT =112

INTEGER, PARAMETER :: NTEMP01 =113, NTEMP02 = NTEMP01+1, NTEMP03 = NTEMP01+2
INTEGER, PARAMETER :: NTEMP04 = NTEMP01+3, NTEMP05 = NTEMP01+4, NTEMP06 = NTEMP01+5
INTEGER, PARAMETER :: NTEMP07 = NTEMP01+6, NTEMP08 = NTEMP01+7, NTEMP09 = NTEMP01+8
INTEGER, PARAMETER :: NTEMP10 = NTEMP01+9, NTEMP11 = NTEMP01+10, NTEMP12 = NTEMP01+11
INTEGER, PARAMETER :: NTEMP13 = NTEMP01+12, NTEMP14 = NTEMP01+13, NTEMP15 = NTEMP01+14
INTEGER, PARAMETER :: NTEMP16 = NTEMP01+15, NTEMP17 = NTEMP01+16, NTEMP18 = NTEMP01+17
INTEGER, PARAMETER :: NTEMP19 = NTEMP01+18, NTEMP20= NTEMP01+19, NTEMP21 = NTEMP01+20
INTEGER, PARAMETER :: NTEMP22 = NTEMP01+21, NTEMP23= NTEMP01+22, NTEMP24 = NTEMP01+23
INTEGER, PARAMETER :: NTEMP25 = NTEMP01+24, NTEMP26= NTEMP01+25, NTEMP27 = NTEMP01+26
INTEGER, PARAMETER :: NTEMP28 = NTEMP01+27, NTEMP29= NTEMP01+28, NTEMP30 = NTEMP01+29
REAL (REAL8), DIMENSION(:), POINTER ::     XN,     YN,    UXN,    UYN
REAL (REAL8), DIMENSION(:), POINTER ::     FX,     FY,  PMASS, RINTER
REAL (REAL8), DIMENSION(:), POINTER ::   VOLN,  CMASS,    XMU,     EN
```

```
REAL (REAL8), DIMENSION(:), POINTER ::     PN,      Q,   PLWK,    DEN
REAL (REAL8), DIMENSION(:), POINTER ::   TEMP,    EPS, EPSDOT,    EFS
REAL (REAL8), DIMENSION(:), POINTER ::   SSPD,    DAM,    DIV,  ALPHA
REAL (REAL8), DIMENSION(:), POINTER ::    T11,    T22,    T12,    TXX
REAL (REAL8), DIMENSION(:), POINTER ::    TYY,    TXY,    TTT,    TVM
REAL (REAL8), DIMENSION(:), POINTER ::  YIELD,   EXXD,   EYYD,   EXYD
REAL (REAL8), DIMENSION(:), POINTER ::  STN11,  STN22,  STN33,  STN12
REAL (REAL8), DIMENSION(:), POINTER ::  PSANG,   VOID, FCOVRV, FCOVRI
REAL (REAL8), DIMENSION(:), POINTER :: FCOVRJ,   DPDX,   DPDY,   DRDX
REAL (REAL8), DIMENSION(:), POINTER ::   DRDY,  DUXDX,  DUXDY,  DUYDX
REAL (REAL8), DIMENSION(:), POINTER ::  DUYDY,   STN1,   STN2,   STR1
REAL (REAL8), DIMENSION(:), POINTER ::   STR2,  SRES1,  SRES2,  BMOM1
REAL (REAL8), DIMENSION(:), POINTER ::  BMOM2, TSHEAR,  THICK,    SML
REAL (REAL8), DIMENSION(:), POINTER ::    VOR, DENNM1,   RCUT,   RNON
REAL (REAL8), DIMENSION(:), POINTER :: ABSVEL,  HNORM,  TFAIL,   PDIL
REAL (REAL8), DIMENSION(:), POINTER ::   EDIL,  VAR01,  VAR02,  VAR03
REAL (REAL8), DIMENSION(:), POINTER ::  VAR04,  VAR05,  VAR06,  VAR07
REAL (REAL8), DIMENSION(:), POINTER ::  VAR08,  VAR09,  VAR10,  VAR11
REAL (REAL8), DIMENSION(:), POINTER ::  VAR12,  VAR13,  VAR14,  VAR15
REAL (REAL8), DIMENSION(:), POINTER ::  VAR16,  VAR17,  VAR18,  VAR19
REAL (REAL8), DIMENSION(:), POINTER ::  VAR20, EPSPRE,  FRATE, RTHIRD
REAL (REAL8), DIMENSION(:), POINTER ::   FCAP,  EPSDO, PCOR11, PCOR22
REAL (REAL8), DIMENSION(:), POINTER :: PCOR33, MASFAC,   VTXX,   VTYY
REAL (REAL8), DIMENSION(:), POINTER ::   VTXY, IGTIME, SBRCRT,    XN0
REAL (REAL8), DIMENSION(:), POINTER ::    YN0,   PGAS, FILDEN, MOTT
REAL (REAL8), DIMENSION(:), POINTER :: TEMP01, TEMP02, TEMP03
REAL (REAL8), DIMENSION(:), POINTER :: TEMP04, TEMP05, TEMP06, TEMP07
REAL (REAL8), DIMENSION(:), POINTER :: TEMP08, TEMP09, TEMP10, TEMP11
REAL (REAL8), DIMENSION(:), POINTER :: TEMP12, TEMP13, TEMP14, TEMP15
REAL (REAL8), DIMENSION(:), POINTER :: TEMP16, TEMP17, TEMP18, TEMP19
REAL (REAL8), DIMENSION(:), POINTER :: TEMP20, TEMP21, TEMP22, TEMP23
REAL (REAL8), DIMENSION(:), POINTER :: TEMP24, TEMP25, TEMP26, TEMP27
REAL (REAL8), DIMENSION(:), POINTER :: TEMP28, TEMP29, TEMP30
```

A.16. MDGRID3, AUTODYN-3D Grid Variable Definitions

Module mdgrid3 contains the definitions for the grid variables. The various pointers and arrays to be used in accessing the grid variables are provided. For a description of the available grid variables see Appendix B.

```
MODULE mdgrid3

USE kindef
USE memory
USE cycvar
USE gloopt
USE ijknow
USE subdef
USE mltmat3
USE microz3
USE prodef3
USE euldef
USE eulmem
USE verdef
USE fctshl


IMPLICIT NONE

SAVE

CHARACTER(LEN=1), DIMENSION(LIMSUB,LIMVAR), TARGET :: IDRVAR

CHARACTER(LEN=1), DIMENSION(LIMSUB,LIMIVR), TARGET :: IDIVAR

CHARACTER(LEN=1), DIMENSION(LIMSUB), TARGET :: IZONE_TMP

INTEGER (INT1), DIMENSION(1),     TARGET :: IAPSNL=99
INTEGER (INT1), DIMENSION(1,1,1), TARGET :: IAP3NL=99
INTEGER (INT4), DIMENSION(1),     TARGET :: IAPNUL=999999
INTEGER (INT4), DIMENSION(1,1),   TARGET :: IAP2NL=999999
REAL (REAL8),   DIMENSION(1),     TARGET :: RAPNUL=999999.0

TYPE (JOIN_POINTERS), DIMENSION(LIMSUB) :: JNGRID, JNTEMP

TYPE (INTEGER_ARRAY_POINTER), DIMENSION(:), POINTER :: JNSUB
TYPE (INTEGER_ARRAY_POINTER), DIMENSION(1),  TARGET :: JNNUL

TYPE WET_PAR
  REAL(REAL8),DIMENSION(6) :: FR
  REAL(REAL8)              :: SC
  REAL(REAL8),DIMENSION(3) :: GD
  INTEGER (INT1)           :: M1,M2
END TYPE WET_PAR

TYPE SCPT
  REAL (REAL8),POINTER :: PT
END TYPE SCPT

TYPE TRANS_VAR
  INTEGER(INT1)                        :: NMP
  INTEGER(INT4),DIMENSION(:),POINTER   :: MT
  REAL  (REAL8),DIMENSION(:),POINTER   :: VR
  REAL  (REAL8),DIMENSION(:),POINTER   :: VTR1
  REAL  (REAL8),DIMENSION(:),POINTER   :: VTR2
  REAL  (REAL8),DIMENSION(:),POINTER   :: VTR3
END  TYPE TRANS_VAR

TYPE MULTI_ARRAY_3D
  INTEGER (INT4) :: NUMLT
```


Published: 2011-10-05

```
  INTEGER (INT4) :: NUMDT
  INTEGER (INT4), DIMENSION(:), POINTER :: M
  REAL (REAL8),   DIMENSION(:), POINTER :: V
  TYPE(TRANS_VAR), POINTER :: TRNS
  TYPE(WET_PAR), POINTER :: WFRV
END TYPE MULTI_ARRAY_3D
TYPE TRMEM
  TYPE(TRANS_VAR), DIMENSION(:), POINTER :: TR
  INTEGER(INT4)  , DIMENSION(:), POINTER :: IV
  REAL (REAL8)   , DIMENSION(:), POINTER :: RV
  REAL (REAL8)   , DIMENSION(:), POINTER :: VTR1
  REAL (REAL8)   , DIMENSION(:), POINTER :: VTR2
  REAL (REAL8)   , DIMENSION(:), POINTER :: VTR3
END TYPE TRMEM
TYPE MULTI_MATERIAL_POINTERS
  TYPE (MULTI_ARRAY_3D), DIMENSION(:), POINTER :: MTS
END TYPE MULTI_MATERIAL_POINTERS

INTEGER (INT1),TARGET,DIMENSION(1) :: NULIAP
REAL (REAL8)  ,TARGET,DIMENSION(1) :: NULRAP
TYPE(TRANS_VAR), DIMENSION(:), POINTER :: TRANS
TYPE(TRANS_VAR), TARGET :: NULTRN
TYPE(WET_PAR)  , DIMENSION(:), POINTER :: WETPT

TYPE (MULTI_MATERIAL_POINTERS), DIMENSION(LIMSUB) :: MTGRID, MTTEMP

TYPE (MULTI_ARRAY_3D), DIMENSION(:), POINTER :: MTSUB
REAL (REAL8), DIMENSION(:), POINTER :: ML, MLX

TYPE (REAL_ARRAY2_POINTER), DIMENSION(LIMMLV) :: CMLT

INTERFACE MEMALLOC
  MODULE PROCEDURE MEMALLOC_MULTMAT_POINTER3
  MODULE PROCEDURE MEMALLOC_MULTI_ARRAY3_3D
  MODULE PROCEDURE MEMALLOC_TRANS_VAR_ARRAY3
  MODULE PROCEDURE MEMALLOC_MICARRAY_POINTER3
END INTERFACE

INTERFACE MEMDEALLOC
  MODULE PROCEDURE MEMDEALLOC_MULTMAT_POINTER3
  MODULE PROCEDURE MEMDEALLOC_MULTI_ARRAY3_3D
  MODULE PROCEDURE MEMDEALLOC_TRANS_VAR_ARRAY3
  MODULE PROCEDURE MEMDEALLOC_MICARRAY_POINTER3
END INTERFACE

INTEGER (INT4), DIMENSION(:), POINTER ::  MVAR

! MULTIMATERIAL VARIABLES STORED FOR EACH MAT IN ZONE:
! 1 CVF - RELATIVE VOLUME 2 CMS -  MASS       3 CEN -  ENERGY
! 4 CMU - RHO/RHOREF-1    5 CTP - TEMPERATURE 6 CAL -  REACTION RATE
! 7 CBF -                 8 CDM - DAMAGE      9 CPS - PLASTIC STRAIN
!10 CCC - C ZERO          11 CSS - S         12 CSN - PLASTIC STRAIN
! STORED IN MTGRID(NSUB)%MTS(IJK)%V(NNMZVR+(1:9)) ...

INTEGER, PARAMETER :: NCVF = 1, NCMS = 2, NCEN = 3, NCMU = 4
INTEGER, PARAMETER :: NCTP = 5, NCAL = 6, NCBF = 7, NCDM = 8
INTEGER, PARAMETER :: NCPS = 9, NCCC =10, NCSS =11, NCSN =12

! ZONAL VARIABLES DEFINED FOR NON VOID ZONES:
! NNVOLN=NOFMV(NVOLN) , ETC... NOFMV DEFINED IN INIT

INTEGER (INT4) :: NNUXN , NNUYN ,  NNUZN
INTEGER (INT4) :: NNVOLN, NNPN  ,  NNPLWK, NNDEN
INTEGER (INT4) :: NNEPSD, NNEFS ,  NNSSPD, NNDIV
INTEGER (INT4) :: NNTXX , NNTYY ,  NNTZZ , NNTXY
INTEGER (INT4) :: NNTYZ , NNTZX ,  NNTVM , NNYLD
INTEGER (INT4) :: NNEXXD, NNEYYD,  NNEZZD
INTEGER (INT4) :: NNEXYD, NNEYZD,  NNEZXD, NNVOID
INTEGER (INT4) :: NNSLP1, NNDPDX,  NNDPDY, NNDPDZ
INTEGER (INT4) :: NNDRDX, NNDRDY,  NNDRDZ
```

```
INTEGER (INT4) :: NNUXDX, NNUXDY,  NNUXDZ
INTEGER (INT4) :: NNUYDX, NNUYDY,  NNUYDZ
INTEGER (INT4) :: NNUZDX, NNUZDY,  NNUZDZ, NNSLP2
INTEGER (INT4) :: NNRIJN, NNRJJN,  NNRKJN, NNZVAR
INTEGER (INT4),DIMENSION(6) :: NNSTRS
INTEGER, PARAMETER :: NNMXVR = 50
! NNMZVR=NNZVAR FOR NPROC=5,0 FOR OTHERS...
! NNMZVR=41, NDAT1=NNMZVR+1 NO PARAMETER AS GETMLT USED BY OTHER PROC TOO
INTEGER (INT4) :: IL
INTEGER (INT4), DIMENSION(LIMVAR) :: NOFMV
INTEGER (INT4), DIMENSION(1),TARGET :: UNUSED=99
REAL (REAL8), DIMENSION(NNMXVR), TARGET :: VOIDTG
REAL (REAL8), DIMENSION(1), TARGET :: NULLTG
REAL (REAL8), TARGET :: ZEROTG=ZERO
REAL (REAL8), DIMENSION(NNMXVR,2), TARGET :: ZVBUFF
REAL (REAL8), DIMENSION(:), POINTER :: VARS, VARSP, SLOPES

REAL (REAL8), DIMENSION(2,LIMMAP), TARGET :: CVF, CMS, CEN, CMU, CTP
REAL (REAL8), DIMENSION(2,LIMMAP), TARGET :: CAL, CBF, CDM, CPS
REAL (REAL8), DIMENSION(2,LIMMAP), TARGET :: CCC, CSS, CSN
REAL (REAL8), DIMENSION(:,:), POINTER :: GVARM
CHARACTER (LEN=1), DIMENSION(LIMMLV), TARGET :: IDRESM, IDHISM, IDPRTM, IDCONM
CHARACTER (LEN=1), DIMENSION(LIMMLV), TARGET :: IDEXMM, IDREZM, IDNOCM

TYPE MICRO_ARRAY_3D
  INTEGER (INT1), DIMENSION(LIMMC,LIMMC,LIMMC) :: MAT
  REAL (REAL8), DIMENSION(LIMMC,LIMMC,LIMMC) :: UX, UY, UZ, RHO, SIE
END TYPE MICRO_ARRAY_3D

TYPE MICRO_ARRAY_3D_POINTER
  TYPE (MICRO_ARRAY_3D), POINTER :: MAR
END TYPE MICRO_ARRAY_3D_POINTER

TYPE MICROZONE_POINTERS_3D
  TYPE (MICRO_ARRAY_3D_POINTER), DIMENSION(:), POINTER :: MCR
END TYPE MICROZONE_POINTERS_3D

TYPE (MICROZONE_POINTERS_3D), DIMENSION(LIMSUB) :: MCGRID

TYPE (MICRO_ARRAY_3D), POINTER :: MCCEL

! POINTER ARRAY TO ALL STANDARD AND TEMPORARY SUBGRID SMALL INTEGER
! VARIABLES
TYPE (SMALL_INTEGER_ARRAY_POINTER),DIMENSION(LIMSUB,LIMIVR+3)::NPACK
INTEGER, PARAMETER :: KMT =  1, KMN =  2, KMS =  3, KBX =  4, KBY =  5
INTEGER, PARAMETER :: KBZ =  6, KBI =  7, KBJ =  8, KBK =  9, KRX = 10
INTEGER, PARAMETER :: KRY = 11, KRZ = 12, KNW = 13, KAL = 14, KED = 15
INTEGER, PARAMETER :: KIC = 16
INTEGER, PARAMETER :: K01 = 17
INTEGER, PARAMETER :: K02 = K01+1, K03 = K01+2, K04 = K01+3, K05 = K01+4
INTEGER, PARAMETER :: K06 = K01+5, K07 = K01+6, K08 = K01+7, K09 = K01+8
INTEGER, PARAMETER :: K10 = K01+9
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKMT, NPKMN, NPKMS, NPKBX
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKBY, NPKBZ, NPKBI, NPKBJ
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKBK, NPKRX, NPKRY, NPKRZ
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKNW, NPKAL, NPK01, NPK02
INTEGER (INT1), DIMENSION(:), POINTER ::  NPK03, NPK04, NPK05, NPK06
INTEGER (INT1), DIMENSION(:), POINTER ::  NPK07, NPK08, NPK09, NPK10
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKED, NPKIC
INTEGER (INT1), DIMENSION(:), POINTER ::  NVAR,  NTVR,  KBIJK

! POINTER ARRAY TO ALL TEMPORARY SUBGRID INTEGER VARIABLES
INTEGER, PARAMETER :: LIMILT  = 10
TYPE (INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB,LIMILT) :: IGRID

INTEGER, PARAMETER :: NITMP01 = 1
INTEGER, PARAMETER :: NITMP02 = 2
INTEGER, PARAMETER :: NITMP03 = 3
INTEGER, PARAMETER :: NITMP04 = 4
INTEGER, PARAMETER :: NITMP05 = 5
```

```
INTEGER, PARAMETER :: NITMP06 = 6
INTEGER, PARAMETER :: NITMP07 = 7
INTEGER, PARAMETER :: NITMP08 = 8
INTEGER, PARAMETER :: NITMP09 = 9
INTEGER, PARAMETER :: NITMP10 = 10
INTEGER (INT4), DIMENSION(:), POINTER :: ITMP01,ITMP02,ITMP03
INTEGER (INT4), DIMENSION(:), POINTER :: ITMP04,ITMP05,ITMP06
INTEGER (INT4), DIMENSION(:), POINTER :: ITMP07,ITMP08,ITMP09
INTEGER (INT4), DIMENSION(:), POINTER :: ITMP10,ITMP11,ITMP12

! POINTER ARRAY TO ALL STANDARD AND TEMPORARY SUBGRID REAL VARIABLES
TYPE (REAL_ARRAY_POINTER),  DIMENSION(LIMSUB,LIMVAR) :: GRID

! POINTER ARRAY TO ALL TEMPORARY SUBGRID COORDINATES FOR SAVE/RESTORE FACILITY
TYPE (REAL_ARRAY_POINTER),  DIMENSION(LIMSUB,3) :: GRID_TMP
TYPE (SMALL_INTEGER_ARRAY_POINTER),DIMENSION(LIMSUB,2)::NPACK_TMP
INTEGER (INT1), DIMENSION(:), POINTER ::  NPKMT_TMP, NPKMN_TMP
REAL (REAL8), DIMENSION(:), POINTER ::  XPP_TMP, YPP_TMP, ZPP_TMP

! POINTER ARRAY FOR TEMPORARY REAL VARIABLES IN EUL3P3
INTEGER (INT4),PARAMETER   :: NSMGVR = 19
TYPE (REAL_ARRAY_POINTER), DIMENSION(LIMSUB,NSMGVR) :: SGRID
! TEMPORARY 2D POINTER ARRAYS FOR TRANSPORTS
TYPE (REAL_ARRAY2_POINTER), DIMENSION(LIMSUB)   :: TRVOL
TYPE (REAL_ARRAY2_POINTER), DIMENSION(LIMSUB)   :: GRDTMS

! POINTER ARRAYS TO BEAM OBJECT LISTS
TYPE (CHARACTER_10_POINTER),  DIMENSION(LIMSUB) :: NAMEBM
TYPE (INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: NBASBM
TYPE (INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: IDIABB
TYPE (SMALL_INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: IBMXBB
TYPE (SMALL_INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: JBMXBB
TYPE (SMALL_INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: KBMXBB
TYPE (SMALL_INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: INCXBB
TYPE (SMALL_INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: INCYBB
TYPE (SMALL_INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: INCZBB
TYPE (SMALL_INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB) :: NBOJTB
INTEGER (INT4), DIMENSION(:), POINTER :: NBSBOJ, IDIAGB
INTEGER (INT1), DIMENSION(:), POINTER :: IBMXBM, JBMXBM, KBMXBM
INTEGER (INT1), DIMENSION(:), POINTER :: INCXBM, INCYBM, INCZBM
INTEGER (INT1), DIMENSION(:), POINTER :: NBOJTY
CHARACTER (LEN=10), DIMENSION(:), POINTER :: NAMBOJ
INTEGER (INT4) :: IMDUM

INTEGER, PARAMETER ::     NXN =  1,     NYN =  2,     NZN =  3
INTEGER, PARAMETER ::    NUXN =  4,    NUYN =  5,    NUZN =  6
INTEGER, PARAMETER ::     NFX =  7,     NFY =  8,     NFZ =  9
INTEGER, PARAMETER ::  NPMASS = 10, NBMAREA = 11,   NVOLN = 12
INTEGER, PARAMETER ::     NPN = 13,      NQ = 14,   NPLWK = 15
INTEGER, PARAMETER ::    NDEN = 16, NEPSDOT = 17,    NEFS = 18
INTEGER, PARAMETER ::   NSSPD = 19,    NDIV = 20,    NT11 = 21
INTEGER, PARAMETER ::    NT22 = 22,    NT33 = 23,    NTXX = 24
INTEGER, PARAMETER ::    NTYY = 25,    NTZZ = 26,    NTXY = 27
INTEGER, PARAMETER ::    NTYZ = 28,    NTZX = 29,    NTVM = 30
INTEGER, PARAMETER ::  NYIELD = 31,   NEXXD = 32,   NEYYD = 33
INTEGER, PARAMETER ::   NEZZD = 34,   NEXYD = 35,   NEYZD = 36
INTEGER, PARAMETER ::   NEZXD = 37,    NWXN = 38,    NWYN = 39
INTEGER, PARAMETER ::    NWZN = 40,   NRI11 = 41,   NRI22 = 42
INTEGER, PARAMETER ::   NRI33 = 43,   NVOID = 44,   NDPDX = 45
INTEGER, PARAMETER ::   NDPDY = 46,   NDPDZ = 47,   NDRDX = 48
INTEGER, PARAMETER ::   NDRDY = 49,   NDRDZ = 50,  NDUXDX = 51
INTEGER, PARAMETER ::  NDUXDY = 52,  NDUXDZ = 53,  NDUYDX = 54
INTEGER, PARAMETER ::  NDUYDY = 55,  NDUYDZ = 56,  NDUZDX = 57
INTEGER, PARAMETER ::  NDUZDY = 58,  NDUZDZ = 59,   NSTN1 = 60
INTEGER, PARAMETER ::   NSTN2 = 61,  NSTN12 = 62,  NSTRS1 = 63
INTEGER, PARAMETER ::  NSTRS2 = 64, NSTRS12 = 65,  NSRES1 = 66
INTEGER, PARAMETER ::  NSRES2 = 67,  NBMOM1 = 68,  NBMOM2 = 69
INTEGER, PARAMETER :: NBMOM12 = 70,  NTHICK = 71,  NDIRNX = 72
INTEGER, PARAMETER ::  NDIRNY = 73,  NDIRNZ = 74,    NT12 = 75
INTEGER, PARAMETER ::    NT23 = 76,    NT31 = 77,  NSTN11 = 78
```

```
INTEGER, PARAMETER ::  NSTN22 = 79,  NSTN33 = 80, NSTN12V = 81
INTEGER, PARAMETER ::  NSTN23 = 82,  NSTN31 = 83, NRIJOIN = 84
INTEGER, PARAMETER :: NRJJOIN = 85, NRKJOIN = 86,  NCOSFI = 87
INTEGER, PARAMETER ::  NSINFI = 88,  NPSANG = 89, NSPARE1 = 90
INTEGER, PARAMETER :: NBMLENZ = 91, NSPARE2 = 92,    NRJJ = 93
INTEGER, PARAMETER ::   NFAXI = 94,   NFTOR = 95, NBMOMYI = 96
INTEGER, PARAMETER :: NBMOMYJ = 97, NBMOMZI = 98, NBMOMZJ = 99
INTEGER, PARAMETER ::  NBBV11 =100,  NBBV12 =101,  NBBV13 =102
INTEGER, PARAMETER ::  NBBV21 =103,  NBBV22 =104,  NBBV23 =105
INTEGER, PARAMETER ::  NBBV31 =106,  NBBV32 =107,  NBBV33 =108
INTEGER, PARAMETER ::  NEBV11 =109,  NEBV12 =110,  NEBV13 =111
INTEGER, PARAMETER ::  NEBV21 =112,  NEBV22 =113,  NEBV23 =114
INTEGER, PARAMETER ::  NEBV31 =115,  NEBV32 =116,  NEBV33 =117

INTEGER, PARAMETER ::  NVAR01 =118,  NVAR02 =119,  NVAR03 =120
INTEGER, PARAMETER ::  NVAR04 =121,  NVAR05 =122,  NVAR06 =123
INTEGER, PARAMETER ::  NVAR07 =124,  NVAR08 =125,  NVAR09 =126
INTEGER, PARAMETER ::  NVAR10 =127,  NVAR11 =128,  NVAR12 =129
INTEGER, PARAMETER ::  NVAR13 =130,  NVAR14 =131,  NVAR15 =132
INTEGER, PARAMETER ::  NVAR16 =133,  NVAR17 =134,  NVAR18 =135
INTEGER, PARAMETER ::  NVAR19 =136,  NVAR20 =137

INTEGER, PARAMETER ::    NSML =138, NDENNM1 =139,   NRNON =140
INTEGER, PARAMETER ::   NVORX =141,   NVORY =142,   NVORZ =143
INTEGER, PARAMETER ::   NHQM1 =144,   NHQM2 =145,   NHQB1 =146
INTEGER, PARAMETER ::   NHQB2 =147,   NHQB3 =148, NFCOVRI =149
INTEGER, PARAMETER ::  NFCOVRJ=150, NFCOVRK =151, NFCOVRV =152
INTEGER, PARAMETER ::  NRBLEND=153,   NPDIL =154,   NEDIL =155
INTEGER, PARAMETER :: NEPSPRE =156,  NFRATE =157,   NFCAP =158
INTEGER, PARAMETER ::  NEPSDO =159, NRTHIRD =160,  NTFAIL =161
INTEGER, PARAMETER ::  NHNORM =162, NPCOR11 =163, NPCOR22 =164
INTEGER, PARAMETER :: NPCOR33 =165,   NVTXX =166,   NVTYY =167
INTEGER, PARAMETER ::   NVTZZ =168,   NVTXY =169,   NVTYZ =170
INTEGER, PARAMETER ::   NVTZX =171, NIGTIME =172, NSBRCRT =173
INTEGER, PARAMETER :: NABSVEL =174,  NFMASS =175, NNUMCEL =176
INTEGER, PARAMETER :: NSPDMMS =177, NDTMPDX =178, NDTMPDY =179
INTEGER, PARAMETER :: NDTMPDZ =180, NTHMENG =181,    NXN0 =182
INTEGER, PARAMETER ::    NYN0 =183,    NZN0 =184,   NPGAS =185
INTEGER, PARAMETER :: NFILDEN =186,   NMOTT =187

INTEGER, PARAMETER ::  NSTRT = 187

INTEGER, PARAMETER :: NTEMP01 =NSTRT+01, NTEMP02 =NSTRT+02
INTEGER, PARAMETER :: NTEMP03 =NSTRT+03, NTEMP04 =NSTRT+04
INTEGER, PARAMETER :: NTEMP05 =NSTRT+05, NTEMP06 =NSTRT+06
INTEGER, PARAMETER :: NTEMP07 =NSTRT+07, NTEMP08 =NSTRT+08
INTEGER, PARAMETER :: NTEMP09 =NSTRT+09, NTEMP10 =NSTRT+10
INTEGER, PARAMETER :: NTEMP11 =NSTRT+11, NTEMP12 =NSTRT+12
INTEGER, PARAMETER :: NTEMP13 =NSTRT+13, NTEMP14 =NSTRT+14
INTEGER, PARAMETER :: NTEMP15 =NSTRT+15, NTEMP16 =NSTRT+16
INTEGER, PARAMETER :: NTEMP17 =NSTRT+17, NTEMP18 =NSTRT+18
INTEGER, PARAMETER :: NTEMP19 =NSTRT+19, NTEMP20 =NSTRT+20
INTEGER, PARAMETER :: NTEMP21 =NSTRT+21, NTEMP22 =NSTRT+22
INTEGER, PARAMETER :: NTEMP23 =NSTRT+23, NTEMP24 =NSTRT+24
INTEGER, PARAMETER :: NTEMP25 =NSTRT+25, NTEMP26 =NSTRT+26
INTEGER, PARAMETER :: NTEMP27 =NSTRT+27, NTEMP28 =NSTRT+28
INTEGER, PARAMETER :: NTEMP29 =NSTRT+29, NTEMP30 =NSTRT+30
INTEGER, PARAMETER :: NTEMP31 =NSTRT+31, NTEMP32 =NSTRT+32
INTEGER, PARAMETER :: NTEMP33 =NSTRT+33

INTEGER (INT4)     :: NTTAL=0,NBLAL=0,NBLRL=0,NNALL=0,NMTPAR=0
 ! LIST OF EUL. ZONE VARIABLES DEFINED IN GRID
INTEGER (INT4)            :: NGDVAR = 0
INTEGER,DIMENSION(LIMVAR) :: LGDVAR
! LIST OF TEMP EUL. ZONE VARIABLES DEFINED FOR NON VOID ZONES
INTEGER (INT4),PARAMETER  :: NVRNVZ =14
INTEGER,DIMENSION(NVRNVZ) :: LVRNVZ= (/(IL,IL=NTEMP01,NTEMP14)/)

REAL (REAL8), DIMENSION(:), POINTER ::     XN,     YN,     ZN,    UXN
REAL (REAL8), DIMENSION(:), POINTER ::    UYN,    UZN,     FX,     FY
```

```
REAL (REAL8), DIMENSION(:), POINTER ::     FZ,  PMASS, BMAREA,   VOLN
REAL (REAL8), DIMENSION(:), POINTER ::     PN,      Q,   PLWK,    DEN
REAL (REAL8), DIMENSION(:), POINTER :: EPSDOT,    EFS,   SSPD,    DIV
REAL (REAL8), DIMENSION(:), POINTER ::    T11,    T22,    T33,    TXX
REAL (REAL8), DIMENSION(:), POINTER ::    TYY,    TZZ,    TXY,    TYZ
REAL (REAL8), DIMENSION(:), POINTER ::    TZX,    TVM,  YIELD,   EXXD
REAL (REAL8), DIMENSION(:), POINTER ::   EYYD,   EZZD,   EXYD,   EYZD
REAL (REAL8), DIMENSION(:), POINTER ::   EZXD,    WXN,    WYN,    WZN
REAL (REAL8), DIMENSION(:), POINTER ::   RI11,   RI22,   RI33,   VOID
REAL (REAL8), DIMENSION(:), POINTER ::   DPDX,   DPDY,   DPDZ,   DRDX
REAL (REAL8), DIMENSION(:), POINTER ::   DRDY,   DRDZ,  DUXDX,  DUXDY
REAL (REAL8), DIMENSION(:), POINTER ::  DUXDZ,  DUYDX,  DUYDY,  DUYDZ
REAL (REAL8), DIMENSION(:), POINTER ::  DUZDX,  DUZDY,  DUZDZ,   STN1
REAL (REAL8), DIMENSION(:), POINTER ::   STN2,  STN12,  STRS1,  STRS2
REAL (REAL8), DIMENSION(:), POINTER :: STRS12,  SRES1,  SRES2,  BMOM1
REAL (REAL8), DIMENSION(:), POINTER ::  BMOM2, BMOM12,  THICK,  DIRNX
REAL (REAL8), DIMENSION(:), POINTER ::  DIRNY,  DIRNZ,    T12,    T23
REAL (REAL8), DIMENSION(:), POINTER ::    T31,  STN11,  STN22,  STN33
REAL (REAL8), DIMENSION(:), POINTER :: STN12V,  STN23,  STN31, RIJOIN
REAL (REAL8), DIMENSION(:), POINTER :: RJJOIN, RKJOIN,  COSFI,  SINFI

REAL (REAL8), DIMENSION(:), POINTER ::  PSANG, BMLENZ
REAL (REAL8), DIMENSION(:), POINTER ::    RJJ,   FAXI,   FTOR, BMOMYI
REAL (REAL8), DIMENSION(:), POINTER :: BMOMYJ, BMOMZI, BMOMZJ,  BBV11
REAL (REAL8), DIMENSION(:), POINTER ::  BBV12,  BBV13,  BBV21,  BBV22
REAL (REAL8), DIMENSION(:), POINTER ::  BBV23,  BBV31,  BBV32,  BBV33
REAL (REAL8), DIMENSION(:), POINTER ::  EBV11,  EBV21,  EBV31,  EBV12
REAL (REAL8), DIMENSION(:), POINTER ::  EBV22,  EBV32,  EBV13,  EBV23
REAL (REAL8), DIMENSION(:), POINTER ::  EBV33
REAL (REAL8), DIMENSION(:), POINTER ::  VAR01,  VAR02,  VAR03,  VAR04
REAL (REAL8), DIMENSION(:), POINTER ::  VAR05,  VAR06,  VAR07,  VAR08
REAL (REAL8), DIMENSION(:), POINTER ::  VAR09,  VAR10,  VAR11,  VAR12
REAL (REAL8), DIMENSION(:), POINTER ::  VAR13,  VAR14,  VAR15,  VAR16
REAL (REAL8), DIMENSION(:), POINTER ::  VAR17,  VAR18,  VAR19,  VAR20
REAL (REAL8), DIMENSION(:), POINTER ::    SML, DENNM1,   RNON,   VORX
REAL (REAL8), DIMENSION(:), POINTER ::   VORY,   VORZ,   HQM1,   HQM2
REAL (REAL8), DIMENSION(:), POINTER ::   HQB1,   HQB2,   HQB3, FCOVRI
REAL (REAL8), DIMENSION(:), POINTER :: FCOVRJ, FCOVRK, FCOVRV, RBLEND
REAL (REAL8), DIMENSION(:), POINTER ::   PDIL,   EDIL, EPSPRE,  HNORM
REAL (REAL8), DIMENSION(:), POINTER :: FRATE,    FCAP,  EPSDO, RTHIRD
REAL (REAL8), DIMENSION(:), POINTER :: TFAIL,  PCOR11, PCOR22, PCOR33
REAL (REAL8), DIMENSION(:), POINTER ::  VTXX,    VTYY,   VTZZ,   VTXY
REAL (REAL8), DIMENSION(:), POINTER ::  VTYZ,    VTZX, IGTIME, SBRCRT
REAL (REAL8), DIMENSION(:), POINTER ::ABSVEL,   FMASS, NUMCEL, SPDMMS
REAL (REAL8), DIMENSION(:), POINTER ::DTMPDX,  DTMPDY, DTMPDZ, THMENG
REAL (REAL8), DIMENSION(:), POINTER ::   XN0,     YN0,    ZN0,   PGAS
REAL (REAL8), DIMENSION(:), POINTER ::FILDEN,    MOTT

REAL (REAL8), DIMENSION(:), POINTER :: TEMP01, TEMP02, TEMP03, TEMP04
REAL (REAL8), DIMENSION(:), POINTER :: TEMP05, TEMP06, TEMP07, TEMP08
REAL (REAL8), DIMENSION(:), POINTER :: TEMP09, TEMP10, TEMP11, TEMP12
REAL (REAL8), DIMENSION(:), POINTER :: TEMP13, TEMP14, TEMP15, TEMP16
REAL (REAL8), DIMENSION(:), POINTER :: TEMP17, TEMP18, TEMP19, TEMP20
REAL (REAL8), DIMENSION(:), POINTER :: TEMP21, TEMP22, TEMP23, TEMP24
REAL (REAL8), DIMENSION(:), POINTER :: TEMP25, TEMP26, TEMP27, TEMP28
REAL (REAL8), DIMENSION(:), POINTER :: TEMP29, TEMP30, TEMP31, TEMP32
REAL (REAL8), DIMENSION(:), POINTER :: TEMP33

REAL (REAL8), DIMENSION(:), POINTER ::   GVAR, GTVR,  UNIJK
REAL (REAL8), DIMENSION(:,:), POINTER :: DVOL
REAL (REAL8), DIMENSION(:), POINTER :: ULN, DPDL, DRDL
REAL (REAL8), DIMENSION(:), POINTER :: DUXDL, DUYDL, DUZDL, DULDL
REAL (REAL8), DIMENSION(:), POINTER ::   XPP,   YPP,   ZPP
REAL (REAL8), DIMENSION(:), POINTER ::  UXREL,  UYREL, UZREL
REAL (REAL8), DIMENSION(:), POINTER ::   BX,   BY,   BZ
REAL (REAL8), DIMENSION(:), POINTER ::   EB11,EB12,EB13,EB21,EB22,EB23
REAL (REAL8), DIMENSION(:), POINTER ::   EB31,EB32,EB33
REAL (REAL8), DIMENSION(:,:), POINTER ::  CAREA, CDIMT

INTEGER (INT4), PARAMETER :: LIMSBL=3, LIMSBT = 100, LIMSHV = 19
```

```
TYPE SHELL_VAR
  TYPE (REAL_ARRAY_POINTER), DIMENSION(:,:), POINTER :: P
END TYPE
TYPE (SHELL_VAR), DIMENSION(LIMSUB)  :: GRIDSH

TYPE (SMALL_INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB,LIMSBT):: NSMAT, NSFAIL
TYPE (REAL_ARRAY_POINTER), DIMENSION(LIMSUB) :: ZZTP, HHTP
REAL (REAL8), DIMENSION(:), POINTER :: ZZT, HHT

INTEGER , PARAMETER :: LIMPPD=10
INTEGER (INT4)  :: NUMPPD, NPPD
CHARACTER (LEN=8), DIMENSION(LIMPPD) :: NAMPPD
INTEGER (INT4), DIMENSION (LIMSUB,LIMPPD) :: IPPD, JPPD, KPPD, IPPBAS
TYPE (SMALL_INTEGER_ARRAY3_POINTER),DIMENSION(LIMSUB,LIMPPD)::PPPROC
TYPE (INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB,LIMPPD) :: IISPAT
TYPE (INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB,LIMPPD) :: JJSPAT
TYPE (INTEGER_ARRAY_POINTER), DIMENSION(LIMSUB,LIMPPD) :: KKSPAT
```

A.17. MDPP, Parallel Calculation Variables

The module MDPP contains variables relating to the execution of parallel simulations.

IFPP	=1 if AUTODYN is running in parallel
MYTASK	Slave task number (0 is master process)

Published: 2011-10-05

A.18. MDSOLV, Unstructured Entity Types

```
MODULE MDSOLV

USE mdvar_all

IMPLICIT NONE

! THIS MODULE CONTAINS SOLVER DATA FOR EACH PART

! COMMON FLAGS
INTEGER (INT4), PARAMETER :: ISF_SOLVER         = 1

! NODE TYPES
INTEGER(INT4), PARAMETER :: NDTYPE_BASIC              =1
INTEGER(INT4), PARAMETER :: NDTYPE_3DOF               =2
INTEGER(INT4), PARAMETER :: NDTYPE_6DOF_SHELL         =3
INTEGER(INT4), PARAMETER :: NDTYPE_6DOF_BEAM          =4
INTEGER(INT4), PARAMETER :: NDTYPE_6DOF_SHELL_BEAM    =5
INTEGER(INT4), PARAMETER :: NDTYPE_3DOF_ANP           =6
INTEGER(INT4), PARAMETER :: NDTYPE_6DOF_ANP           =7
INTEGER(INT4), PARAMETER :: NDTYPE_ORIENT_BEAM        =80
INTEGER(INT4), PARAMETER :: NDTYPE_EXTERNAL           =99
INTEGER(INT4), PARAMETER :: NDTYPE_ALL                =100

! ALL ELEMENTS FOR PRE-PROCESSING
INTEGER(INT4), PARAMETER :: ELTYPE_BASIC             =99

! SOLID ELEMENT TYPES
INTEGER(INT4), PARAMETER :: ELTYPE_HEX8              =100
INTEGER(INT4), PARAMETER :: ELTYPE_HEX8FE            =101
INTEGER(INT4), PARAMETER :: ELTYPE_PENTA6            =102
INTEGER(INT4), PARAMETER :: ELTYPE_TET4              =103
INTEGER(INT4), PARAMETER :: ELTYPE_TET4_ANP          =104
INTEGER(INT4), PARAMETER :: ELTYPE_PYRAMID5          =105


! SHELL/BEAM ELEMENT TYPES
INTEGER(INT4), PARAMETER :: ELTYPE_SHL4              =200
INTEGER(INT4), PARAMETER :: ELTYPE_SHL3              =201
INTEGER(INT4), PARAMETER :: ELTYPE_SHL4BLT           =202
INTEGER(INT4), PARAMETER :: ELTYPE_BEAM2             =203

! EULER ELEMENT TYPES
INTEGER(INT4), PARAMETER :: ELTYPE_HEX8_EUL          =300
INTEGER(INT4), PARAMETER :: ELTYPE_PENTA6_EUL        =301
INTEGER(INT4), PARAMETER :: ELTYPE_TET4_EUL          =302
INTEGER(INT4), PARAMETER :: ELTYPE_HEX8_FCT          =303

! ALE ELEMENT TYPES
INTEGER(INT4), PARAMETER :: ELTYPE_HEX8_SALE         =400
INTEGER(INT4), PARAMETER :: ELTYPE_HEX8_ALE          =401
INTEGER(INT4), PARAMETER :: ELTYPE_PENTA6_ALE        =402
INTEGER(INT4), PARAMETER :: ELTYPE_TET4_ALE          =403


! STRUCTURED MESH SOLVERS
INTEGER (INT4), PARAMETER ::      ELTYPE_LAG        = 1
INTEGER (INT4), PARAMETER ::      ELTYPE_EUL        = 2
INTEGER (INT4), PARAMETER ::      ELTYPE_ALE        = 3
INTEGER (INT4), PARAMETER ::      ELTYPE_SHL        = 4
INTEGER (INT4), PARAMETER ::      ELTYPE_FCT        = 5
INTEGER (INT4), PARAMETER ::      ELTYPE_SPH        = 6
INTEGER (INT4), PARAMETER ::      ELTYPE_BEAM       = 7
INTEGER (INT4), PARAMETER ::      ELTYPE_RB         = 8
INTEGER (INT4), PARAMETER ::      ELTYPE_ALL        = 8
```

```
! FACE TYPES AS IN OPT(3) = OPT(FC_SOPT_FACETYPE)
INTEGER (INT4), PARAMETER ::      FATYPE_EXTERNAL   = 1
INTEGER (INT4), PARAMETER ::      FATYPE_INTERNAL   = 2
INTEGER (INT4), PARAMETER ::      FATYPE_PLOAD      = 3
INTEGER (INT4), PARAMETER ::      FATYPE_TRANSMIT   = 4
INTEGER (INT4), PARAMETER ::      FATYPE_BASIC      = 99

! FACE TOPOLOGY AS IN OPT(2) = OPT(FC_SOPT_TRIAQUAD)
INTEGER (INT4), PARAMETER ::      FATYPE_PLOAD3     = 3
INTEGER (INT4), PARAMETER ::      FATYPE_PLOAD4     = 4
INTEGER (INT4), PARAMETER ::      FATYPE_PLOAD2     = 5

! JOIN TYPES
INTEGER (INT4), PARAMETER ::      JOINTYPE_BASIC    = 1
INTEGER (INT4), PARAMETER ::      JOINTYPE_IJKUS    = 2

! RIGID BODY TYPES
INTEGER (INT4), PARAMETER ::      RBODTYPE_MATRIG    = 1
INTEGER (INT4), PARAMETER ::      RBODTYPE_BASIC     = 99 !NOT REALLY USED

! CLASSIFICATION FLAGS
INTEGER (INT4), PARAMETER ::      ICLASS_VOLUME   = 1
INTEGER (INT4), PARAMETER ::      ICLASS_SHELL    = 2
INTEGER (INT4), PARAMETER ::      ICLASS_BEAM     = 3
INTEGER (INT4), PARAMETER ::      ICLASS_POINT    = 4

! TOPOLOGY FLAGS
INTEGER (INT4), PARAMETER ::      ITOPO_NODE   = 1
INTEGER (INT4), PARAMETER ::      ITOPO_LINE   = 2
INTEGER (INT4), PARAMETER ::      ITOPO_TRI    = 3
INTEGER (INT4), PARAMETER ::      ITOPO_QUAD   = 4
INTEGER (INT4), PARAMETER ::      ITOPO_TET    = 5
INTEGER (INT4), PARAMETER ::      ITOPO_PYRAMID= 6
INTEGER (INT4), PARAMETER ::      ITOPO_PENTA  = 8
INTEGER (INT4), PARAMETER ::      ITOPO_HEX    = 9

TYPE SOLVER_DEF
  CHARACTER(LEN=12) :: NAME
  INTEGER(INT4) :: CLASS  ! SOLID_ELEM, SHELL_ELEM, BEAM_ELEM
  INTEGER(INT4) :: TOPOLOGY  ! HEX, TET, QUAD
  INTEGER(INT1), DIMENSION(NUM_RVAR_ALL) :: AVAILABLE_RVAR  ! 1=AVAILABLE, 0=NOT
END TYPE
```

A.19. POLGON, Polygon Variable Definitions

Module polgon contains several polygon variables defined for the current problem.

```
MODULE polgon
USE kindef
IMPLICIT NONE
SAVE
INTEGER, PARAMETER :: LIMPOL=100,LIMPPT=4000,LIMINB=100
INTEGER (INT4) :: NUMPOL, NPOLY, NSPOLY, IFBLEN, NUMINB, NINB
INTEGER (INT4) :: IFEULC,IFFCTC,IFGODC
INTEGER (INT4), DIMENSION(LIMPOL) :: NBPOL, NUMPPT
INTEGER (INT4), DIMENSION(LIMPPT) :: INTPOL,IVRPOL,IPOL,JPOL,MPOL
INTEGER (INT4), DIMENSION(LIMINB,2) :: INBVAR
REAL (REAL8), DIMENSION(LIMPOL) :: PVPOR
REAL (REAL8), DIMENSION(LIMPPT), TARGET :: XPOL, YPOL
REAL (REAL8), DIMENSION(LIMINB) :: VARINB
CHARACTER (LEN=10), DIMENSION(LIMPOL) :: NAMPOL
CHARACTER (LEN=10), DIMENSION(LIMINB) :: NAMINB
END MODULE polgon
```

LIMPOL	*not available*
LIMPPT	“
LIMINBLIMINB	“
NUMPOLNUMPOL	“
NPOLY	“
NSPOLY	“
IFBLEN	“
NBPOL	“
NUMPPT	“
INTPOLINTPOL	“
IVRPOL	“
IPOL	“
JPOL	“
MPOL	“
NUMINB	“
NINB	“
INBVAR	“
XPOL	Polygon points X-array
YPOL	Polygon points Y-array
VARINB	*not available*
PVPOR	Polygon porosity array
NAMPOL	Polygon name array
NAMINB	Eul-Lag boundary name array

Published: 2011-10-05

A.20. RUNDEF, Run Variable Definitions

Module rundef contains several run variables defined for the current problem.

```
MODULE rundef
USE kindef
IMPLICIT NONE
SAVE

INTEGER (INT4) :: ISYM, IFINC, IFIMP, IFDATA, IFBAT, IFLOG, NUNITM
INTEGER (INT4) :: NUNITL, NUNITT, NUNITD, ISYMX, ISYMY, ISYMZ, IFIDNT
INTEGER (INT4) :: IFSEC, MAXEXEC=99, MAXPCS=99, MAXSLV=99, NUMPCS
INTEGER (INT4) :: LDEBUG=0
INTEGER (INT4) :: SVLSCYC=0
INTEGER, PARAMETER :: NUNTYP = 20,NUNITS = 21
INTEGER (INT4), DIMENSION(NUNTYP) :: IPOWER
INTEGER (INT4), DIMENSION(NUNTYP,NUNITS) :: IUNITN
CHARACTER (LEN=100) :: ITEMS,DESCR
CHARACTER (LEN=40)  :: TITLE,   HEAD
CHARACTER (LEN=40), DIMENSION(4), TARGET :: COMENT
CHARACTER (LEN=10), DIMENSION(3) :: UNITM,  UNITL,  UNITT
CHARACTER (LEN=2), DIMENSION(3) :: UNITMS, UNITLS, UNITTS
CHARACTER (LEN=8), DIMENSION(NUNTYP,NUNITS) :: UNITTX

END MODULE rundef
```

ISYM	Symmetry switch
IFINC	Incompressible switch (future use)
IFIMP	Implicit time integration switch (future use)
IFDATA	Indicates if data has been modified
IFBAT	Batch mode switch
IFLOG	Log file write switch
NUNITM	Mass unit
NUNITL	Length unit
NUNITT	Time unit
NUNITD	Display Units Switch
IUNITN	*not available*
IPOWER	“
TITLE	Heading (title) for calculation
HEAD	Heading for top right of screen
ITEMS	Text array used to hold menus etc.
UNITM	Character array with mass unit names
UNITL	Character array with length unit names
UNITT	Character array with time unit names
UNITMS	Character array with abbreviated mass unit names
UNITLS	Character array with abbreviated length unit names
UNITTS	Character array with abbreviated time unit names
UNITTX	Character array: pressure, velocity, etc. names

A.21. SUBDEF, Global Part Variable Definitions

Module subdef contains global variables pertaining to Parts.

```
MODULE subdef
USE kindef
IMPLICIT NONE
SAVE

INTEGER (INT4) :: NUMSUB, NUMEUL,  NSP, NUMJON, MAXIJK, NNMZVR, NUMZNS, NDAT1
INTEGER (INT4) :: NUMBOJ, IFIMPT, NERODED
INTEGER (INT4), DIMENSION(LIMSUB+1) :: IJKBAS
INTEGER (INT4), DIMENSION(LIMSUB) :: NUMPRO, NUMI, NUMJ, NUMK, MATSV
INTEGER (INT4), DIMENSION(LIMSUB) :: IDN,JDN,ISIZ,JSIZ,KSIZ,IJKSIZ
INTEGER (INT4), DIMENSION(LIMSUB) :: NSPACK, IOPRT, NUMEUP, NUMLAP, NMZNS
INTEGER (INT4), DIMENSION(LIMSUB) :: IBPRT, IEPRT, JBPRT,  JEPRT, IJKBAZ
INTEGER (INT4), DIMENSION(LIMSUB) :: KBPRT, KEPRT, IFEBVL, NPLTSL
INTEGER (INT1), DIMENSION(LIMSUB,LIMSUB), TARGET :: JONSUB, MAPPED
INTEGER (INT4), DIMENSION(3) :: IJKMAX
INTEGER (INT1), DIMENSION(LIMSUB) :: IACTIV, IVOLOP
INTEGER (INT1), DIMENSION(LIMSUB,LIMSUB) :: IFSLAV
REAL (REAL8) :: TOLJON, TOTMAX, TOTWRK, TOTXIM, TOTYIM, TOTZIM, TOTHI
REAL (REAL8) :: TOTMSB, TOTVLB, TOTIEB, TOTKEB, TOTDEB
REAL (REAL8) :: TOTMS, TOTVL, TOTIE, TOTKE, TOTDE
REAL (REAL8) :: TOTXMB, TOTYMB, TOTZMB, TOTXM, TOTYM, TOTZM
REAL (REAL8), DIMENSION(LIMSUB), TARGET :: XSUBMN, YSUBMN, ZSUBMN
REAL (REAL8), DIMENSION(LIMSUB), TARGET :: XSUBMX, YSUBMX, ZSUBMX
REAL (REAL8), DIMENSION(LIMSUB), TARGET :: USUBMX, ASUBMX, SUBMS,  SUBVL
REAL (REAL8), DIMENSION(LIMSUB), TARGET :: VARSB1, VARSB2, VARSB3, VARSB4, SUBDE
REAL (REAL8), DIMENSION(LIMSUB), TARGET :: SUBMSB, SUBVLB, SUBDEB, SUBIE
REAL (REAL8), DIMENSION(LIMSUB), TARGET :: SUBKE, SUBXM, SUBYM, SUBZM, SUBIEB
REAL (REAL8), DIMENSION(LIMSUB), TARGET :: SUBKEB, SUBXMB, SUBYMB, SUBZMB
REAL (REAL8), DIMENSION(LIMSUB), TARGET :: ACTIME, DCTIME, TSTPFC
REAL (REAL8), DIMENSION(LIMSUB), TARGET ::  XNMIN,  XNMAX,  YNMIN,  YNMAX,  ZNMIN,
                                                                         ZNMAX
REAL (REAL8), DIMENSION(LIMSUB) ::  AVLEN
REAL (REAL8) :: AVLENL
CHARACTER(LEN=10), DIMENSION(LIMSUB) :: NAMSUB
CHARACTER(LEN=1), DIMENSION(LIMSUB) :: NEWSUB
INTEGER (INT4) :: NSUB, NPROC, IMAX, JMAX, KMAX, NWRKSB, NPLTSB, NZSUB, NSHTYP
INTEGER (INT4) :: NSBBEG, NSBEND, NSBLAY, MATLOC, IMAXP, JMAXP, KMAXP, KMAXBM
INTEGER (INT4) :: NPROE, NPROL, NUMIJK, NSBOLD, IFDEZN
REAL (REAL8) :: DUMMYV, ALERTI, ALERTJ, ALERTK, ALEFRX
CHARACTER(LEN=10) :: DUMMYN

END MODULE subdef
```

NSUB	Current Part number
NPROC	Processor type (Lagrange, Euler, etc.) for current Part
IMAX	Maximum I-index for current Part
JMAX	Maximum J-index for current Part
KMAX	Not used in 2D
NSBLAY	Number of sublayers (shell Parts)
MATLOC	Material location (shell Parts)
IMAXP	IMAX + 1
JMAXP	JMAX + 1

Published: 2011-10-05

ALERTI	I-line spacing ratio (ALE Parts)
ALERTJ	J-line spacing ratio (ALE Parts)
ALEFRX	Relaxation coefficient (ALE Parts)

LIMSUB	Limit on number of Parts
NUMSUB	Number of Parts in problem
NUMEUL	Number of Euler Parts in problem
NSP	Current data page
NUMPRO	Processor types for Parts
NUMI	Maximum I index for Parts
NUMJ	Maximum J index for Parts
IJKBASE	Base addresses for Parts
IOPRT	Order of printout for each Part
IBPRT	Index ranges for Part prints
IEPRT	"
JBPRT	"
JEPRT	"
XSUBMN	(X,Y) ranges for Parts
YSUBMN	"
XSUBMX	"
YSUBMX	"
USUBMX	Maximum velocity in Part
ASUBMX	Maximum cell area in Part
VARSB1	*not available*
VARSB2	“
VARSB3	“
ACTIME	Activity times array by Part
NAMSUB	Part names
NEWSUB	Indicates if Part is newly created
TOTMAX	Total energy
TOTWRK	Total work
TOTXIM	Total X-impulse
TOTYIM	Total Y-impulse
TOTHI	Total hoop impulse
TOTMSB	Total mass at t(n)
TOTVLB	Total volume at t(n)
TOTIEB	Total internal energy at t(n)
TOTKEB	Total kinetic energy at t(n)
TOTDEB	Total distortional energy at t(n)

TOTXMB	Total X-momentum at t(n)
TOTYMB	Total Y-momentum at t(n)
TOTMS	Total mass at t(n+1)
TOTVL	Total volume at t(n+1)
TOTIE	Total internal energy at t(n+1)
TOTKE	Total kinetic energy at t(n+1)
TOTDE	Total distortional energy at t(n+1)
TOTXM	Total X-momentum at t(n+1)
TOTYM	Total Y-momentum at t(n+1)
SUBMS	Part masses at t(n+1)
SUBVL	Part volume at t(n+1)
SUBDE	Part distortional energy at t(n+1)
SUBMSB	Part masses at t(n)
SUBVLB	Part volume at t(n)
SUBDEB	Part distortional energy at t(n)
SUBIE	Part internal energy at t(n+1)
SUBKE	Part kinetic energy at t(n+1)
SUBXM	Part X-momentum at t(n+1)
SUBYM	Part Y-momentum at t(n+1)
SUBIEB	Part internal energy at t(n)
SUBKEB	Part kinetic energy at t(n)
SUBXMB	Part X-momentum at t(n)
SUBYMB	Part Y-momentum at t(n)
NUMJON	Joined Part switch
JONSUB	Joined Part array
TOLJON	Joined Part tolerance
MATSV	*not available*

Published: 2011-10-05

A.22. WRAPUP, Execution Termination Variables

```
MODULE wrapup
USE kindef
IMPLICIT NONE

SAVE

INTEGER (INT4) :: NCYLIM, NSWRAP, NCYREF
INTEGER (INT4) :: IDEGEN, JDEGEN, KDEGEN, MDEGEN
REAL (REAL8) :: TIMLIM, ENFRAC

END MODULE wrapup
```

NCYLIM	Cycle limit for wrapup
NSWRAP	Wrapup switch
NCYREF	Energy reference cycle
TIMLIM	Time limit for wrapup
ENFRAC	Energy fraction for wrapup
IDEGEN	I index for degenerate cell on wrapup
JDEGEN	J index for degenerate cell on wrapup
KDEGEN	Not used for 2D
MDEGEN	Part number for degenerate cell on wrapup

A.23. OBJECT, SPH Object Definitions

Module object contains data pertaining to SPH objects that are used for initializing the model.

```
MODULE object
USE kindef
IMPLICIT NONE

SAVE

INTEGER, PARAMETER :: LIMOBJ = 100
INTEGER, PARAMETER :: LIMPTS = 112
INTEGER, PARAMETER :: LIMOBC = 6
INTEGER, PARAMETER :: LIMSET = 100
INTEGER, PARAMETER :: LIMSPH = 500000

INTEGER (INT4) :: NUMOBJ
INTEGER (INT4), DIMENSION(LIMOBJ) :: NOBJC , NOBJT, NOBJP, NOBJS
INTEGER (INT4), DIMENSION(LIMOBJ) :: NSPHOB, MATOBJ, IFACOB, MATSET
INTEGER (INT4), DIMENSION(LIMOBJ) :: NTRIOB, OBJBND
INTEGER (INT4) :: NOBJ, NOBTYP, NOBCOL, MAXOB, MAXSET, NUMSET
INTEGER (INT4) :: OSIM
REAL (REAL8), DIMENSION(LIMOBJ) :: ACTOBJ, DCTOBJ
REAL (REAL8), DIMENSION(LIMOBJ) :: OBJA  , RPSZOB
REAL (REAL8), DIMENSION(LIMOBJ,3) :: OBJO, OBJN
REAL (REAL8), DIMENSION(LIMOBJ,10) :: OBJS
REAL (REAL8), DIMENSION(LIMOBJ,LIMPTS) :: XOBJ, YOBJ, ZOBJ
REAL (REAL8), DIMENSION(LIMSET) :: UXNOBJ, UYNOBJ, UZNOBJ
REAL (REAL8), DIMENSION(LIMSET) :: URNOBJ, RHOOBJ, ENOBJ
REAL (REAL8) :: XORG, YORG, ZORG, XSIZ, YSIZ, ZSIZ, RSIZ, THETA, THETA0
REAL (REAL8) :: XDIRN, YDIRN, ZDIRN, ANGOBJ, ROUT1, RIN1, ROUT2, RIN2
REAL (REAL8) :: ROUT1Y, RIN1Y, ROUT2Y, RIN2Y, ROUT1Z, RIN1Z, ROUT2Z, RIN2Z
REAL (REAL8) :: RSIZIN, RCOUT, RCIN, CLEN, CTHICK
CHARACTER (LEN=1),  DIMENSION(LIMOBJ) :: YONO
CHARACTER (LEN=10), DIMENSION(LIMSET) :: NAMSET
CHARACTER (LEN=12), DIMENSION(LIMOBJ) :: NAMOBJ
CHARACTER (LEN=12) :: NAMEO

! NAMOBJ(LIMOBJ) - object NAME
! NAMEO - CURRENT object NAME
! NUMOBJ     - NUMBER OF objectS IN SPH SUBGRID
! NOBJ       - CURRENT object NUMBER
! NOBJT(LIMOBJ)  - object TYPE
! NOBTYP - CURRENT object TYPE
! NOBJC(LIMOBJ)  - object COLOUR
! NOBCOL - CURRENT object COLOUR
! NOBJP(LIMOBJ)  - NUMBER OF POINTS IN object POLYGON
! IFACOB(LIMOBJ) - object ACTIVE (PACK) INDICATOR
! OBJO(LIMOBJ,2) - object ORIGIN COORDS
! XORG, YORG, ZORG - CURRENT object X, Y AND Z ORIGIN
! OBJS(LIMOBJ,3) - MAXIMUM object SIZE
! XSIZ, YSIZ, ZSIZ - CURRENT object MAXIMUM SIZES
! OBJN((LIMOBJ,3) - object PRINCI3PAL DIRECTION
! OBJA((LIMOBJ,3) - object ANGLE (PRINCI3PAL DIRECTION)
! XOBJ(LIMOBJ,LIMPTS) - X COORD. OF POINTS IN object POLYGON
! YOBJ(LIMOBJ,LIMPTS) - Y COORD. OF POINTS IN object POLYGON
! ZOBJ(LIMOBJ,LIMPTS) - Z COORD. OF POINTS IN object POLYGON
! NSPHOB(LIMOBJ) - NUMBER OF SPH NODES PACKED IN object
! RPSZOB(LIMOBJ) - PARTICLE SIZE IN object
! MAXOB  - TOTAL NUMBER OF SPH objectS

! MATERIAL SET VARIBALES
! MATSET(LIMOBJ) - MATERIAL SET FOR object
! MAXSET     - NUMBER OF MATERIAL SETS
! MATOBJ(LIMSET) - MATERIAL NUMBER FOR MATERIAL ASSIGNED TO SET
```


Published: 2011-10-05

```
! UXNOBJ(LIMSET) - INITIAL X-VELOCITY ASSIGNED TO MATERIAL SET
! UYNOBJ(LIMSET) - INITIAL Y-VELOCITY ASSIGNED TO MATERIAL SET
! UZNOBJ(LIMSET) - INITIAL Z-VELOCITY ASSIGNED TO MATERIAL SET
! URNOBJ(LIMSET) - INITIAL R-VELOCITY ASSIGNED TO MATERIAL SET
! RHOOBJ(LIMSET) - INITIAL DENSITY ASSIGNED TO MATERIAL SET
! ENOBJ(LIMSET)  - INITIAL INTERN3AL ENERGY ASSIGNED TO MATERIAL SET

END MODULE object
```

NAMOBJ(LIMOBJ)	object name
NAMEO	current object name
NUMOBJ	number of objects in sph Part
NOBJ	current object number
NOBJT(LIMOBJ)	object type
NOBTYP	current object type
NOBJC(LIMOBJ)	object color
NOBCOL	current object color
NOBJP(LIMOBJ)	number of points in object polygon
IFACOB(LIMOBJ)	object active (pack) indicator
OBJO(LIMOBJ,2)	object origin coords
XORG, YORG	current object x and y origin
OBJS(LIMOBJ,2)	maximum object size
XSIZ, YSIZ	current object maximum sizes
XOBJ(LIMOBJ,LIMPTS)	x coord. of points in object polygon
YOBJ(LIMOBJ,LIMPTS)	y coord. of points in object polygon
NSPHOB(LIMOBJ)	number of sph nodes packed in object
MAXOB	total number of sph objects
Material Set Variables	
MATSET(LIMOBJ)	material set for object
MAXSET	number of material sets
MATOBJ(LIMSET)	material number for material assigned to set
UXNOBJ(LIMSET)	initial x-velocity for a set
UYNOBJ(LIMSET)	initial y-velocity for a set
URNOBJ(LIMSET)	initial r-velocity for a set
RHOOBJ(LIMSET)	initial density assigned to material set
ENOBJ(LIMSET)	initial internal energy assigned to material set

Published: 2011-10-05

Appendix B. AUTODYN Variables Listings

The AUTODYN variables for structured 2D and 3D are listed below. The external output name is first given. This is the name shown on plots, printout, and when interactively examining values on the screen. The next column is the array name used internally. These internal names are to be used when writing user subroutines. Grid variables are either associated with a node (e.g. X coordinate), or a cell center (e.g. Pressure), or with a particle (SPH only). These are indicated as Node, Cell, and Particle respectively. Depending on the processor (solver), certain variables are not defined. This is indicated by a blank entry in the table.

The listings are provided for both versions 4(Fortran 90) and versions 3(Fortran 77)

B.1. AUTODYN-2D – Structured (IJK) Solvers

AUTODYN output name	Internal Array	Lag/ALE	Euler	Shell	Godunov	FCT	SPH	Description	Note
X	XN	Node	Node	Node	Node	Node	Particle	X space co-ordinate	
Y	YN	Node	Node	Node	Node	Node	Particle	Y space co-ordinate	
X-VELOCITY	UXN	Node	Cell	Node	Cell	Cell	Particle	X component of velocity	
Y-VELOCITY	UYN	Node	Cell	Node	Cell	Cell	Particle	Y component of velocity	
X-FORCE	FX	Node		Node			Particle	X component of force	
Y-FORCE	FY	Node		Node			Particle	Y component of force	
NODE-MASS	PMASS	Node		Node				Nodal mass	
I.P.INDEX	RINTER	Node		Node				Interactive point index	
VOLUME	VOLN	Cell	Cell	Segment	Cell		Particle	Volume	
CELL-MASS	CMASS	Cell	Cell	Segment	Cell	Cell	Particle	Cell mass	
MASS	CMS				Cell	Cell		Material mass in cell	F
COMPRESS.	XMU	Cell	Cell		Cell	Cell	Particle	Compression	F
COMPRESS	CMU				Cell	Cell		Material compression	F
INT.ENERGY	EN	Cell	Cell		Cell	Cell	Particle	Internal energy	F
INT.ENERGY	CEN				Cell	Cell		Material internal energy	F
PRESSURE	PN	Cell	Cell		Cell	Cell	Particle	Pressure	
PSEUDO.V.Q	Q	Cell	Cell				Particle	Artificial viscosity	
DIS.ENERGY	SDE	Cell	Cell	Segment	Cell		Particle	Specific distortional energy	
DENSITY	DEN	Cell	Cell		Cell	Cell	Particle	Density	
TEMP.	TEMP	Cell	Cell	Segment	Cell	Cell	Particle	Temperature	F
TEMP	CTP				Cell	Cell		Material temperature	F
EFF.PL.STN	EPS	Cell	Cell	Segment	Cell		Particle	Effective plastic strain	E
E.P.S.RATE	EPSDOT	Cell	Cell	Segment	Cell		Particle	Effective plastic strain rate	E
EFFECT.STN	EFS	Cell	Cell	Segment	Cell		Particle	Effective strain	E
SOUNDSPEED	SSPD	Cell	Cell	Segment	Cell	Cell	Particle	Local sound speed	
DAMAGE	DAM	Cell	Cell	Segment	Cell		Particle	Damage	C
DIVERGENCE	DIV	Cell	Cell		Cell		Particle	Divergence	
ALPHA	ALPHA	Cell	Cell		Cell	Cell	Particle	Material model variable	A,F
ALPHA	CAL				Cell	Cell		Material Alpha	F
P.STRESS.1	T11	Cell					Particle	Total Principal stress 1	
P.STRESS.2	T22	Cell					Particle	Total Principal stress 2	
STRESS.12	T12	Cell					Particle	Total Principal shear stress 12	
STRESS TXX	TXX	Cell	Cell		Cell		Particle	Tot stress tensor, XX component	
STRESS TYY	TYY	Cell	Cell		Cell		Particle	Tot stress tensor, YY component	
STRESS TXY	TXY	Cell	Cell		Cell		Particle	Tot stress tensor, XY component	
STRESS TTT	TTT	Cell	Cell		Cell		Particle	Tot stress tensor, TT component	B
MIS.STRESS	TVM	Cell	Cell		Cell		Particle	Von Mises stress	

Published: 2011-10-05

AUTODYN output name	Internal Array	Lag/ALE	Euler	Shell	Godunov	FCT	SPH	Description	Note
YLD.STRESS	YIELD	Cell	Cell		Cell		Particle	Current yield stress	
EXXDOT	EXXD	Cell	Cell		Cell		Particle	Total strain rate, XX component	
EYYDOT	EYYD	Cell	Cell		Cell		Particle	Total strain rate, YY component	
EXYDOT	EXYD	Cell	Cell		Cell		Particle	Total strain rate, XY component	
P.STRAIN.1	STN11	Cell					Particle	Total principal strain 1	
P.STRAIN.2	STN22	Cell					Particle	Total principal strain 2	
P.STRAIN.3	STN33	Cell					Particle	Total principal strain 3	B
STRAIN.12	STN12	Cell					Particle	Total principal shear strain 12	
P.ST.ANG	PSANG	Cell					Particle	Angle of principal stress	
VOID FRAC.	VOID		Cell		Cell			Volume fraction of void	
COVERED-V	FCOVRV		Cell					Volume cover fraction	
COVERED-I	FCOVRI		Cell					I-face cover fraction	
COVERED-J	FCOVRJ		Cell					J-face cover fraction	
DPDX	DPDX				Cell			Pressure slope in X direction	
DPDY	DPDY				Cell			Pressure slope in Y direction	
DRDX	DRDX				Cell			Density slope in X direction	
DRDY	DRDY				Cell			Density slope in Y direction	
DUXDX	DUXDX				Cell			X velocity slope in X direction	
DUXDY	DUXDY				Cell			X velocity slope in Y direction	
DUYDX	DUYDX				Cell			Y velocity slope in X direction	
DUYDY	DUYDY				Cell			Y velocity slope in Y direction	
HOOP STN.	STN1			Segment				Hoop strain	
LONG.STN.	STN2			Segment				Longitudinal strain	
HOOP.STR.	STR1			Segment				Hoop stress	
LONG.STR.	STR2			Segment				Longitudinal stress	
STR.RES.1	SRES1			Segment				Stress resultant in direction 1	D
STR.RES.2	SRES2			Segment				Stress resultant in direction 2	D
BEND.MOM.1	BMOM1			Segment				Bending moment in direction 1	D
BEND.MOM.2	BMOM2			Segment				Bending moment in direction 2	D
TRANS.SHR.	TSHEAR			Segment				Transverse shear	
THICKNESS	THICK			Segment				Thickness (of shell)	
SMOOTH.LEN	SML						Particle	SPH smoothing length	
VORTICITY	VOR						Particle	SPH vorticity	
OLDDENSITY	DENNM1						Particle	Previous density for SPH	
CUTOFF.RAD	RCUT						Particle	SPH cutoff radius	
NO.NEIGH.	RNON						Particle	Number of SPH neighbors	
ABS.VEL	ABSVEL	Node	Cell	Node	Cell	Cell	Particle	Absolute velocity magnitude	
SOFT.SLOPE	HNORM	Cell					Particle	Crack softening slope	
FAIL.STRES	TFAIL	Cell					Particle	Crack softening failure stress	
DIL.PRESS	PDIL	Cell	Cell		Cell		Particle	Johnson-Holmquist dilatation press	

AUTODYN output name	Internal Array	Lag/ALE	Euler	Shell	Godunov	FCT	SPH	Description	Note
DIL.ENERGY	EDIL	Cell	Cell		Cell		Particle	Johnson-Holmquist dilatation en.	
EPSPRE		Cell	Cell		Cell		Particle	RHT pre-softening plastic strain	
FRATE	FRATE	Cell	Cell		Cell		Particle	RHT strain rate enhancement	
RTHIRD	RTHIRD	Cell	Cell		Cell		Particle	RHT lode angle	
FCAP	FCAP	Cell	Cell		Cell		Particle	RHT elastic cap factor	
EPSDO	EPSDO	Cell	Cell		Cell		Particle	RHT strain rate at previous cycle	
PCOR11	PCOR11	Cell					Particle	AMMHIS pressure correction	
PCOR22	PCOR22	Cell					Particle	AMMHIS pressure correction	
PCOR33	PCOR33	Cell					Particle	AMMHIS pressure correction	
MASS FACT.	FMASS	Node					Particle	SPH joined face mass	
VTXX	VTXX	Cell	Cell		Cell		Particle	Viscoelastic stress	
VTYY	VTYY	Cell	Cell		Cell		Particle	Viscoelastic stress	
VTXY	VTXY	Cell	Cell		Cell		Particle	Viscoelastic stress	
IGTIME	IGTIME	Cell					Particle	Slow burn ignition time	
SBRCRT	SBRCRT	Cell					Particle	Slow burn reaction ratio	
INITIAL X	XN0	Node		Node			Particle	Original X space co-ordinate	
INITIAL Y	YN0	Node		Node			Particle	Original Y space co-ordinate	
GAS.PRESS	PGAS	Cell	Cell		Cell	Cell	Particle	Slow-burn gas pressure	
FILL.DENS.	FILDEN	Cell	Cell		Cell	Cell	Particle	Slow-burn fill density	
VAR.1	VAR01	X	X	X	X	X	X	User defined variable 1	
VAR.2	VAR02	X	X	X	X	X	X	User defined variable 2	
VAR.3	VAR03	X	X	X	X	X	X	User defined variable 3	
VAR.4	VAR04	X	X	X	X	X	X	User defined variable 4	
VAR.5	VAR05	X	X	X	X	X	X	User defined variable 5	
VAR.6	VAR06	X	X	X	X	X	X	User defined variable 6	
VAR.7	VAR07	X	X	X	X	X	X	User defined variable 7	
VAR.8	VAR08	X	X	X	X	X	X	User defined variable 8	
VAR.9	VAR09	X	X	X	X	X	X	User defined variable 9	
VAR.10	VAR10	X	X	X	X	X	X	User defined variable 10	
VAR.11	VAR11	X	X	X	X	X	X	User defined variable 11	
VAR.12	VAR12	X	X	X	X	X	X	User defined variable 12	
VAR.13	VAR13	X	X	X	X	X	X	User defined variable 13	
VAR.14	VAR14	X	X	X	X	X	X	User defined variable 14	
VAR.15	VAR15	X	X	X	X	X	X	User defined variable 15	
VAR.16	VAR16	X	X	X	X	X	X	User defined variable 16	
VAR.17	VAR17	X	X	X	X	X	X	User defined variable 17	
VAR.18	VAR18	X	X	X	X	X	X	User defined variable 18	
VAR.19	VAR19	X	X	X	X	X	X	User defined variable 19	
VAR.20	VAR20	X	X	X	X	X	X	User defined variable 20	

The index value for a given grid variable is N*name*, where "*name*" is the internal variable name. For example, if you wanted the index for x-velocity(UXN) it would be NUXN. This index could then be used in the various AUTODYN functions (e.g. GV(NUXN,IJK).

Published: 2011-10-05

B.2. AUTODYN-3D – Structured (IJK) Solvers

AUTODYN output name	Internal Array	Lag/ALE	Shell	Euler Godunov	Euler FCT	SPH	Beam	Description	Note
X	XN	Node	Node	Node	Node	Particle	Node	X space co-ordinate	
Y	YN	Node	Node	Node	Node	Particle	Node	Y space co-ordinate	
Z	ZN	Node	Node	Node	Node	Particle	Node	Z space co-ordinate	
X-VELOCITY	UXN	Node	Node	Cell	Cell	Particle	Node	X component of velocity	
Y-VELOCITY	UYN	Node	Node	Cell	Cell	Particle	Node	Y component of velocity	
Z-VELOCITY	UZN	Node	Node	Cell	Cell	Particle	Node	Z component of velocity	
X-FORCE	FX	Node	Node			Particle	Node	X component of force	
Y-FORCE	FY	Node	Node			Particle	Node	Y component of force	
Z-FORCE	FZ	Node	Node			Particle	Node	Z component of force	
NODE-MASS	PMASS	Node	Node			Particle	Node	Nodal mass	
VOLUME	VOLN	Cell	Element	Cell		Particle	Element	Volume	
VOLUME FRACTION	CVF	Cell	Element	Cell				Material volume fraction	F,G
MASS	CMS	Cell	Element	Cell	Cell	Particle	Element	Material mass in cell	F,G
COMPRESS	CMU	Cell		Cell	Cell	Particle		Material compression	F,G
INT.ENERGY	CEN	Cell		Cell	Cell	Particle	Element	Material internal energy	F,G
PRESSURE	PN	Cell		Cell	Cell	Particle		Pressure	
PSEUDO.V.Q	Q	Cell				Particle		Artificial viscosity	
PLASTIC.WK	PLWK	Cell	Element	Cell		Particle	Element	Specific plastic work	
DENSITY	DEN	Cell		Cell	Cell	Particle	Element	Density	
TEMP	CTP	Cell	Element	Cell	Cell	Particle	Element	Material temperature	F,G
EFF.PL.STN	CPS	Cell	Element	Cell		Particle	Element	Material effective plastic strain	E,F,G
E.P.S.RATE	EPSDOT	Cell	Element	Cell		Particle	Element	Effective plastic strain rate	E
EFFECT.STN	EFS	Cell	Element	Cell		Particle	Element	Effective strain	E
SOUNDSPEED	SSPD	Cell	Element	Cell	Cell	Particle	Element	Local sound speed	
DIVERGENCE	DIV	Cell		Cell		Particle		Divergence	
ALPHA	CAL	Cell		Cell	Cell	Particle		Material Alpha	F,G
P.STRESS.1	T11	Cell				Particle		Total principal stress 1	
P.STRESS.2	T22	Cell				Particle		Total principal stress 2	
P.STRESS.3	T33	Cell				Particle		Total principal stress 3	
STRESS TXX	TXX	Cell		Cell		Particle		Tot stress tensor, XX component	
STRESS TYY	TYY	Cell		Cell		Particle		Tot stress tensor, YY component	
STRESS TZZ	TZZ	Cell		Cell		Particle		Tot stress tensor, ZZ component	
STRESS TXY	TXY	Cell		Cell		Particle		Tot stress tensor, XY component	
STRESS TYZ	TYZ	Cell		Cell		Particle		Tot stress tensor, YZ component	
STRESS TZX	TZX	Cell		Cell		Particle		Tot stress tensor, ZX component	
MIS.STRESS	TVM	Cell	Element	Cell		Particle	Element	Von Mises stress	
YLD.STRESS	YIELD	Cell	Element	Cell		Particle	Element	Current yield stress	

Published: 2011-10-05

AUTODYN output name	Internal Array	Lag/ALE	Shell	Euler Godunov	Euler FCT	SPH	Beam	Description	Note
EXXDOT	EXXD	Cell		Cell		Particle		Total strain rate, XX component	
EYYDOT	EYYD	Cell		Cell		Particle		Total strain rate, YY component	
EZZDOT	EZZD	Cell		Cell		Particle		Total strain rate, ZZ component	
EXYDOT	EXYD	Cell		Cell		Particle		Total strain rate, XY component	
EYZDOT	EYZD	Cell		Cell		Particle		Total strain rate, YZ component	
EZXDOT	EZXD	Cell		Cell		Particle		Total strain rate, ZX component	
ANG.X.VEL	WXN		Node				Node	X-angular velocity	
ANG.Y.VEL	WYN		Node				Node	Y-angular velocity	
ANG.Z.VEL	WZN		Node				Node	Z-angular velocity	
INERTIA1	RI11		Node				Node	Inertia about local 11 axis	
INERTIA2	RI22		Node				Node	Inertia about local 22 axis	
INERTIA3	RI33		Node				Node	Inertia about local 33 axis	
VOID FRAC.	VOID			Cell				Volume fraction of void	
DPDX	DPDX			Cell				Pressure slope in X direction	
DPDY	DPDY			Cell				Pressure slope in Y direction	
DPDZ	DPDZ			Cell				Pressure slope in Z direction	
DRDX	DRDX			Cell				Density slope in X direction	
DRDY	DRDY			Cell				Density slope in Y direction	
DRDZ	DRDZ			Cell				Density slope in Z direction	
DUXDX	DUXDX			Cell				X velocity slope in X direction	
DUXDY	DUXDY			Cell				X velocity slope in Y direction	
DUXDZ	DUXDZ			Cell				X velocity slope in Z direction	
DUYDX	DUYDX			Cell				Y velocity slope in X direction	
DUYDY	DUYDY			Cell				Y velocity slope in Y direction	
DUYDZ	DUYDZ			Cell				Y velocity slope in Z direction	
DUZDX	DUZDX			Cell				Z velocity slope in X direction	
DUZDY	DUZDY			Cell				Z velocity slope in Y direction	
DUZDZ	DUZDZ			Cell				Z velocity slope in Z direction	
STRAIN.1	STN1		Element				Element	Total strain 1	
STRAIN.2	STN2		Element					Total strain 2	
STRAIN.12	STN12		Element					Total strain 12	
STRESS.1	STRS1		Element				Element	Total stress 1	
STRESS.2	STRS2		Element					Total stress 2	
STRESS.12	STRS12		Element					Total stress 12	
STR.RES.1	SRES1		Element					Stress resultant in direction 1	
STR.RES.2	SRES2		Element					Stress resultant in direction 2	
BEND.MOM.1	BMOM1		Element					Bending moment in direction 1	
BEND.MOM.2	BMOM2		Element					Bending moment in direction 2	
BEND.MOM.12	BMOM12		Element					Bending moment in direction 12	
THICKNESS	THICK		Element					Thickness (of shell)	
DIRNX	DIRNX	Cell		Cell		Particle		Principal direction – x (Ortho and directional failure)	

Published: 2011-10-05

AUTODYN output name	Internal Array	Lag/ALE	Shell	Euler Godunov	Euler FCT	SPH	Beam	Description	Note
DIRNY	DIRNY	Cell		Cell		Particle		Principal direction – y (Ortho and directional failure)	
DIRNZ	DIRNZ	Cell		Cell		Particle		Principal direction – z (Ortho and directional failure)	
T12	T12	Cell		Cell		Particle		Stress 12	
T23	T23	Cell		Cell		Particle		Stress 23	
T31	T31	Cell		Cell		Particle		Stress 31	
STN11	STN11	Cell		Cell		Particle		Strain 11	
STN22	STN22	Cell		Cell		Particle		Strain 22	
STN33	STN33	Cell		Cell		Particle		Strain 33	
STN12V	STN12V	Cell		Cell		Particle		Strain 12	
STN23	STN23	Cell		Cell		Particle		Strain 23	
STN31	STN31	Cell		Cell		Particle		Strain 31	
RIJOIN	RIJOIN			Cell	Cell			I Join	
RJJOIN	RJJOIN			Cell	Cell			J Join	
RKJOIN	RKJOIN			Cell	Cell			K Join	
COSFI	COSFI		Element					Rotation correction cosine	
SINFI	SINFI		Element					Rotation correction sine	
PSANG	PSANG	Cell	Element	Cell		Particle		Principal angle	
BEAM-AREA	BMAREA						Element	Beam cross-sectional area	
RJJ	RJJ						Element	Beam 11 inertia	
AXIAL.FRC	FAXI						Element	Beam axial force	
TORT.FRC	FTOR						Element	Beam torsion moment	
B.MOM.YI	BMOMYI						Node	Moment about 22 at node IJK-1	
B.MOM.YJ	BMOMYJ						Node	Moment about 22 at node IJK	
B.MOM.ZI	BMOMZI						Node	Moment about 33 at node IJK-1	
B.MOM.ZJ	BMOMZJ						Node	Moment about 33 at node IJK	
BOD.B.V.11	BBV11		Node				Node	Body base vector 1 component x	
BOD.B.V.12	BBV12		Node				Node	Body base vector 1 component y	
BOD.B.V.13	BBV13		Node				Node	Body base vector 1 component z	
BOD.B.V.21	BBV21		Node				Node	Body base vector 2 component x	
BOD.B.V.22	BBV22		Node				Node	Body base vector 2 component y	
BOD.B.V.23	BBV23		Node				Node	Body base vector 2 component z	
BOD.B.V.31	BBV31		Node				Node	Body base vector 3 component x	
BOD.B.V.32	BBV32		Node				Node	Body base vector 3 component y	
BOD.B.V.33	BBV33		Node				Node	Body base vector 3 component z	
ELM.B.V.11	EBV11						Node	Element base vector 1 component x	
ELM.B.V.12	EBV12						Node	Element base vector 1 component y	
ELM.B.V.13	EBV13						Node	Element base vector 1 component z	
ELM.B.V.21	EBV21						Node	Element base vector 2 component x	

AUTODYN output name	Internal Array	Lag/ALE	Shell	Euler Godunov	Euler FCT	SPH	Beam	Description	Note
ELM.B.V.22	EBV22						Node	Element base vector 2 component y	
ELM.B.V.23	EBV23						Node	Element base vector 2 component z	
ELM.B.V.31	EBV31						Node	Element base vector 3 component x	
ELM.B.V.32	EBV32						Node	Element base vector 3 component y	
ELM.B.V.33	EBV33						Node	Element base vector 3 component z	
SMOOTH.LEN	SML					Particle		Smoothing Length	
OLDDENSITY	DENNM1					Particle		Density on previous cycle	
NO.NEIGH	NON					Particle		Number of neighbours	
VORTCTY.X	VORX					Particle		Vorticity around X axis	
VORTCTY.Y	VORY					Particle		Vorticity around Y axis	
VORTCTY.Z	VORZ					Particle		Vorticity around Z axis	
HQM.1	HQM1		Element				Element	Hourglass damping moment about 11 axis	
HQM.2	HQM2		Element				Element	Hourglass damping moment about 22 axis	
HQB.1	HQB1		Element				Element	Hourglass damping force 1	
HQB.2	HQB2		Element				Element	Hourglass damping force 2	
HQB.3	HQB3		Element				Element	Hourglass damping force 12	
F.COVER.I	FCOVRI				Cell			I-face cover fraction	
F.COVER.J	FCOVRJ				Cell			J-face cover fraction	
F.COVER.K	FCOVRK				Cell			K-face cover fraction	
F.COVER.V	FCOVRV				Cell			Cell cover volume	
RBLEND	RBLEND				Cell			FCT blend fraction	
DIL.PRES	PDIL	Cell		Cell		Particle		JH2 Pressure due to bulking	
DAM.ENERG	EDIL	Cell		Cell		Particle		JH2 Distortional energy due to damage	
EPSPRE	EPSPRE	Cell		Cell		Particle		RHT pre-softening plastic strain	
FRATE	FRATE	Cell		Cell		Particle		RHT strain rate enhancement	
RTHIRD	RTHIRD	Cell		Cell		Particle		RHT lode angle	
FCAP	FCAP	Cell		Cell		Particle		RHT elastic cap factor	
EPSDO	EPSDO	Cell		Cell		Particle		RHT strain rate at previous cycle	
FAIL.STRESS	TFAIL	Cell				Particle		Crack softening failure stress	
SOFT.SLOPE	HNORM	Cell				Particle		Crack softening slope	
PCOR11	PCOR11	Cell				Particle		AMMHIS pressure correction	
PCOR22	PCOR22	Cell				Particle		AMMHIS pressure correction	
PCOR33	PCOR33	Cell				Particle		AMMHIS pressure correction	
VTXX	VTXX	Cell				Particle		Viscoelastic stress	
VTYY	VTYY	Cell				Particle		Viscoelastic stress	
VTZZ	VTZZ	Cell				Particle		Viscoelastic stress	
VTXY	VTXY	Cell				Particle		Viscoelastic stress	

AUTODYN output name	Internal Array	Lag/ALE	Shell	Euler Godunov	Euler FCT	SPH	Beam	Description	Note
VTYZ	VTYZ	Cell				Particle		Viscoelastic stress	
VTZX	VTZX	Cell				Particle		Viscoelastic stress	
IGTIME	IGTIME	Cell				Particle		Slow burn ignition time	
SBRCRT	SBRCRT	Cell				Particle		Slow burn reaction ratio	
ABS.VEL	ABSVEL	Node	Node	Cell	Cell	Particle	Node	Absolute velocity	
EDGE.MASS	FMASS	Node				Particle		SPH joined face mass	
JOIN.CELLS	NUMCEL	Node				Particle		SPH joins – number of joined cells	
INITIAL X	XN0	Node	Node			Particle	Node	Original X space co-ordinate	
INITIAL Y	YN0	Node	Node			Particle	Node	Original Y space co-ordinate	
INITIAL Z	ZN0	Node	Node			Particle	Node	Original Z space co-ordinate	
GAS.PRESS	PGAS	Cell		Cell	Cell	Particle		Slow-burn gas pressure	
FILL.DENS	FILDEN	Cell		Cell	Cell	Particle		Slow-burn fill density	
VAR.1.. VAR20	VARxx	X	X	X	X	X	X	User defined variable 1 to 20	
TEMP.1 .. TEMP.31	TEMPxx							Not available	

The index value for a given grid variable is N*name*, where "*name*" is the internal variable name. For example, if you wanted the index for x-velocity(UXN) it would be NUXN. This index could then be used in the various AUTODYN functions (e.g. GV(NUXN,IJK).

Notes

A) ALPHA is a material model dependent variable, which can have the following meanings for the specified material models:

JWL:	Burn fraction	
Porous:	Compaction, defined as current density / solid density	
Tillotson:	Current phase of material)	
Puff:	Current phase of material)	See Theory Manual
Twophase:	Current phase of material)	

C) Damage is not used in standard material models. It will only be non-zero if defined in user subroutines (e.g. EXDAM).

E) An explanation and the derivation of the equations for effective plastic strain, effective plastic strain rate and effective strain are given below:

The plane which makes equal angles with each of the principal directions is called the octahedral plane. The shear stress on this plane is given by:

$$\tau_{\text{oct}} = \sqrt{\frac{2J_2}{3}}$$

where the second invariant of the stress deviators is given by:

$$J_2 = \frac{1}{6}\left[(\sigma_{11} - \sigma_{22})^2 + (\sigma_{22} - \sigma_{33})^2 + (\sigma_{33} - \sigma_{11})^2\right]$$

where σ_{ij} is the total stress tensor in the ij direction. Directions 11, 22 and 33 are the principal stress directions.

The Von Mises yield criterion states that yielding begins when the octahedral shearing reaches a critical value defined by:

$\tau_{\text{oct}} = \sqrt{\frac{2}{3}}k$ where k is the yield stress in pure shear

and the yield criterion is:

$$f(J_2) = J_2 - k^2 = 0$$

Yielding will occur in a uniaxial tension test when:

$$\sigma_1 = \sigma_y, \sigma_2 = \sigma_3 = 0$$

Substituting these values in the above equations gives the uniaxial yield stress as:

$$\sigma_y = \sqrt{3}k = \sqrt{3J_2}$$

In AUTODYN at each cycle the stress state is checked against the yield criterion and if the yield criterion is exceeded an increment of effective plastic strain is computed as follows:

$$\Delta\varepsilon_{eff}^{p} = \frac{\sqrt{3J_2} - \sigma_y}{3G}$$

The **effective plastic strain** is the integrated value of these increments during the calculation:

$$\varepsilon_{eff}^{p} = \int \Delta\varepsilon_{eff}^{p} dt$$

The **effective plastic strain rate** is given by:

$$\dot{\varepsilon}_{eff}^{p} = \frac{\Delta\varepsilon_{eff}^{p}}{\Delta t}$$ where Δt is the current timestep

and the **effective strain** is given by:

$$\varepsilon_{eff} = \int \Delta\varepsilon_{eff}\, dt$$

where $\Delta\varepsilon_{eff} = \Delta t\left[\frac{2}{3}\left(\dot{\varepsilon}_{xx}^{2} + \dot{\varepsilon}_{yy}^{2} + \dot{\varepsilon}_{zz}^{2} + \dot{\varepsilon}_{xy}^{2} + \dot{\varepsilon}_{yz}^{2} + \dot{\varepsilon}_{zx}^{2}\right)\right]^{\frac{1}{2}}$

where ε_{ij} is the total strain tensor in the ij direction which includes elastic and plastic components

F) When using the Euler and Euler Godunov processors a given cell may contain more than one material. In such a case, there is not a single value for such variables as compression and energy. In order to obtain these "multi-material" variables one has to reference the multiple material arrays. For AUTODYN-2D, a mass-weighted value for compression, internal energy, temperature, and alpha is available in the "standard" array locations (e.g. XMU, EN, TEMP, and ALPHA). In AUTODYN-3D, the multi-material access method is used for all processors including Lag/ALE and Shells.

Refer to the User Subroutine tutorial for further details and examples.

G) When using the Lagrange, ALE, Shell, Beam, SPH and FCT solvers, these variables are most efficiently accessed from the multi-material array structure using the direct method: For each cell, set the material variable array pointer using

ML => MTSUB(IJK)%V(1:NUMMLV)

Then access/set the material data using

ML(*index*).

For example, to set the cell damage to one and internal energy to zero use

ML(NCDM) = 1.0
ML(NCEN) = 0.0

Published: 2011-10-05

B.3. Unstructured Solvers

The complete list of unstructured variables for both 2D and 3D can be obtained through the *Output, Save, Review variables* option in the interface. Both the real and integer variables can be viewed by selecting to review all unstructured variables.

Published: 2011-10-05

Appendix C. Subroutine MDSTR_USER_1 Example

The attached listing of the file MDSTR_USER_1.TUT is used for the example User Subroutine Tutorial problem (Ident: USER_STRENGTH_EXAMPLE). List the file included with your distribution for the latest version of this subroutine.

```
! ***********************************************************************

! THIS MODULE IS A CONTAINER FOR THE INITIALISATION AND SOLUTION
! OF A USER STRENGTH MODEL

! THE FOLLOWING ROUTINES ARE INCLUDED:

! MODULE STR_USER_1
!   DEFINE VARIABLES THAT ARE COMMON BETWEEN THE ROUTINES BELOW

! SUBROUTINE INIT_STR_USER_1
!   DEFINE THE INPUT PARAMETERS FOR THE USER STRENGTH MODEL

! SUBROUTINE CHECK_STR_USER_1
!   CHECK PARAMETERS ARE VALID FOR THE USER STRENGTH MODEL

! SUBROUTINE SET_STR_USER_1
!   SET SHORTCUTS TO PARAMETERS FOR THE USER STRENGTH MODEL

! SUBROUTINE SOLVE_STR_USER_1
!   SOLVE THE USER STRENGTH MODEL

! BEFORE EACH ROUTINE IS CALLED, THE FOLLOWING POINTERS ARE SET-UP
!   MTL - POINTER TO THE CURRENT MATERIAL
!   EQ  - POINTER TO THE CURRENT FLAG/EQUATION/MATERIAL OPTION

! ***********************************************************************

MODULE STR_USER_1
USE kindef
IMPLICIT NONE
SAVE

! SPECIFY COMMON VARIABLES TO BE ACCESSED BY ROUITNES BELOW HERE
!INTEGER(INT4) ::
!REAL(REAL8) ::
REAL (REAL8), DIMENSION(3) :: EP, YS

END MODULE STR_USER_1


SUBROUTINE INIT_STR_USER_1(IFACT)
```


Published: 2011-10-05

```
      USE material
      USE str_user_1

      IMPLICIT NONE

      INTEGER (INT4) ::   IFACT

      ! ************************************************************************

      ! THIS SUBROUTINE INITIALISES (ALLOCATES) PARAMETERS AND DATA

      ! FLAG - IMF_STR_USER_1

      ! INPUT - IFACT = 0 JUST GET NAME OF EQUATION AND DEPENDANT FLAGS
      !         IFACT = 1 EQUATION IS ACTIVE HENCE ALLOCATE

      ! ************************************************************************

      ! DEFINE PARAMETERS TO ALLOW ALLOCATION
      EQ%EQTYPE = IMF_STR_USER_1   ! DO NOT MODIFY THIS LINE
      EQ%NAME = 'User Strength #1'
      EQ%NPAR = 7                  ! NUMBER OF REAL INPUT PARAMETER (MINIMUM OF 1)
      EQ%NUMOPT = 0                ! NUMBER OF OPTION LISTS
      EQ%NDEPFLG = 0                  ! NUMBER OF NON-OPTIONAL DEPENDANT (CHILD) FLAGS/MODEL
OPTION

      IF (IFACT==1) THEN
        CALL ALLOC_EQ ! DO NOT MODIFY THIS LINE, ALLOCATES MEMORY

        ! FOR EACH REAL INPUT PARAMETER, ASSIGN DATA
        !                ('name    '     L, T,M,H, val, min,max,default,0,required)
        EQ%PAR(1)=PRMT (1,'Shear Modulus',-1,-2,1,0,ZERO,ZERO,BIG,ZERO    ,0,1)   ! THIS LINE
MUST ALWAYS EXIST
        EQ%PAR(2)=PRMT (2,'EPS #1'       , 0, 0,0,0,ZERO,ZERO,BIG,ZERO   ,0,0)
        EQ%PAR(3)=PRMT (3,'EPS #2'       , 0, 0,0,0,ZERO,ZERO,BIG,ZERO   ,0,0)
        EQ%PAR(4)=PRMT (4,'EPS #3'       , 0, 0,0,0,ZERO,ZERO,BIG,ZERO   ,0,0)
        EQ%PAR(5)=PRMT (5,'YIELD #1'     ,-1,-2,1,0,ZERO,ZERO,BIG,ZERO   ,0,0)
        EQ%PAR(6)=PRMT (6,'YIELD #2'     ,-1,-2,1,0,ZERO,ZERO,BIG,ZERO   ,0,0)
        EQ%PAR(7)=PRMT (7,'YIELD #3'     ,-1,-2,1,0,ZERO,ZERO,BIG,ZERO   ,0,0)

        ! FOR EACH OPTION LIST, ASSIGN DATA
        !   FOR EXAMPLE,
        !   EQ%OPTION(1)%NAME = 'Strain rate dependant'  ! OPTION LIST NAME
        !   EQ%OPTION(1)%NUMOPT = 2          ! NUMBER OF OPTIONS IN THE LIST
        !   EQ%OPTION(1)%DEFAULT = 1         ! DEFAULT OPTION
        !   EQ%OPTION(1)%SELECTED = 1        ! SELECTED OPTION
        !   CALL ALLOC_OPTION(1)             ! ALLOCATE THE MEMORY
        !   DEFINE OPTIONS
        !                                ('name  ',active,' ',0 / Dependant (child) flag)
        !   EQ%OPTION(1)%OPTS(1) = OPTION('Yes','Y',' ',0)
        !   EQ%OPTION(1)%OPTS(2) = OPTION('No','Y',' ',0)

        ! FOR EACH NON-OPTIONAL DEPENDANT (CHILD) FLAG/MODEL OPTION, ASSIGN DEPENDANT FLAG
```

```
  !EQ%DEPFLG(1) = IMF_YP_PCWISE
ENDIF

! SET IN ACTIVE SWITCH FOR APPROPRIATE PROCESSOR TYPE:: ALL ON BY DEFAULT
EQ%IFSOLVER(ISLV_FCT) = 0

RETURN
END SUBROUTINE INIT_STR_USER_1


SUBROUTINE SET_STR_USER_1

USE material
USE str_user_1

IMPLICIT NONE

! **********************************************************************

! THIS SUBROUTINE ASSIGNS SHORTCUTS FOR DIRECT USE IN THE SOLVER

! **********************************************************************

! FOR EXAMPLE
SHRMDZ = EQ%PAR(1)%VAL  ! THIS LINE MUST BE PRESENT
EP(1) = EQ%PAR(2)%VAL
EP(2) = EQ%PAR(3)%VAL
EP(3) = EQ%PAR(4)%VAL
YS(1) = EQ%PAR(5)%VAL
YS(2) = EQ%PAR(6)%VAL
YS(3) = EQ%PAR(7)%VAL

!   ISEL_OPT = EQ%OPTION(1)%SELECTED

RETURN

END SUBROUTINE SET_STR_USER_1

SUBROUTINE CHECK_STR_USER_1

USE material
USE str_user_1

IMPLICIT NONE

INTEGER (INT4) :: IERROR

! **********************************************************************

! THIS SUBROUTINE CHECKS EOS INPUT DATA

! **********************************************************************
```

```
! PLACE USER CHECKS HERE

! CHECK THAT EPS IS MONOTONICALLY INCREASING
IERROR = 0
IF (EP(1)>EP(2).OR.EP(1)>EP(3)) IERROR = 1
IF (EP(2)>EP(3)) IERROR = 1
IF (IERROR==1) THEN
  CALL USR_ERROR (' ERROR  !','USER STRENGTH MODEL. Plastic strain must be monotonically
                    increasing.')
END IF

RETURN

END SUBROUTINE CHECK_STR_USER_1

SUBROUTINE SOLVE_STR_USER_1_2D (PRES,TT1,TT2,TT3,XMUT,EPST,EPSD,TEMPT,DAMAGE,
                                    YIELDT,IFAIL)

USE material
USE str_user_1
USE cycvar
USE edtdef
USE ijknow
USE wrapup
USE mdgrid

IMPLICIT NONE

INTEGER (INT1) ::  IFAIL
INTEGER (INT4) ::    IJK
REAL (REAL8)   ::   EPSD,    EPST,   PRES,   TEMPT,    TT1,    TT2
REAL (REAL8)   ::    TT3,    XMUT, YIELDT, DAMAGE

INTEGER (INT4) ::      I,      IM

! *************************************************************************

! THIS IS A USER SUPPLIED SUBROUTINE WHICH CAN BE USED TO COMPUTE
! THE YIELD STRESS FOR A MATERIAL

! INPUT PARAMETER

! PRES    PRESSURE
! Tnn     PRINCIPAL STRESSES
! XMUT     COMPRESSION
! EPST     EFFECTIVE PLASTIC STRAIN
! EPSD    EFFECTIVE PLASTIC STRAIN RATE
! TEMP    TEMPERATURE
! DAMAGE  DAMAGE
! IFAIL   STRESS STATE INDICATOR
! = 0   HYDRO
! = 1   ELASTIC
! = 2   PLASTIC
```

```
! = 3   BULK FAILURE (WITH HEAL)
! = 4   BULK FAILURE (NO HEAL)

! OUTPUT PARAMETERS

! YIELD     YIELD STRESS FOR CURRENT MATERIAL
! IFAIL     STRESS STATE INDICATOR (SEE ABOVE)

! THE FOLLOWING MODULES CONTAIN INFORMATION WHICH MAY BE
! USEFUL FOR COMPUTING THE OUTPUT PARAMETERS :-

! MODULE  'IJKNOW'

! INOW - I INDEX FOR CURRENT CELL
! JNOW - J INDEX FOR CURRENT CELL
! KNOW - K INDEX FOR CURRENT CELL
! MNOW - CURRENT SUBGRID NUMBER

! MODULE  'MATDEF'
! MATNO         -  THE MATERIAL NUMBER OF THE CURRENT MATERIAL
! MATERIALS(MATNO)%NAME -  THE MATERIAL NAME OF THE CURRENT MATERIAL

! MODULE  'CYCVAR'
! NCYCLE - CURRENT CYCLE NUMBER
! TIME   - CURRENT TIME
! DLTH   - TIME STEP FOR CURRENT CYCLE

! MODULE  'EDTDEF'
! NTCODE - DIMENSIONS: 2 = 2D, 3 = 3D

! EN(IJK)   -  CELL SPECIFIC INTERNAL ENERGY
! DAM(IJK)  -  DAMAGE

! TO OBTAIN THE VALUE OF THE INDEX IJK FOR THE CURRENT CELL, USE
!   IJK = IJKSET(INOW,JNOW,KNOW)
! THE INDEX IJK MUST ALSO BE DEFINED AS AN INTEGER:  -  INTEGER (INT4) ::   IJK

! ***********************************************************************

! SUBROUTINE CALLED BY ALL STRENGTH MODELS SO SKIP OUT, BY DEFAULT
IF (NSTR/=IMF_STR_USER_1) GO TO 900

! TO ACCESS A V4.3 USER SUBROUITNE FOR AUTODYN-2D, UNCOMMENT THE NEXT LINE
!CALL EXYLD (PRES,TT1,TT2,TT3,XMUT,EPST,EPSD,TEMPT,YIELDT,IFAIL)

! THIS ROUTINE IS ONLY WRITTEN FOR TANTALUM.
! CHECK THAT NO OTHER MATERIAL TRIES TO USE THIS ROUTINE.
! (NSWRAP = 99 WRAPS UP THE CALCULATION WITH MESSAGE THAT
! PROBLEM HAS BEEN TERMINATED DUE TO USER DETECTED ERROR)
IF (MTL%NAME(1:8)/='TANTALUM') THEN
  CALL USR_MESSAG ('USER STRENGTH MODEL called for invalid material')
  NSWRAP = 99
  YIELDT = ZERO
```

```
        GO TO 900
      END IF

      ! SET CURRENT YIELD STRESS

      ! PLASTIC STRAIN LESS THAN EP(1)
      IF (EPST<=EP(1)) THEN
        YIELDT = YS(1)

      ! PLASTIC STRAIN GREATER THAN EP(3)
      ELSE IF (EPST>=EP(3)) THEN
        YIELDT = YS(3)

      ! INTERPOLATE YIELD STRESS FROM YP VS. EP CURVE
      ELSE

        DO I = 2,3
          IF (EPST<=EP(I)) THEN
            IM = I-1
            YIELDT = YS(IM) + (YS(I)-YS(IM))*(EPST-EP(IM))/(EP(I)-EP(IM))
            EXIT
          END IF
        END DO

      END IF

  900 RETURN

      END SUBROUTINE SOLVE_STR_USER_1_2D

       SUBROUTINE SOLVE_STR_USER_1_3D
       (PRES,TT1,TT2,TT3,XMUT,EPST,EPSD,TEMPT,DAMAGE,YIELDT,IFAIL)

      USE material
      USE str_user_1
      USE cycvar
      USE edtdef
      USE ijknow
      USE wrapup
      USE mdgrid3

      IMPLICIT NONE

      INTEGER (INT1) ::  IFAIL
      INTEGER (INT4) ::    IJK
      REAL (REAL8)   ::   EPSD,    EPST,   PRES,   TEMPT,    TT1,    TT2
      REAL (REAL8)   ::    TT3,    XMUT, YIELDT, DAMAGE

      ! ************************************************************************

      ! THIS IS A USER SUPPLIED SUBROUTINE WHICH CAN BE USED TO COMPUTE
      ! THE YIELD STRESS FOR A MATERIAL
```


Published: 2011-10-05

```
! INPUT PARAMETER

! PRES    PRESSURE
! Tnn     PRINCIPAL STRESSES
! XMUT     COMPRESSION
! EPST     EFFECTIVE PLASTIC STRAIN
! EPSD    EFFECTIVE PLASTIC STRAIN RATE
! TEMP    TEMPERATURE
! DAMAGE  DAMAGE
! IFAIL   STRESS STATE INDICATOR
! = 0   HYDRO
! = 1   ELASTIC
! = 2   PLASTIC
! = 3   BULK FAILURE (WITH HEAL)
! = 4   BULK FAILURE (NO HEAL)

! OUTPUT PARAMETERS

! YIELD     YIELD STRESS FOR CURRENT MATERIAL
! IFAIL     STRESS STATE INDICATOR (SEE ABOVE)

! THE FOLLOWING MODULES CONTAIN INFORMATION WHICH MAY BE
! USEFUL FOR COMPUTING THE OUTPUT PARAMETERS :-

! MODULE  'IJKNOW'

! INOW - I INDEX FOR CURRENT CELL
! JNOW - J INDEX FOR CURRENT CELL
! KNOW - K INDEX FOR CURRENT CELL
! MNOW - CURRENT SUBGRID NUMBER

! MODULE  'MATDEF'
! MATNO         -  THE MATERIAL NUMBER OF THE CURRENT MATERIAL
! MATERIALS(MATNO)%NAME -  THE MATERIAL NAME OF THE CURRENT MATERIAL

! MODULE  'CYCVAR'
! NCYCLE - CURRENT CYCLE NUMBER
! TIME   - CURRENT TIME
! DLTH   - TIME STEP FOR CURRENT CYCLE

! MODULE  'EDTDEF'
! NTCODE - DIMENSIONS: 2 = 2D, 3 = 3D

! THE FOLLOWING GRID VARIABLES MAY ALSO BE USEFUL :-
! ML(NCEN)   -  CELL SPECIFIC INTERN3AL ENERGY
! XMU (IJK) -  CELL COMPRESSION (RHO/RHOREF-ONE)
! ML(NCDM)  -  DAMAGE

! TO OBTAIN THE VALUE OF THE INDEX IJK FOR THE CURRENT CELL, USE
!   IJK = IJKSET(INOW,JNOW,KNOW)
! THE INDEX IJK MUST ALSO BE DEFINED AS AN INTEGER:  -  INTEGER (INT4) ::   IJK

! *************************************************************************
```

```
     ! SUBROUTINE CALLED BY ALL STRENGTH MODELS SO SKIP OUT, BY DEFAULT
     IF (NSTR/=IMF_STR_USER_1) GO TO 900

     ! TEMPORARY ERROR MESSAGE - REPLACE WITH YOUR OWN CODE
     CALL USR_MESSAG ('User subroutine STR_USER_1_3D missing')
     NSWRAP = 9

     ! TO ACCESS A V4.3 USER SUBROUITNE FOR AUTODYN-3D, UNCOMMENT THE NEXT LINE
     !CALL EXYLD (PRES,TT1,TT2,TT3,XMUT,EPST,EPSD,TEMPT,YIELDT,IFAIL)
     ! ALSO NOTE:: USE mdgrid3 AND NOT mdgrid for AUTODYN-3D


 900 RETURN

     END SUBROUTINE SOLVE_STR_USER_1_3D
```


Published: 2011-10-05

This page left intentionally blank

Published: 2011-10-05

Appendix D. Subroutine EXVEL Example

The attached listing is for a user supplied EXVEL subroutine which is used for two user defined velocity constraints: V-POSITIVE and V-NEGATIVE which are used to constrain the y-component of velocity.

```
SUBROUTINE EXVEL (NAMVEL,RBC,K,XB,YB,UXT,UYT)

      USE kindef
      USE bnddef
      USE ijknow
      USE wrapup

      IMPLICIT NONE

      INTEGER (INT4) ::      K
      REAL (REAL8)   ::    UXT,    UYT,     XB,     YB
      REAL (REAL8), DIMENSION(5) :: RBC
      CHARACTER (LEN=10) :: NAMVEL
!
! ******************************************************************
!
!  THIS IS A USER SUPPLIED SUBROUTINE WHICH APPLIES VELOCITY
!  CONSTRAINTS TO NODES. THE ROUTINE IS CALLED ONCE PER CYCLE FOR
!  EACH NODE ASSIGNED WITH THE USER VELOCITY CONSTRAINT
!  THROUGH INPUT. THE USER VELOCITY CONSTRAINT IS USED WHEN THE
!  X AND Y VELOCITY CONSTRAINTS CANNOT BE DESCRIBED BY A
!  COMBINATION OF THE STANDARD VELOCITY CONSTRAINTS
!  AVAILABLE IN AUTODYN INPUT PARAMETERS

!  INPUT PARAMETERS

!    NAMVEL    BOUNDARY CONDITION NAME
!              (SUPPLIED BY USER DURING INPUT)
!    RBC(1-5)  INPUT PARAMETERS FOR BOUNDARY CONDITION
!    K    = 0, REGULAR NODE, (I,J) FOR NODE IS DEFINED BY 'INOW'
!              AND 'JNOW' IN MODULE 'IJKNOW'
!         < 0, INTERACTIVE NODE, K IS THE INTERACTIVE NODE #
!    XB        X-COORDINATE OF NODE AT BEGINNING OF CYCLE
!    YB        Y-COORDINATE OF NODE AT BEGINNING OF CYCLE
!    UXT       TENTATIVE X-VELOCITY BEFORE CONSTRAINTS
!    UYT       TENTATIVE Y-VELOCITY BEFORE CONSTRAINTS

!  OUTPUT PARAMETERS
!
!    UXT   X-VELOCITY AFTER CONSTRAINT HAS BEEN APPLIED
!    UYT   Y-VELOCITY AFTER CONSTRAINT HAS BEEN APPLIED
!   NOTE: AFTER RETURNING FROM THIS ROUTINE,
!     END OF CYCLE COORDINATES WILL BE COMPUTED USING:-
!     XE = XB + UXT*DLTH    YE = YB + UYT*DLTH
```


Published: 2011-10-05

```
!
!   SO YOU CAN CONSTRAIN THE DOMAIN OF X AND Y BY DIRECTLY
!   MODIFYING UXT AND UYT TO ENSURE THAT XE AND/OR YE REMAIN
!   WITHIN CERTAIN BOUNDS


!  THE FOLLOWING MODULES CONTAIN INFORMATION WHICH MAY
!  BE USEFUL FOR COMPUTING THE VELOCITY CONSTRAINTS :-

!       MODULE'IJKNOW'

!              INOW - I INDEX FOR CURRENT CELL
!              JNOW - J INDEX FOR CURRENT CELL
!              MNOW - CURRENT PART NUMBER

!       MODULE'BNDDEF'

!              LIMBDY - Limit on number of boundary conditions
!              LIMBDC - Limit on number of parameters stored for each boundary condition
!              NUMBDY - Number of boundary conditions

!       MODULE'KINDEF'

!              PI     = 3.1415927
!              THIRD  = 1.0/3.0
!              SMALL  = 1.0E-20
!              ZERO   = 0.0
!              ONE    = 1.0

! *****************************************************************
!
       CHARACTER*40 TEXT40
!
       IF (NAMVEL.EQ.'V-POSITIVE') THEN
                      UYT = MAX (UYT,0.0)
       ELSE IF (NAMVEL.EQ.'V-NEGATIVE') THEN
                      UYT = MIN (UYT,0.0)
               ELSE
                      WRITE (TEXT40,'(3A)') '$Error EXVEL is called as :',NAMVEL,'$'
                      CALL USR_MESSAG (TEXT40)
                      NSWRAP = 99
        ENDIF

!  TERMINATION OF SUBROUTINE EXVEL
       RETURN
       END SUBROUTINE EXVEL
```


Published: 2011-10-05

Appendix E. Subroutine EXALE Example

The attached listing of the file EXALE.F90 is used to define the ALE motion of a part whereby each I-column remains parallel to the Y-axis and the nodes are equally spaced along the column.

```
SUBROUTINE EXALE (IREZ,JREZ,NREZ,XREZ,YREZ)

       USE mdgrid
       USE kindef
       USE locsub
       USE wrapup

       IMPLICIT NONE

       INTEGER (INT4) ::   IREZ,   JREZ,   NREZ
       REAL (REAL8)   ::   XREZ,   YREZ

       INTEGER (INT4) ::  IJMAX,     IJ
       REAL (REAL8)   ::    DYB

!      ******************************************************************
!
!      THIS SUBROUTINE COMPUTES THE CONSTRAINED GRID VELOCITIES FOR
!      VERTICES ASSIGNED THE "USER" MOTION CONSTRAINT
!
!      INPUT PARAMETERS:
!
!         IREZ - I INDEX FOR VERTEX TO BE CONSTRAINED
!         JREZ - J INDEX FOR VERTEX TO BE CONSTRAINED
!         NREZ - PART # FOR VERTEX TO BE CONSTRAINED
!         XREZ - CURRENT X-COORDINATE OF VERTEX
!         YREZ - CURRENT Y-COORDINATE OF VERTEX
!
!      OUTPUT PARAMETERS:
!
!         XREZ - CONSTRAINED X-COORDINATE OF VERTEX
!         YREZ - CONSTRAINED Y-COORDINATE OF VERTEX
!
!      ******************************************************************
!
!      THIS LOGIC SETS THE X-COORDINATE OF EACH VERTEX (I,J) TO THE
!      X-VALUE AT VERTEX (I,JMAX) AND EQUALLY SPACES THE Y-COORDINATES
!      BETWEEN (I,1) AND (I,JMAX)
!
       IJMAX = IJSET(IREZ,JMAX)
       IJ    = IJSET(IREZ,JREZ)
       XREZ  = XN(IJMAX)
       DYB   = YN(IJMAX)/FLOAT(JMAX-1)
       YREZ  = DYB*FLOAT(JREZ-1)
```


Published: 2011-10-05

```
!
!       TERMINATION OF SUBROUTINE EXALE
        RETURN
        END SUBROUTINE EXALE
```

Appendix F. Subroutine EXEDIT - 2D Example

The attached listing of the file EXEDIT.F90 is used store the maximum pressure over time for all cells in the problem. The example is for AUTODYN-2D but 3D would be analogous by simply including logic for the K index.

```
SUBROUTINE EXEDIT

      ! *********************************************************************

      ! THIS IS A USER SUPPLIED SUBROUTINE WHICH CAN BE USED TO PROVIDE
      ! SPECIAL CUSTOM EDITING. THE FREQUENCY AT WHICH THIS SUBROUTINE
      ! IS CALLED IS DEFINED THROUGH INPUT (GLOBAL-EDIT-USER). WHEN
      ! REQUESTED, IT IS CALLED BY THE EDIT PROCESSOR AT THE END OF A
      ! COMPUTATIONAL CYCLE. THE ROUTINE IS CALLED BEFORE ANY OTHER
      ! TYPES OF STANDARD EDITS ARE CALLED FOR THAT CYCLE (EG. PRINT,
      ! SAVE, HISTORY, DISPLAY, ETC), SO IT MAY ALSO BE USED TO SET UP
      ! DATA TO BE PROCESSED BY OTHER EDIT TYPES.

      ! *********************************************************************

      USE mdgrid
      USE ranges
      USE kindef
      USE cycvar
      USE wrapup
      USE subdef

      IMPLICIT NONE

      INTEGER (INT4) ::      I,      J,     NS,    IJK
      REAL (REAL8)  ::       PRESSURE

      ! *********************************************************************

      ! THIS SUBROUTINE ATTAINS MAXIMUM PRESSURE FOR EACH CELL (2D)
         !
         !     VAR01    ARRAY STORING MAXIMUM PRESSURE
         !     VAR02    ARRAY STORING TIME WHEN MAXIMUM PRESSURE OCCURS

      ! *********************************************************************

      ! INITIALIZATION OF ARRAYS
         IF (NCYCLE == 1) THEN
            ! LOOP OVER ALL PARTS
            DO NS = 1, NUMSUB

               NSUB = NS
               CALL GETSUB
               ! LOOP OVER ALL CELLS
```

```
                DO I = 1,IMAX
                        DO J = 1,JMAX
                                IJK             = IJSET(I,J)
                                VAR01(IJK)      = - BIG
                                VAR02(IJK)      = - BIG
                        ! END LOOP ON CELLS
                        END DO
                END DO

        ! END LOOP ON PARTS
        END DO
   END IF

! GET MAXIMUM PRESSURE
! LOOP OVER ALL PARTS
IF (NCYCLE > 1) THEN
        DO NS = 1, NUMSUB
                IF (NS/=NSUB) THEN
                NSUB = NS
                CALL GETSUB
                END IF

                ! LOOP OVER ALL CELLS
                DO I = 1,IMAX
                        DO J = 1,JMAX

                ! GET MAXIMUM PRESSURE FOR EACH CELL
                                IJK = IJSET(I,J)
                                PRESSURE = PN(IJK)
                                IF(PRESSURE >= VAR01(IJK)) THEN
                                        VAR01(IJK) = PRESSURE
                                        VAR02(IJK) = TIME
                                END IF

                !       END LOOP ON CELLS
                        END DO
                END DO

        ! END LOOP ON PARTS
        END DO

   END IF

RETURN

END SUBROUTINE EXEDIT
```

Appendix G. Subroutine EXEDIT3 – 3D Example

The attached listing of the file EXEDIT3.F90 is used store the maximum momentum over time for all cells in the problem, except Euler. The example is for AUTODYN-3D

```
SUBROUTINE EXEDIT3

      ! ************************************************************************

      ! THIS IS A USER SUPPLIED SUBROUTINE WHICH CAN BE USED TO PROVIDE
      ! SPECIAL CUSTOM EDITING. THE FREQUENCY AT WHICH THIS SUBROUTINE
      ! IS CALLED IS DEFINED THROUGH INPUT (GLOBAL-EDIT-USER). WHEN
      ! REQUESTED, IT IS CALLED BY THE EDIT PROCESSOR AT THE END OF A
      ! COMPUTATIONAL CYCLE. THE ROUTINE IS CALLED BEFORE ANY OTHER
      ! TYPES OF STANDARD EDITS ARE CALLED FOR THAT CYCLE (EG. PRINT,
      ! SAVE, HISTORY, DISPLAY, ETC), SO IT MAY ALSO BE USED TO SET UP
      ! DATA TO BE PROCESSED BY OTHER EDIT TYPES.

      ! ************************************************************************

      USE mdgrid3
      USE ranges
      USE kindef
      USE cycvar
      USE wrapup
      USE subdef

      IMPLICIT NONE

      INTEGER (INT4) ::      I,      J,     k,       NS,    IJK
      REAL (REAL8)  ::    MOM, RES_VEL,  MASS

      ! ************************************************************************

      ! THIS SUBROUTINE ATTAINS MAXIMUM MOMENTUM FOR EACH CELL (3D)
         !
         !      VAR01    ARRAY STORING MAXIMUM MOMENTUM
         !      VAR02    ARRAY STORING TIME WHEN MAXIMUM MOMENTUM OCCURS

      ! ************************************************************************


      ! INITIALIZATION OF ARRAYS
         IF (NCYCLE == 1) THEN
               ! LOOP OVER ALL PARTS
               DO NS = 1, NUMSUB
                       NSUB = NS
                       CALL GETSUB3
                       ! LOOP OVER ALL CELLS
                       DO I = 1,IMAX
```

```
                        DO J = 1,JMAX
                                DO K = 1,KMAX
                                        IJK          = IJKSET3(I,J,K)
                                        VAR01(IJK)   = - BIG
                                        VAR02(IJK)   = - BIG
                                ENDDO
                        END DO
                END DO
        END DO ! END LOOP ON PARTS
  END IF

! GET MAXIMUM MOMENTUM
! LOOP OVER ALL PARTS
IF (NCYCLE > 1) THEN
        DO NS = 1, NUMSUB
                NSUB = NS
                CALL GETSUB3
                ! LOOP OVER ALL CELLS
                DO I = 1,IMAX
                        DO J = 1,JMAX
                                DO K = 1,KMAX
                                        ! GET MAXIMUM MOMENTUM FOR EACH CELL
                                        IJK = IJKSET3(I,J,K)

                                        ! SET-UP POINTER TO MULTI-MATERIAL ARRAYS
                                        ML => MTSUB(IJK)%V(1:NUMMLV)
                                        MASS = ML(NCMS)

                                        RES_VEL = SQRT ( UXN(IJK)*UXN(IJK) + &
                                                         UYN(IJK)*UYN(IJK) + &
                                                         UZN(IJK)*UZN(IJK) )

                                        MOM = MASS*RES_VEL
                                        IF(MOM >= VAR01(IJK)) THEN
                                                VAR01(IJK) = MOM
                                                VAR02(IJK) = TIME
                                        END IF
                                END DO
                        END DO
                END DO
        ! END LOOP ON PARTS
        END DO

END IF

RETURN

END SUBROUTINE EXEDIT
```

Appendix H. Unstructured Element Data Access

The following examples demonstrate different methods for obtaining the unique internal index of an unstructured element and subsequent data access/storage. Examples are given for

- Direct access through user element number
- Access to all elements in a Part
- Access to all elements in a Component
- Access to all elements in a Group
- Access to nodal variables for NBS tetrahedral elements

Published: 2011-10-05

H.1. Direct Access through User Element Number

The following codes provides an example of how to directly access/store information for a single entity (node or element) knowing the user number.

```
USE mdstring

IMPLICIT NONE

INTEGER(INT4) :: NEL_USER, NEL, N

NEL = 0
DO N = 1, NUM_ENTITY_ENTRIES(TYPE_ELEM)
  IF (ENTITY_TYPES(TYPE_ELEM)%ID(N)==NEL_USER) THEN
    NEL = N
    EXIT
  END IF
END DO
```

H.2. Access to All Elements in a Part

The following code loops over all Unstructured Parts in a model and defines the value of user variable VAR01 to be the impedance of the material for all Parts containing volume elements.

```
USE mdpart
USE mdvar_all
USE mdsolv

IMPLICIT NONE

INTEGER(INT4) :: NPART, N, NEL, NINST

DO NPART = 1, NUM_PARTS
  PART => PARTS(NPART)%P
  IF (PART%ELEM_CLASS/=ICLASS_VOLUME) CYCLE ! SKIP NON-SOLID ELEMENTS

  ! LOOP OVER ELEMENTS IN PART
  DO N = 1, PART%NUMELM
    NEL = PART%ELEMENT_LIST(N) ! OBTAIN GLOBAL INTERNAL INDEX IF ELEMENT

    ! COPY ELEMENT VARIABLES INTO LOCAL VARIABLE VECTOR
    NINST = 0
    CALL GET_ELEM_VAL(NEL,NINST)

    ! CALCULATE IMPEDANCE
    RVL(IVR_VAR01) = RVL(IVR_DENSITY)*RVL(IVR_SOUNDSPEED)

    ! COPY UPDATED LOCAL ELEMENT DATA BACK TO MAIN STORAGE
    CALL PUT_ELEM_VAL(NEL,NINST)

  END DO ! END LOOP ON ELEMENTS
END DO ! END LOOP ON PARTS
```


Published: 2011-10-05

H.3. Access to All Elements in a Component

The following code loops over all Components in a model and for component 'Solid' defines and stores the impedance of each element in the component in user variable VAR01. The code works for both Structured and Unstructured Parts/Solvers.

```
USE mdpart
USE mdvar_all
USE mdsolv
USE mdcomponents
USE mdgrid3

IMPLICIT NONE

INTEGER(INT4) :: NPART, NEL, NINST, N, NN, NNN
INTEGER(INT4) :: I, J, K, IJK

DO N = 1, NUM_COMPONENTS
  COMP => COMPONENTS(N)%P
  IF (COMP%NAME/='Solid') CYCLE

  DO NN = 1, COMP%NUM_PART
    NPART = COMP%PART_LIST(NN)

    IF (NPART>NUMSUB) THEN
      ! UNSTRUCTURED PART
      NPART = NPART-NUMSUB
      PART => PARTS(NPART)%P

      ! LOOP OVER ELEMENTS IN PART
      DO NNN = 1, PART%NUMELM
        NEL = PART%ELEMENT_LIST(NNN) ! OBTAIN GLOBAL INTERNAL INDEX IF ELEMENT

        ! COPY ELEMENT VARIABLES INTO LOCAL VARIABLE VECTOR
        NINST = 0
        CALL GET_ELEM_VAL(NEL,NINST)

        ! CALCULATE IMPEDANCE
        RVL(IVR_VAR01) = RVL(IVR_DENSITY)*RVL(IVR_SOUNDSPEED)

        ! COPY UPDATED LOCAL ELEMENT DATA BACK TO MAIN STORAGE
        CALL PUT_ELEM_VAL(NEL,NINST)

      END DO ! END LOOP ON ELEMENTS
    ELSE
      ! STRUCTURED (IJK) PART
      NSUB = NPART
      CALL GETSUB3

      ! LOOP OVER ELEMENTS IN PART
      DO I = 1, IMAX
```

```
          DO J = 1, JMAX
            DO K = 1, KMAX
              IJK = IJKSET3(I,J,K)
              IF (ASSOCIATED(VAR01)) THEN
                VAR01(IJK) = DEN(IJK)*SSPD(IJK)
              END IF
            END DO
          END DO
        END DO
      END IF
    END DO ! END LOOP ON PARTS
  END DO ! END LOOP ON COMPONENTS
```

H.4. Access to All Elements in a Group

The following code loops over all Groups in a model and for element Group 'Proj' defines and stores the impedance of each element in the group in user variable VAR01. The code works for both Structured and Unstructured Parts/Solvers.

```
 USE mdvar_all
 USE mdsolv
 USE mdgrid3
 USE mdgroups

 IMPLICIT NONE

 INTEGER(INT4) ::  NEL, NINST, N, NN
 INTEGER(INT4) :: I, J, K, IJK, IJKS, M

 DO N = 1, NUM_GROUPS
   GRP => GROUPS(N)%P
   IF (GRP%NAME/='Proj') CYCLE

   ! SKIP ALL BUT ELEMENT GROUPS (OPTIONS: GRPTYP_NODE, GRPTYP_ELEM, GRPTYP_FACE)
   IF (GRP%GRPTYP/=GRPTYP_ELEM) CYCLE

   ! LOOP OVER ELEMENTS IN GROUP
   DO NN = 1, GRP%GRP_SIZE

     ! CHECK IF ELEMENT IS FROM STRUCTURED OR UNSTRUCTURED SOLVER
     IF (GRP%IJKS_LIST(NN)>IJKBAS(NUMSUB+1)) THEN
       ! GET UNSTRUCTURED ELEMENT INDEX
       NEL = GRP%IJKS_LIST(NN) - IJKBAS(NUMSUB+1)

       ! COPY ELEMENT VARIABLES INTO LOCAL VARIABLE VECTOR
       NINST = 0
       CALL GET_ELEM_VAL(NEL,NINST)

       ! CALCULATE IMPEDANCE
       RVL(IVR_VAR01) = RVL(IVR_DENSITY)*RVL(IVR_SOUNDSPEED)

       ! COPY UPDATED LOCAL ELEMENT DATA BACK TO MAIN STORAGE
       CALL PUT_ELEM_VAL(NEL,NINST)

     ELSE
       ! GET STRUCTURED IJK INDEX
       IJKS = GRP%IJKS_LIST(NN)
       CALL IJANDKS3 (IJKS,I,J,K,M)
       NSUB = M
       CALL GETSUB3
       IJK = IJKSET3(I,J,K)
       IF (ASSOCIATED(VAR01)) THEN
         VAR01(IJK) = DEN(IJK)*SSPD(IJK)
       END IF

     END IF
   END DO ! END LOOP ON ELEMENTS
END DO ! END LOOP ON GROUPS
```

H.5. Access to nodal variables for NBS tetrahedral elements

The following is an example of how erosion of NBS tetrahedral elements based on the effective plastic strain at the nodes might be achieved. This serves as an example of how to obtain data from unstructured nodes and elements, The example can be found in your distribution in file MDERO_USER_1.TUT

```
! ************************************************************************

! THIS MODULE IS A CONTAINER FOR THE INITIALISATION AND SOLUTION
! OF A MATERIAL MODELLING EQUATION/OPTION

! THE FOLLOWING ROUTINES ARE INCLUDED:

! MODULE ERO_USER_1
!   DEFINE COMMON PARAMETERS TO BE ACCESSED IN ROUTINES BELOW

! SUBROUTINE INIT_ERO_USER_1
!   ALLOCATE SPACE AND DEFINE THE PARAMETERS FOR A GIVEN FLAG

! SUBROUTINE CHECK_ERO_USER_1
!   CHECK PARAMETERS ARE VALID FOR flagname

! SUBROUTINE SET_ERO_USER_1 (optional)
!   SET PARAMETERS FOR SUBSEQUENT USE IN THE SOLVER

! SUBROUTINE SOLVE_ERO_USER_1_2D
! SUBROUTINE SOLVE_ERO_USER_1_3D
!   SOLVE EQUATION (CALLED FROM SOLVER)

! BEFORE EACH ROUTINE IS CALLED, THE FOLLOWING POINTERS ARE SET-UP
!   MTL - POINTER TO THE CURRENT MATERIAL
!   EQ  - POINTER TO THE CURRENT FLAG/EQUATION/MATERIAL OPTION

! ************************************************************************

MODULE ERO_USER_1
USE kindef
IMPLICIT NONE
SAVE

! SPECIFY COMMON VARIABLES TO BE ACCESSED BY ROUITNES BELOW HERE
!INTEGER(INT4) ::
REAL(REAL8) :: MAX_EPS

END MODULE ERO_USER_1

SUBROUTINE INIT_ERO_USER_1(IFACT)

USE material
USE ero_user_1

IMPLICIT NONE

INTEGER (INT4) ::   IFACT

! ************************************************************************

! THIS SUBROUTINE INITIALISES (ALLOCATES) PARAMETERS AND DATA

! FLAG - IMF_ERO_USER_1

! INPUT - IFACT = 0 JUST GET NAME OF EQUATION AND DEPENDANT FLAGS
!         IFACT = 1 EQUATION IS ACTIVE HENCE ALLOCATE
```

```
! *************************************************************************

! DEFINE PARAMETERS TO ALLOW ALLOCATION
EQ%EQTYPE = IMF_ERO_USER_1
EQ%NAME = 'User Erosion #1'
EQ%NPAR = 1
EQ%NUMOPT = 0
EQ%NDEPFLG = 0
EQ%NCHAR = 0
EQ%NPAR_VEC = 0

IF (IFACT==1) THEN
  ! ALLOCATE ARRAYS FOR EQUATION/FLAG
  CALL ALLOC_EQ

  ! SET PARAMETER NAMES
  EQ%PAR(1)=PRMT(1,'Erosion Plastic Strain',0,0,0,0,BIG,SMALL,BIG,BIG,0,1)
ENDIF

! SET IN ACTIVE SWITCH FOR APPROPRIATE PROCESSOR TYPE:: ALL ON BY DEFAULT
EQ%IFSOLVER(ISLV_FCT) = 0
EQ%IFSOLVER(ISLV_EULER) = 0
EQ%IFSOLVER(ISLV_EULER_GOD) = 0

RETURN
END SUBROUTINE INIT_ERO_USER_1


SUBROUTINE SET_ERO_USER_1

USE material
USE ero_user_1
USE mdvar_all

IMPLICIT NONE

! *************************************************************************

! THIS SUBROUTINE ASSIGNS EOS CONSTANTS FOR DIRECT USE IN THE SOLVER

! *************************************************************************

EROMOD = 5 ! DO NOT MODIFY THIS LINE

MAX_EPS = EQ%PAR(1)%VAL

IF_IVAR_ALL(IVI_EROSION) = 2

RETURN

END SUBROUTINE SET_ERO_USER_1

SUBROUTINE CHECK_ERO_USER_1

USE material
USE ero_user_1

IMPLICIT NONE

! *************************************************************************

! THIS SUBROUTINE CHECKS EOS INPUT DATA

! *************************************************************************

! NO CHECKS REQUIRED

RETURN

END SUBROUTINE CHECK_ERO_USER_1
```

```
SUBROUTINE SOLVE_ERO_USER_1_2D (ISTAT)

USE material
USE ero_user_1
USE mdgrid
USE wrapup

IMPLICIT NONE

INTEGER (INT4) ::  ISTAT

! ***********************************************************************

! THIS IS A USER SUPPLIED SUBROUTINE WHICH CAN BE USED TO ERODE THE
! CURRENT CELL ACCORDING TO ANY CRITERIA THE USER DECIDES.

! OUTPUT PARAMETER

!    ISTAT   EROSION SWITCH - ASSIGN TO NON-ZERO TO ERODE THE CURRENT CELL

! IN ADDITION TO THE FORMAL PARAMETERS, MODULE "MATDEF"
! CONTAINS THE FOLLOWING INFORMATION

!    MATNO          THE MATERIAL NUMBER OF THE MATERIAL BEING PROCESSED
!    MTL%NAME       THE MATERIAL NAME OF THE MATERIAL BEING PROCESSED

! ***********************************************************************

! TEMPORARY ERROR MESSAGE - REPLACE NEXT TWO LINES WITH YOUR OWN CODE
CALL USR_MESSAG ('User subroutine SOLVE_ERO_USER_1_2D missing')
NSWRAP = 9

! UNCOMMENT THE NEXT LINE TO USE OLD V4.3 USER SUBROUITNE
! CALL EXEROD2 (ISTAT)

RETURN

END SUBROUTINE SOLVE_ERO_USER_1_2D

SUBROUTINE SOLVE_ERO_USER_1_3D (ISTAT)

USE material
USE ero_user_1
USE mdgrid3
USE wrapup
USE mdvar_all
USE mdstring
USE mdsolv
USE cycvar

IMPLICIT NONE

INTEGER (INT4) ::  ISTAT, N, ELTYPE, NBS_TET_HGMODEL
INTEGER (INT4) :: LOCMAT
INTEGER (INT4) :: IDNODEEPS
INTEGER (INT4), DIMENSION(4) :: NODENM, MATLOCL

REAL (REAL8) ::   PUSOELEM_EPS, ELEM_EPS, TET_HG_COEFF
REAL (REAL8), DIMENSION(4) :: NODEEPS

! ***********************************************************************

! THIS IS A USER SUPPLIED SUBROUTINE WHICH CAN BE USED TO ERODE THE
! CURRENT CELL ACCORDING TO ANY CRITERIA THE USER DECIDES.

! OUTPUT PARAMETER

!    ISTAT   EROSION SWITCH - ASSIGN TO NON-ZERO TO ERODE THE CURRENT CELL
```


Published: 2011-10-05

```
! IN ADDITION TO THE FORMAL PARAMETERS, MODULE "MATDEF"
! CONTAINS THE FOLLOWING INFORMATION

!    MATNO          THE MATERIAL NUMBER OF THE MATERIAL BEING PROCESSED
!    MTL%NAME       THE MATERIAL NAME OF THE MATERIAL BEING PROCESSED

! AN EXAMPLE OF USING USER EROSION WITH NBS TETRAHEDRA CAN BE FOUND IN
! THE APPENDIX OF THE USER SUBROUTINE MANUAL

! **********************************************************************

! THE FOLLOWING ERODES NBS ELEMENTS BASED ON PLASTIC STRAIN - IT WILL DO
! NOTHING FOR OTHER ELEMENT TYPES

ISTAT = 0

IF (ELEM_NOW == 0) GO TO 100

CALL GET_ELEM_VAR(ELEM_NOW,0)
! ASSESS WHETHER NBS ELEMENT OR NOT AND IF NOT RETURN
ELTYPE = DATA_STR(NSTRING)%P%OPT(EL_SOPT_ELEMTYPE)

IF (ELTYPE /= ELTYPE_TET4_ANS) GO TO 100

! DETERMINE IF PUSO STABILISATION IS USED AND THE PUSO COEFFICIENT IF NECESSARY
NBS_TET_HGMODEL     = DATA_STR(NSTRING)%P%OPT(EL_SOPT_NBS_HGMODEL)
TET_HG_COEFF        = DATA_STR(NSTRING)%P%RPARAM(RPAR_NBS_HGCOEFF)

! GET EFFECTIVE PLASTIC STRAIN IN THE PUSO MATERIAL
PUSOELEM_EPS = RVL(IVR_NBS_EPS)

! GET GLOBAL NODE NUMBERS CONNECTED TO ELEMENT
NODENM(1) = IVL(IVI_CON1)
NODENM(2) = IVL(IVI_CON2)
NODENM(3) = IVL(IVI_CON3)
NODENM(4) = IVL(IVI_CON4)
! NBS NODES HAVE A LAYERED STORAGE STRUCTURE FOR VARIABLES DUE TO MULTI-
! MATERIALS ON THE NODES. THE FOLLOWING GETS THE LAYER NUMBER OF THE
! CURRENT ELEMENT MATERIAL FOR EACH NODE
MATLOCL(1) = IVL(IVI_NBS_MATLOC1)
MATLOCL(2) = IVL(IVI_NBS_MATLOC2)
MATLOCL(3) = IVL(IVI_NBS_MATLOC3)
MATLOCL(4) = IVL(IVI_NBS_MATLOC4)

! GET THE PLASTIC STRAIN AT THE NODES
DO N = 1,4
  ! CALL TO GET_NODE_VAR(NODENO,MATLOC) RETRIEVES NODAL DATA FOR FOR NODE
  ! WITH GLOBAL NODE NUMBER, NODENO, AND LAYER NUMBER, MATLOC
  CALL GET_NODE_VAR(NODENM(N),MATLOCL(N))
  ! RETRIEVE EFFECTIVE PLASTIC STRAIN FOR NODE N
  NODEEPS(N) = RVL(IVR_NBS_EPS)
END DO

! SET ELEMENT TO FAILED IF PLASTIC STRAIN EXCEEDED
ELEM_EPS = MINVAL(NODEEPS)
ELEM_EPS = MAX(ELEM_EPS,PUSOELEM_EPS)
IF (ELEM_EPS >= MAX_EPS) ISTAT = 1

100   CONTINUE

RETURN

END SUBROUTINE SOLVE_ERO_USER_1_3D
```