DEFORM™

Design Environment for FORMing

DEFORM™ 3D Version 6.1 (sp1)
User's Manual

Oct 10th 2007

Scientific Forming Technologies Corporation

2545 Farmers Drive, Suite 200
Columbus, Ohio, 43235
Tel (614) 451-8330
Fax (614) 451-8325
Email support@deform.com

Table of Contents

PREFACE TO THIS MANUAL 6

Chapter 1. Overview of DEFORM 7

1.1 DEFORM family of products 7
1.2 Capabilities 8
1.3. Analyzing manufacturing processes with DEFORM 11
1.4. Before you begin 11
1.5. Geometry representation. 12
1.6. The DEFORM system 13
1.7. Pre-processing 14
1.8. Creating input data 14
1.9. File system 16
1.10. Running the simulation 17
1.11. Post-processor 17
1.12. Units 18

Chapter 2. Pre-Processor 19

2.1. Simulation Controls 19
2.1.1. Main controls 20
2.1.2. Step Controls 23
2.1.3. Advanced Step Controls 26
2.1.4. Stopping Controls 29
2.1.5. Remesh Criteria 31
2.1.6. Iteration Controls 31
2.1.7. Processing Conditions 37
2.1.8. Advanced Controls 39
2.1.9. Control Files 43
2.2 Material Data 46
2.2.1. Phases and mixtures 47
2.2.2. Elastic data 48
2.2.3. Thermal data 51
2.2.4. Plastic Data 52
2.2.5. Diffusion data 61
2.2.6. Hardness data [MIC] 63
2.2.7. Grain growth/recrystallization model 64
2.2.8. Advanced material properties 70
2.2.9. Material data requirements 71
2.3. Inter Material Data 73
2.3.1. Transformation relation (PHASTF) 73
2.3.2. Kinetics model (TTTD) 74
2.3.3. Latent heat (PHASLH). 79
2.3.4. Transformation induced volume change (PHASVL) 79
2.3.5. Transformation plasticity (TRNSFP) 81
2.3.6. Other Transformation Data 81

2.4 Object Definition. 82
2.4.1. Adding, deleting objects 83
2.4.2. Object name (OBJNAM) 84
2.4.3. Primary Die (PDIE) 85
2.4.4. Object type (OBJTYP) 85
2.4.5. Object geometry 87
2.4.6. Object meshing 95
2.4.7. Object material 107
2.4.8. Object initial conditions 107
2.4.9. Object properties 108
2.4.10. Object boundary conditions 115
2.4.11. Contact boundary conditions 118
2.4.12. Object movement controls 118
2.4.13. Object node variables. 131
2.4.14. Object element variables 138
2.4.16. Data Interpolation 145
2.5.1. Inter object Interface 149
2.5.2. Positioning 162
2.5.3. Inter object boundary conditions 164
2.6. Database Generation 165
Chapter 3. Running Simulations 167
3.1. Simulation Options 168
3.2. Switching between Solvers (Conjugate-Gradient and Sparse) 168
3.3. Multi Processing 169
3.3. Email the Result 170
3.4. Starting the simulation 170
3.5. Simulation graphics 171
3.6. Add to Queue (Batch Queue) 172
3.7 Process Monitor 173
3.8. Stopping a simulation 174
3.9. Troubleshooting problems 174
3.9.1. Message file messages 174
3.9.2. Simulation aborted by user 174
3.9.3. Cannot remesh at a negative step 175
3.9.4. Remeshing is highly recommended 175
3.9.5Negative Jacobian 175
3.9.6. Solution does not converge 176
3.9.7. Stiffness matrix is non-positive definite 179
3.9.8. Zero pivot 179
3.9.9. Extrapolation of data 179
3.9.10. Bad Element Shape 180
3.9.11. Inconsistent Step Number 181
Chapter 4: Post-Processor 182
4.1.Post-Processor Overview 183
4.2 Graphical display 184
4.2.1. Window layout 184

4.3.Post-Processing Summary 193
4.3.1.Simulation Summary 193
4.3.2.State Variable 195
Displacement 200
Density 200
Strain 200
Velocity 202
Normal Pressure 202
Temperature 202
Volume Fraction 202
Grain Size 203
Hardness 203
Dominant atom 203
User Variables 203
4.3.3.Point tracking 203
4.3.4.Load stroke curves 205
4.3.5.Coordinate Systems 206
4.3.6. Step Selection & Manipulation 207
4.3.7. Steps list 208
4.3.8.View Changes Within Viewport 210
4.3.9. Coordinate System Selection 210
4.3.10.Rotation 211
4.3.11.Coordinate Axis View 211
4.3.12.Point Selection 211
4.3.13 Multiple Viewports 212
4.3.14. Nodes 212
4.3.15. Elements 213
4.3.16. Viewport 215
4.3.17. Data Extraction 217
4.3.18.Flownet 218
4.3.19. Mirroring 222
4.3.20 Animation controls and saving. 223
Chapter 5: Elementary Concepts in Metalforming and Finite Element Analysis 225
Chapter 6: User Routines 237
User-Defined FEM Routines 237
User-Defined Post-Processing Routines 241
6.1. User defined FEM routines 241
6.2. User defined post-processing routines 267
Quick Reference 273
Hot Forming 276
Appendix A: Running DEFORM in text mode 283
Appendix B: Inserting DEFORM™ Animations in Powerpoint Presentations 287
Appendix C: DETAILS OF MOVEMENT CONTROLS IN SPIN.KEY 289
Appendix D: Data Files 291
Appendix E: 2D to 3D Conversion Utility 293
Appendix F: Fracture with Element Deletion and Damage Softening 295

Appendix G: Rotating Work piece Simulations 300
Appendix H: Sheet Forming in DEFORM-3D 308
Appendix I: Eulerian treatment of the 3D rolling process 317
Appendix J: Preventing leakage of nodes in sectioned simulations 318
Appendix K: The Double Concave Corner Constraint 321
Appendix L: Rolling Simulation Overview (In Progress) 324
Appendix M: Checking the forming loads results of a simulation 326
Appendix N: Model setup for steady state machining 328
Appendix O: Document on constructing linear friction simulations 336
Appendix P: On Using Spring-Loaded Dies 345
Appendix Q: THE DEFORM ELASTO-PLASTIC MODEL 347
Appendix R: Setting Up Multiple Processor Simulations 353
Appendix S: Coupled Die Stress Analysis 356
Appendix T: Setting up steady state extrusion 357
Appendix U: Setting up 3D machining models 365

Preface to this manual

This manual describes the features and capabilities of the DEFORM-3D system. It also contains a description of the inputs and actions required to setup problems and run simulations. If you have not used DEFORM before we would recommend that you go through the lab manuals first for an introduction on how to use the system and how to run different types of simulations. The labs for DEFORM-3D, DEFORM-HT are provided as PDF (Portable document format) documents which can be viewed using Adobe Acrobat provided with DEFORM. All keywords which are used in DEFORM-3D are documented in the keyword reference manuals which are also provided as a PDF document. All documents can be accessed from the help menus in the main program, pre-processor, and post-processor.

Overview of DEFORM

Presents an overview of the DEFORM family of products.

Analyzing manufacturing processes with DEFORM

Describes how to use DEFORM products to analyze manufacturing processes.

The DEFORM system

Introduces the DEFORM-3D system and describes the components that make up the system.

Pre-Processor

Describes the layout of the DEFORM Pre-Processor.

Running Simulations

Describes how to run simulations and also how to handle errors that occur during simulations.

Post-Processor

Describes post-processing results from simulations and how to interpret results.

User Routines

Describes user FORTRAN routines in detail. DEFORM allows the user to write FORTRAN programs to describe the flow stress, die speeds, damage accumulation, and other features, as well as defining and storing new variables which can be tracked in the post-processor along with the standard DEFORM variables.

Release Notes

Contains release notes.

Chapter 1. Overview of DEFORM

DEFORM is a Finite Element Method (FEM) based process simulation system designed to analyze various forming and heat treatment processes used by metal forming and related industries. By simulating manufacturing processes on a computer, this advanced tool allows designers and engineers to:

- Reduce the need for costly shop floor trials and redesign of tooling and processes.
- Improve tool and die design to reduce production and material costs.
- Shorten lead time in bringing a new product to market.

Unlike general purpose FEM codes, DEFORM is tailored for deformation modeling. A user friendly graphical user interface provides easy data preparation and analysis so engineers can focus on forming, not on learning a cumbersome computer system. A key component of this is a fully automatic, optimized remeshing system tailored for large deformation problems.
DEFORM-HT adds the capability of modeling heat treatment processes, including normalizing, annealing, quenching, tempering, aging, and carburizing. DEFORM-HT can predict hardness, residual stresses, quench deformation, and other mechanical and material characteristics important to those that heat treat.

1.1 DEFORM family of products

DEFORM-2D (2D)
Available on UNIX/LINUX platforms (HP, DEC, LINUX) as well as personal computers running Windows-XP/Vista. Capable of modeling plane strain or axisymmetric parts with a simple 2 dimensional model. A full function package containing the latest innovations in Finite Element Modeling, equally well suited for production or research environments.

DEFORM-3D (3D)
Available on UNIX/LINUX platforms (HP, DEC, LINUX) as well as personal computers running Windows-XP/Vista.DEFORM-3D is capable of modeling complex three dimensional material flow patterns. Ideal for parts which cannot be simplified to a two dimensional model.

DEFORM-F2 (2D)
Available on personal computers running Windows XP/Vista. Capable of modeling-two dimensional axisymmetric or plane strain problems. Suitable for small to mid-sized shops starting in Finite Element Modeling.

DEFORM-F3 (3D)
Available on personal computers running Windows XP/Vista. A powerful three-dimensional modeling package for modeling cold, warm and hot forging processes.

DEFORM-HT
Available as an add-on to DEFORM-2D and DEFORM-3D. In addition to the deformation modeling capabilities, DEFORM-HT can model the effects of heat treating, including hardness, volume fraction of metallic structure, distortion, residual stress, and carbon content.

1.2 Capabilities

Deformation

- Coupled modeling of deformation and heat transfer for simulation of cold, warm, or hot forging processes (all products).
- Extensive material database for many common alloys including steels, aluminums, titaniums, and super-alloys (all products).
- User defined material data input for any material not included in the material database (all products).
- Information on material flow, die fill, forging load, die stress, grain flow, defect formation and ductile fracture (all products).
- Rigid, elastic, and thermo-viscoplastic material models, which are ideally suited for large deformation modeling (all products).
- Elastic-plastic material model for residual stress and spring back problems. (2D, 3D).
- Porous material model for modeling forming of powder metallurgy products (2D, 3D).
- Integrated forming equipment models for hydraulic presses, hammers, screw presses, and mechanical presses (all products).
- User defined subroutines for material modeling, press modeling, fracture criteria and other functions (2D, 3D).
- FLOWNET (2D, PC,) and point tracking (all products) for important material flow information.
- Contour plots of temperature, strain, stress, damage, and other key variables simplify post processing (all products).
- Self contact boundary condition with robust remeshing allows a simulation to continue to completion even after a lap or fold has formed (2D, 3D).
- Multiple deforming body capability allows for analysis of multiple deforming work pieces or coupled die stress analysis. (2D, 3D).
- Fracture initiation and crack propagation models based on well known damage factors allow modeling of shearing, blanking, piercing, and machining (2D, 3D).

Heat Treatment

- Simulate normalizing, annealing, quenching, tempering, and carburizing.

  Normalizing (not available yet)
  Heating a ferrous alloy to a suitable temperature above the transformation range and cooling in air to a temperature substantially below the transformation range.
  Annealing
  A generic term denoting a treatment, consisting of heating to and holding at a suitable temperature followed by cooling at a suitable rate, used primarily to soften metallic materials. In ferrous alloys, annealing usually is done above the upper critical temperature, but the time-temperature cycles vary both widely in both maximum temperature attained and in cooling rate employed.
  Tempering (not available yet)
  Reheating hardened steel or hardened cast iron to some temperature below the eutectoid temperature for the purpose of decreasing hardness and increasing toughness.
  Stress relieving
  Heating to a suitable temperature, holding long enough to reduce residual stresses, and then cooling slowly enough to minimize the development of new residual stresses.
  Quenching
  A rapid cooling whose purpose is for the control of microstructure and phase products.
- Predict hardness, volume fraction metallic structure, distortion, and carbon content.
- Specialized material models for creep, phase transformation, hardness and diffusion.
- Jominy data can be input to predict hardness distribution of the final product.
- Modeling of multiple material phases, each with its own elastic, plastic, thermal, and hardness properties. Resultant mixture material properties depend upon the percentage of each phase present at any step in the heat treatment simulation.

DEFORM models a complex interaction between deformation, temperature, and, in the case of heat treatment, transformation and diffusion. There is coupling between all phenomenon, as illustrated in the figure below. When appropriate modules are licensed and activated, these coupling effects include heating due to deformation work, thermal softening, and temperature controlled transformation, latent heat of transformation, transformation plasticity, transformation strains, stress effects on transformation, and carbon content effects on all material properties.

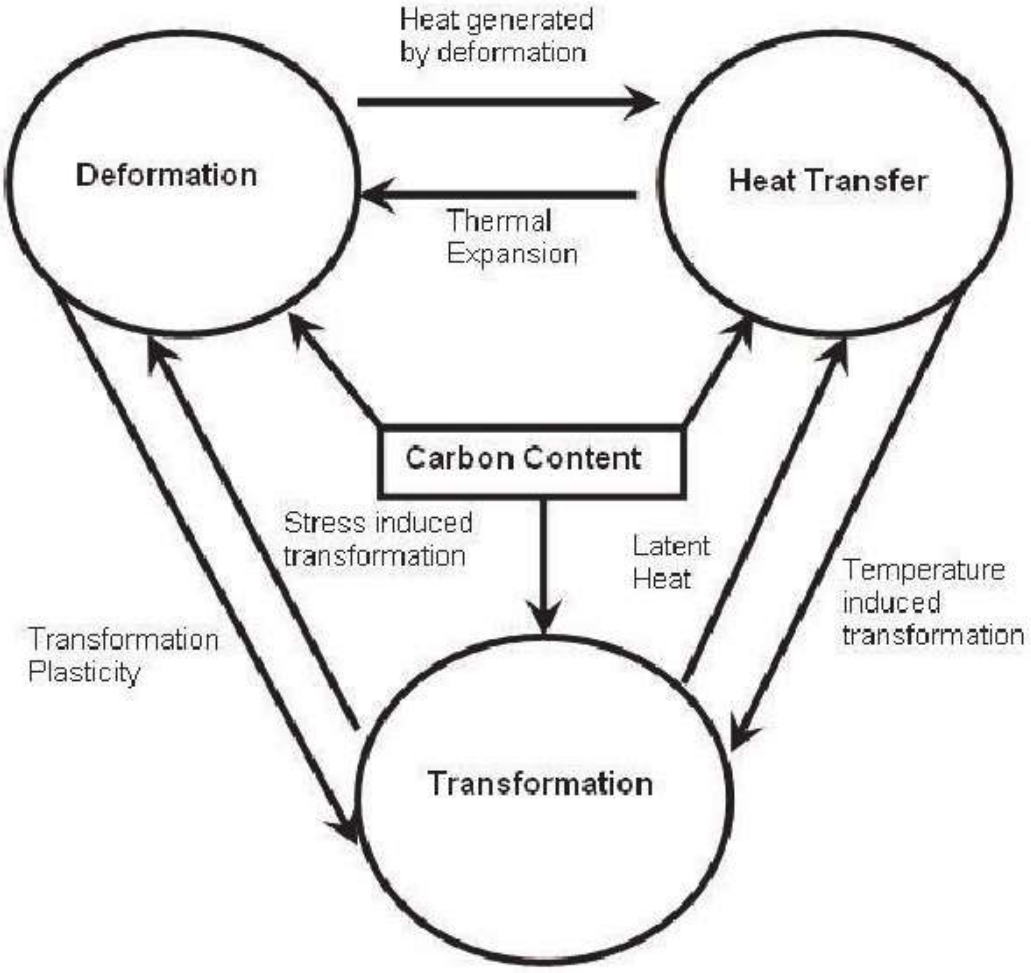


Figure 1.2.1 : Relationship between various DEFORM modules.

1.3. Analyzing manufacturing processes with DEFORM

DEFORM can be used to analyze most thermo-mechanical forming processes, and many heat treatment processes. The general approach is to define the geometry and material of the initial work piece in DEFORM, then sequentially simulate each process that is to be applied to the work piece.
The recommended sequence for designing a manufacturing process using DEFORM

- Define your proposed process
- Final forged part geometry
- Material
- Tool progressions
- Starting work piece/billet geometry
- Processing temperatures, reheats, etc.
- Gather required data
- Material data
- Processing condition data
- Using the DEFORM pre-processor, input the problem definition for the first operation
- Submit the data for simulation
- Using the DEFORM post-processor, review the results
- Repeat the preprocess-simulate-review sequence for each operation in the process
- If the results are unacceptable, use your engineering experience and judgment to modify the process and repeat the simulation sequence.

1.4. Before you begin

Before you begin work on your DEFORM simulation, spend some time planning the simulation. Consider the type of information you hope to gain from the analysis. Are temperatures important? What about die fill? Press loads? Material deformation patterns? Ductile fracture of the part? Die failure? Buckling? Can the part be modeled as a two dimensional part, or is a three dimensional simulation necessary? Having a definite goal will help you design a simulation which will provide the information most vital to understanding your manufacturing process.

1.5. Geometry representation

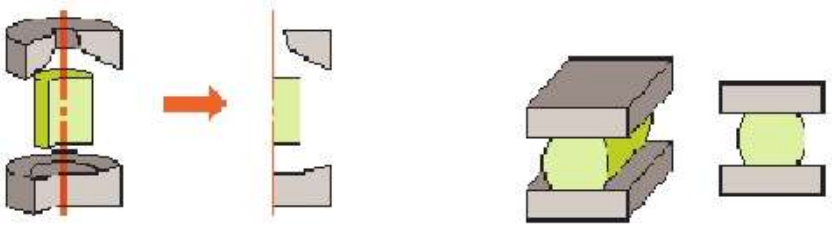


Figure 1.5.1 : Axisymmetric and plane strain examples.

DEFORM simulations can be run either as two dimensional (2D) or three dimensional (3D) models. In general, 2D models are smaller, easier to set up, and run more quickly than 3D models. Frequently, the added detail of a 3D model is not worth the additional time required over a 2D simulation if the process can reasonably be represented in 2D.
There are two 2D geometry representations: axisymmetric and plane strain. Axisymmetric geometries assume that the geometry of every plane radiating out from the centerline is identical. Plane strain requires that there is no material flow in the out of plane direction, and that flow in every plane parallel to the section modeled is identical. Figure 1.5.1 illustrates axisymmetric and plane strain models.
Objects that are closely approximated by axisymmetric or plane strain models can also be modeled in 2D by neglecting minor variations. For example, if the head shape is not critical a hex head bolt can be modeled as axisymmetric by defining a head radius which maintains constant volume (radius = 0.525*(distance across flats)). A gradually tapering part such as a turbine blade can be modeled by modeling several plane strain sections.

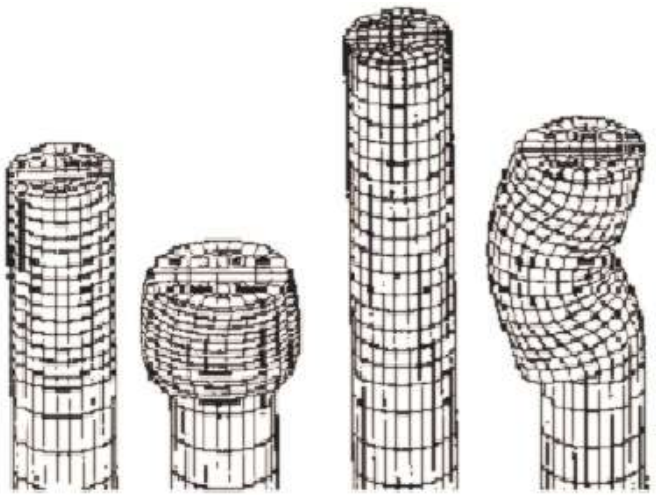


Figure 1.5.2 : Buckling.

Buckling of cylindrical parts is a fully three dimensional process, and must be modeled as such if such behavior is expected. An axisymmetric simulation will not show buckling; even if it will occur in the actual process (Figure 1.5.2).Parts which cannot be simplified to 2D must be modeled as 3D.

1.6. The DEFORM system

The DEFORM system consists of three major components:

1. A *pre-processor* for creating, assembling, or modifying the data required to analyze the simulation, and for generating the required database file.
2. A *simulation engine* for performing the numerical calculations required to analyze the process, and writing the results to the database file. The simulation engine reads the database file, performs the actual solution calculation, and appends the appropriate solution data to the database file. The simulation engine also works seamlessly with the Automatic Mesh Generation (AMG) system to generate a new FEM mesh on the work piece whenever necessary. While the simulation engine is running, it writes status information, including any error messages, to the message (.MSG) and log (.LOG) files.
3. A *post-processor* for reading the database file from the simulation engine and displaying the results graphically and for extracting numerical data.

1.7. Pre-processing

The DEFORM preprocessor uses a graphical user interface to assemble the data required to run the simulation. Input data includes

Object description
Includes all data associated with an object, including geometry, mesh, temperature, material, etc.

Material data
Includes data describing the behavior of the material under the conditions which it will reasonably experience during deformation.

Inter object conditions
Describes how the objects interact with each other, including contact, friction, and heat transfer between objects.

Simulation controls
Includes instructions on the methods DEFORM should use to solve the problem, including the conditions of the processing environment, what physical processes should be modeled, how many discrete time steps should be used to model the process, etc.

Inter material data
Describes the physical process of one phase of a material transforming into other phases of the same material in a heat treatment process. For example, the transformation of austenite into pearlite, banite, and martensite.

1.8. Creating input data

There are several ways to enter data into the DEFORM pre-processor. Depending on the requirements of a particular problem, a combination of the following methods will frequently be used.

Manual input
The pre-processor menus contain input fields for nearly every possible data input in DEFORM. The user can enter, view, or edit any of these values. Discussions of each field are contained in the reference section of this manual.

Keyword file input
Most of the data fields in the DEFORM pre-processor correspond directly to a DEFORM keyword. Individual keywords describe very specific information about a particular object characteristic, simulation control, material characteristic, or inter-object relationship. Keyword data can be saved in a keyword (.KEY) file. A

keyword file is a human readable (ASCII) representation of DEFORM simulation data.
The typical format of a keyword is:

```
[keyword name] [keyword parameters]
[default                                                    data]
[data]
[data]...
```

A keyword file may contain a complete simulation data set, or it may contain only one or a few specific keywords.

Assembling keyword files

When a keyword file is read into the pre-processor, only the specific data fields listed in that keyword are changed; the remainder is unchanged. Thus, it is possible to assemble a complete set of problem data by loading one keyword file that contains only data for one object, another keyword file that contains material data, etc.
To save specific elements of a keyword file, it is necessary to save the entire file, then use a text editor such as Notepad, VI, emacs, or equivalent to delete unwanted information. The keyword file load and save features on the main pre-processor menu load or save an entire data set. To load partial keyword files, use the *Keyword, Load* option from the *File* menu.

Other file inputs

Various data types, particularly part geometries and material data, can be read from appropriate format files.

Modifying problem data

Solution or input step data from any stored step in a database file can be read into the pre-processor, modified, and either appended to an existing database, or written to a new database file.

Viewing specific problem data

Most problem data stored in the database file is accessible in the post-processor. However, certain specific information such as boundary conditions or inter-object contact conditions is displayed differently in the pre-processor. When debugging a problem which is not running properly, it is sometimes useful to use the pre-processor data display to view this information.

1.9. File system

The primary data storage structure is the database file. The database file stores a complete set of simulation data, including object data, simulation controls, material data, and inter-object relations, both from the original input, and from selected solution steps. The sequence of information storage in a database file is shown in Figure 1.9.1 . The pre-processor uses an ASCII format file called the keyword file to create inputs.

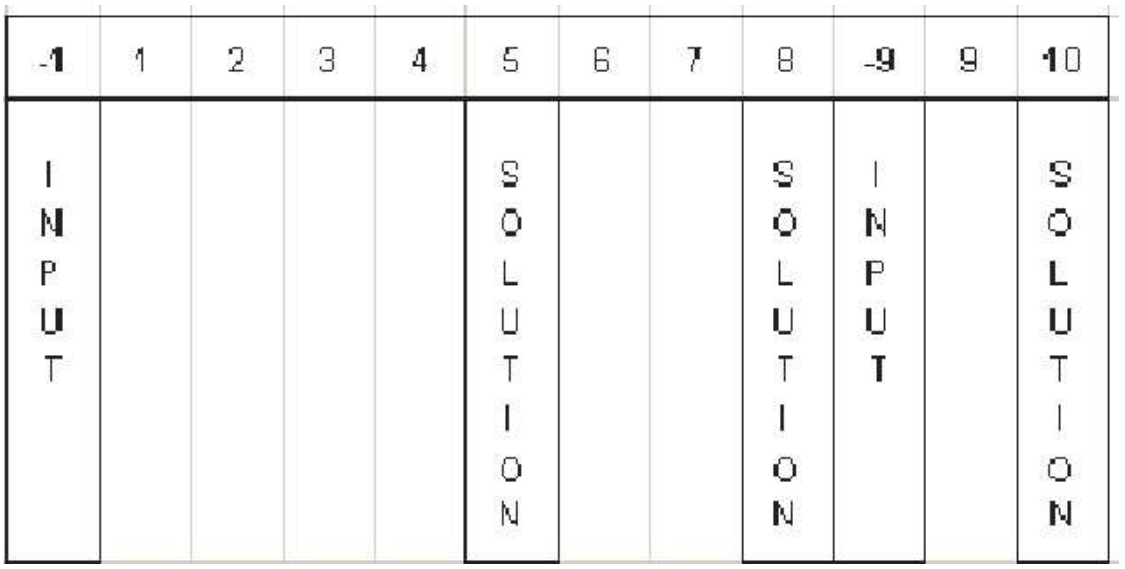


Figure 1.9.1 : DEFORM database structure.

Each DEFORM problem has an associated problem ID and should be created in its own folder or directory. For every problem, the DEFORM system creates four types of files that are generally accessible to users:

Database (DB) files

The database file contains the complete simulation data set for input data and each saved simulation step. The information is stored in a compressed, machine readable format, and is accessible only through the DEFORM pre- and post-processors. As the simulation runs, data for each step is written to the end of the database file. If the step being written is specified as a step to be saved, information for the next step will be appended after the current data step. If the step is not specified to be saved, and a solution is found for the next step, the data for the current step will be overwritten by the data for the next step.

Keyword (KEY) files
Keyword files contain specific problem definition data which is read by the pre-processor and used to create an input database file. A keyword file may contain a complete problem definition, or it may contain only specific information about, for example, a specific object or material. The information is stored in ASCII format, and can be read and edited with any text editor, such as Notepad, VI, or emacs. A keyword reference is available which describes the data format for each keyword.

1.10. Running the simulation

Simulation engine
The simulation engine is the program which actually performs the numerical calculations to solve the problem. The simulation engine reads input data from the database, then writes the solution data back out to the database. As it runs, it creates two user readable files which track its progress.

Log (LOG) files
Log files are created when a simulation is running. They contain general information on starting and ending times, remeshings (if any), and may contain error messages if the simulation stops unexpectedly.

Message (MSG) files
Message files are also created when a simulation is running. They contain detailed information about the behavior of the simulation, and may contain information regarding why a simulation has stopped.

1.11. Post-processor

The postprocessor is used to view simulation data after the simulation has been run. The postprocessor features a graphical user interface to view geometry, field data such as strain, temperature, and stress, and other simulation data such as die loads. The postprocessor can also be used to extract graphic or numerical data for use in other applications.

1.12. Units

DEFORM data may be supplied in any unit system, as long as all variables are consistent (i.e., length, force, time, and temperature measurements are in the same units, and all derived units - such as velocity - are derived from the same base units). This task can be simplified by using either the British or SI system for the default unit system.

Entity	SI unit	English unit	SI unit = English unit * factor
Time	second	second	1.0
Length	mm	in	25.4
Area	mm2	in2	6.4516e2
Volume	mm3	in3	1.6387e4
Force	N	Klb	4.4484e3
Mechanical Energy	N-mm	Klb-in	1.13e5
Stress	MPa	KSI	6.8918
Heat Energy	N-mm	BTU	1.055e6
Temperature	C	F	C = (F-32)/1.8
Conductivity	N/sec/C	Btu/sec/in/F	7.4764e4
Heat Flux Rate	N/mm/sec	BTU/in2/sec	1.6353e3
Heat Capacity	N/mm2/C	BTU/in3/F	1.1589e2
Convection Coefficient	N/sec/mm/C	BTU/sec/in2/F	2.943e3
Lubricant Heat Transfer Coefficient	N/sec/mm/C	BTU/sec/in2/F	2.943e3

Figure 1.12.1 : DEFORM unit system.

Note: It is important to select the unit system at the beginning of the simulation. Once numerical values have been entered in the pre-processor, the numerical values will remain unchanged even if the unit system designation is changed.

The Post-Processor has been equipped with a feature for unit conversion for database viewing. The user has four options for unit conversion. If the conversion factor selected is Default, then the units are picked up automatically depending on whether the database is English or SI. Since there is no conversion necessary, all the conversion factors are set to 1.0 in this column. For the cases of converting English to SI or converting SI to English, the conversion factors and units are picked up from the dialog and the values are converted and displayed in the post-processor. The fourth option gives the user the option of viewing the data from the database in units that are not English or SI. The user is free to enter the conversion factors and the units corresponding to the conversion factors. **There is no user type unit conversion for temperature, since the temperature conversion is not a simple multiplication.**

Chapter 2. Pre-Processor

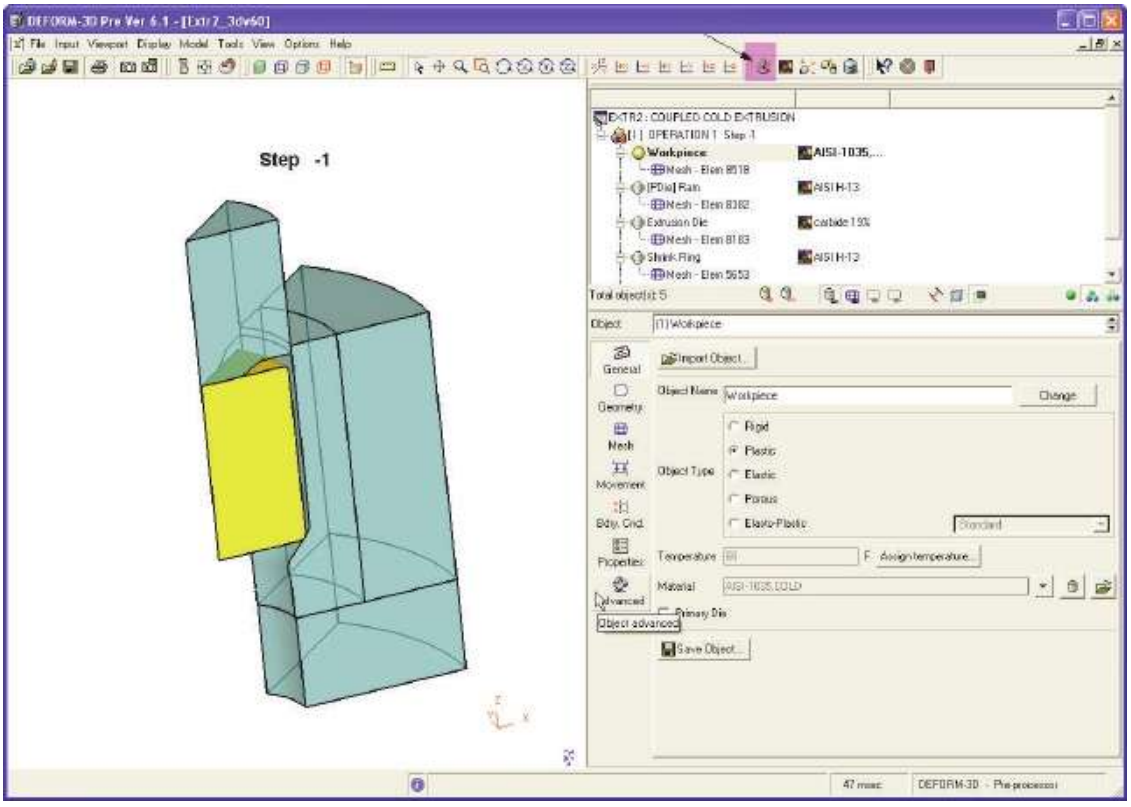


Figure 2.1.0 : The preprocessor of DEFORM-3D. The simulation controls button is highlighted with a red square.

2.1. Simulation Controls

The *Simulation Controls* window can be found by clicking a button in the Preprocessor ( ). Options defined under *Simulation Controls* (See Figure 2.1.1) control the numerical behavior of the solution. *Main controls* details with specifying the simulation title, unit system, geometry type, etc. *Stopping and step controls* are used to specify the time step, the total number of steps and the criteria used to terminate the simulation. Processing conditions like the environment temperature, convection coefficient can be specified under *Processing conditions*. Certain advanced features are explained in the *Advanced controls* section.

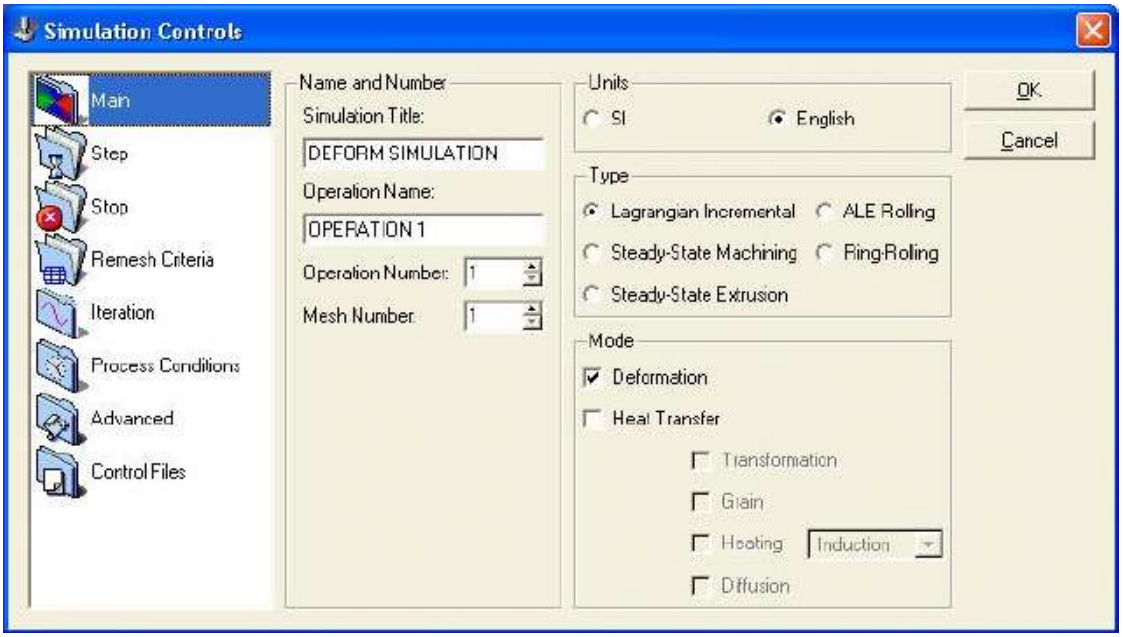


Figure 2.1.1 : Simulation Controls window.

2.1.1. Main controls

Simulation title (TITLE)
The simulation title allows you to title the problem (up to 80 characters) for reference purposes.

Operation name (SIMNAM)
The operation name allows you to title the specific operation (up to 80 characters) for reference purposes.

Units (UNIT)
The DEFORM unit system can be defined as English or Metric (SI). All information in DEFORM should be expressed in consistent units. The unit system should be selected at the beginning of the problem setup procedure, and should not be changed during a simulation or after an operation.

Eotity	SI unit	English unit	SI unit = English unit * factor
Time	second	second	1.0
Length	mm	in	25.4
Area	mm2	in2	6.4516e2
Volume	mm3	in3	1.6387e4
Force	N	Klb	4.4484e3
Mechanical Energy	N-mm	Klb-in	1.13e5
Stress	MPa	KSI	6.8918
Heat Energy	N-mm	BTU	1.055e6
Temperature	C	F	C = (F-32)/1.8
Conductivity	N/sec/C	Btu/sec/in/F	7.4764e4
Heat Flux Rate	N/mm/sec	BTU/in2/sec	1.6353e3
Heat Capacity	N/mm2/C	BTU/in3/F	1.1589e2
Convection Coefficient	N/sec/mm/C	BTU/sec/in2/F	2.943e3
Lubricant Heat Transfer Coefficient	N/sec/mm/C	BTU/sec/in2/F	2.943e3

Figure 2.1.2 : DEFORM unit system.

Type

The five different types of simulations that can be run are:

- Lagrangian Incremental: To be used for all the conventional forming, heat transfer and heat-treat applications. Transient phase of the processes like rolling, machining, extrusion, drawing cogging etc. also can be modeled in this general framework.

- ALE Rolling: ALE model for rolling process can be generated using the 'Shape Rolling template'. When the model is generated using this template, automatically generates the necessary boundary conditions for the entry surface for the billet (indicated in the interface as the Beginning surface, nodes are assigned BCCDEF=4), and the exit surface (indicated in the interface as Free surface, nodes are assigned BCCDEF=5). Template automatically sets the analysis type as 'ALE Rolling'. When the rolling model is setup using the regular pre-processor, user needs to set this analysis type and proper boundary conditions to be able to run the ALE model for rolling.

- Steady-State machining: 3D machining model for turning applications can be generated using the 'Machining Template' in which the initial model can be set up for Lagrangian Incremental run. When sufficient chip has formed the template can be used to generate an additional operation to switch the analysis mode to Steady State. In this stage template can be used to generate the required boundary conditions for the steady state run, which includes defining end surface of the chip (indicated as free surface, with BCCDEF code set as 5 for those nodes). Template automatically sets the analysis type as 'Steady-State Machining'. When the machining model is setup using the regular pre-processor, user needs to set this analysis type and proper free surface and thermal boundary conditions to be able to run the Steady State model for machining.

- Ring Rolling: From 3DV61, simulation engine has been enhanced to handle the non isothermal modeling of ring rolling process. This development includes a special ALE technique that does not depend on any expensive computing resources, nor involves very long modeling times.

- Steady-State extrusion: Provided for future implementation: (Current Eulerian process modeling capability for extrusion, which is under development can be activated using a special data file called 'ALE.DAT'. Please contact SFTC for additional information.)

Simulation modes (SMODE, TRANS)

DEFORM features a group of simulation modes that may be turned on or off individually, or used in various combinations.

Heat transfer

Simulates thermal effects within the simulation, including heat transfer between objects and the environment, and heat generation due to deformation or phase transformation, where applicable.

Deformation

Simulates deformation due to mechanical, thermal, or phase transformation effects.

Transformation

Simulates transformation between phases due to thermomechanical and time effects.

Diffusion

Simulates diffusion of carbon atoms within the material, due to carbon content gradients.

Grain

Simulates grain size calculation and recrystallization calculations.

Heating

Simulates heat generation due to resistance or induction heating. This feature is not activated in the current release.

For backward compatibility with old keywords and databases, before version 3.0, the keyword SMODE (old style isothermal, non-isothermal, heat transfer) is read and the corresponding keyword TRANS mode switches are set in the pre-processor.

Operation number (CURSIM)
Allows the specification of a new operation number for each simulation in the database. If operations numbers are specified, the post-processor displays each operation with its number in the step list.

Mesh number (MESHNO)
This variable records the current mesh based on the number of remeshings that occur between the initial mesh and the current mesh. This variable should not be changed.

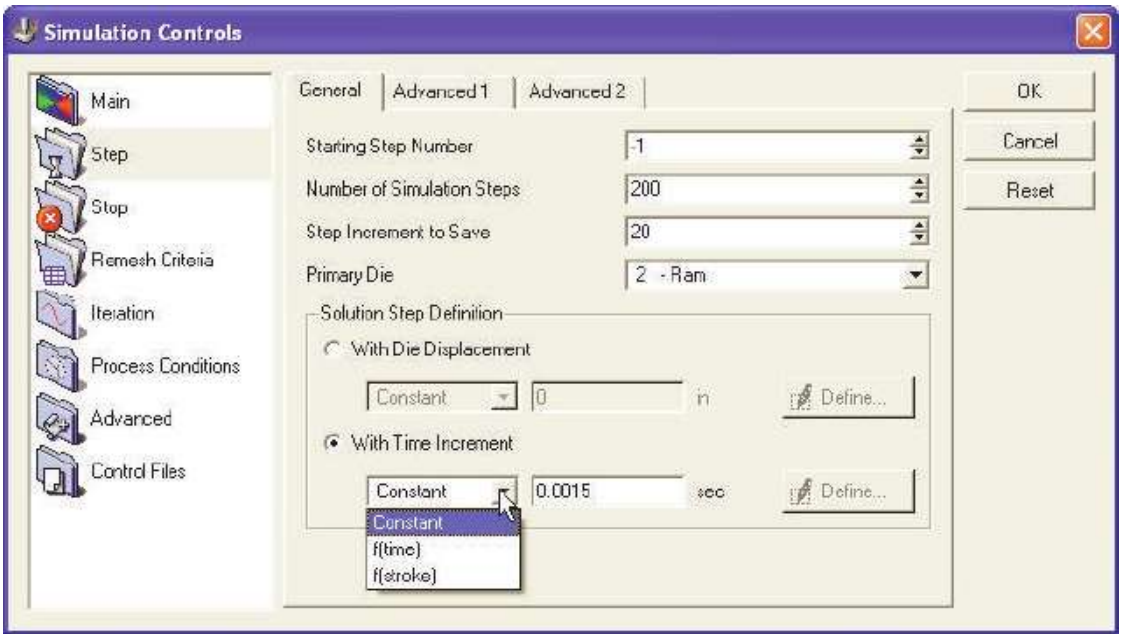


Figure 2.1.3 : Step Controls.

2.1.2. Step Controls

The DEFORM system solves time dependent non-linear problems by generating a series of FEM solutions at discrete time increments. At each time increment, the velocities, temperatures, and other key variables of each node in the finite element mesh are determined based on boundary conditions, thermomechanical properties of the work piece materials and possibly solutions at previous steps. Other state variables are derived from these key values, and updated for each time increment. The length of this time step, and number of steps simulated, are determined based on the information specified in the step controls menu (See

Figure 2.1.3).

Starting step number (NSTART)

If a new database is written, the specified step number will be the first step in the database. If data is written to an existing database, the preprocessor data will be appended to this database in proper numerical order, and any steps after the one specified will be overwritten.
The negative (-n) flag on the step number indicates that the step was written to the database by the pre-processor (either by manual generation of a database step or by an automatic remesh), not by the simulation engine.

Note: All pre-processor generated steps should have a negative step number

Number of simulation steps (NSTEP)

The number of simulation steps parameter defines the number of steps to run from the starting step number. The simulation will stop after this number of simulation steps will have run, if another stopping control is triggered to stop the simulation or if the simulation runs into a problem. For example, if the starting step number is -35 (NSTART), and 30 steps (NSTEP) are specified, the simulation will stop after the 65th step, unless another stopping control is triggered first.

Step increment to save (STPINC)

The step increment to save in the database controls the number of steps that the system will save in the database. When a simulation runs, every step must be computed, but does not necessarily need to be saved in the database. Storing more steps will preserve more information about the process; consequently it will require more storage space.

Primary die (PDIE)

The primary die is the object for which many stopping and stepping criteria are defined. For example, stopping distance based on primary die stroke. When the stroke of the object defined as the primary die reaches the value for primary die displacement, the simulation will be stopped whether or not more steps were specified. The Step By_Stroke feature determines step size based on the movement of the primary die.
The primary die is usually assigned to the object most closely controlled by the forging machinery. For example, the die attached to the ram of a mechanical press would be designated as the primary object.

Step increment control (DSMAX/DTMAX)

Solution step size can be controlled by time step or by displacement of the primary die. If stroke per step is specified, the primary die will move the specified amount in each time step. The total movement of the primary die will be the displacement per step multiplied by the total number of steps. If time per step is specified, the time interval per step will be used. The die displacement per step will be the time step times the die velocity.
From 3DV61, the definition of step increment control has been enhanced to include both the time and stroke dependent step functions. This means, step size (both time per step and stroke per step) can now be defined as a function of time or stroke. This functionality enables finer resolution of saved model information, where it is desired. (Typically towards the end of the stroke, where steep changes of die load and cavity filling or flash formation can take place).
Stroke per step is frequently more intuitive. However, time per step must be specified for any problem in which there is no die movement (such as heat transfer), or for any problem where force control is used.

Selecting time step and number of steps

Proper time step selection is important. Too large a time step can cause inaccuracy in the solution, rapid mesh distortion or convergence problems. Too small a time step can lead to unnecessarily long solution times. The following section provides some guidelines for selecting time steps.

The maximum displacement for any node should not exceed about 1/3 the length of its element edge length in one step. For flow around a tight corner, flash forming, or similar highly localized deformations, time steps may need to be defined to give a node movement of as small as 1/10 or the element edge length. Thus, for a finer mesh, smaller steps are required than for a coarser mesh. This prevents the mesh from becoming overly distorted in a single time step.
The time step can be determined by the following method:

1. Using the measurement tool, measure one of the smaller elements in the deforming object (this must be done after a mesh has been generated)
2. Estimate the maximum velocity of any region of the work piece (for most problems, this will be the die velocity. For extrusion problems it will be the die velocity times the extrusion ratio) If some steps have already be run, display object velocity under Object->Nodes (use the ``eye" icon to display a velocity vector plot and maximum and minimum values).
3. Divide the result of 1 by the result of 2, and take about 1/3 of this value as the time step. This is a rough estimate, so extreme accuracy is not critical.
4. The number of steps is given by where n is the number of steps, x is the total movement of the primary die, V is the primary die velocity, and is the time increment per step.

Refer also to the Polygon Length Sub-Step feature under Advanced Step Controls
If there is insufficient information available to calculate the total number of steps, three alternatives are available:

1. A general guideline of 1% to 3% height reduction per step can be used.
2. Specify an arbitrarily large number of steps, and use an alternative stopping control, such as time or total die stroke.
3. Make a good estimate of the number of steps required for the given step size, and then specify about 120% of this value. Allow the simulation to overshoot the target, and then use a step near, but not at the end as a final solution.

2.1.3. Advanced Step Controls

This menu gives more options for special simulations where precision control of time step size is required (See Figure 2.1.4 and Figure 2.1.5).

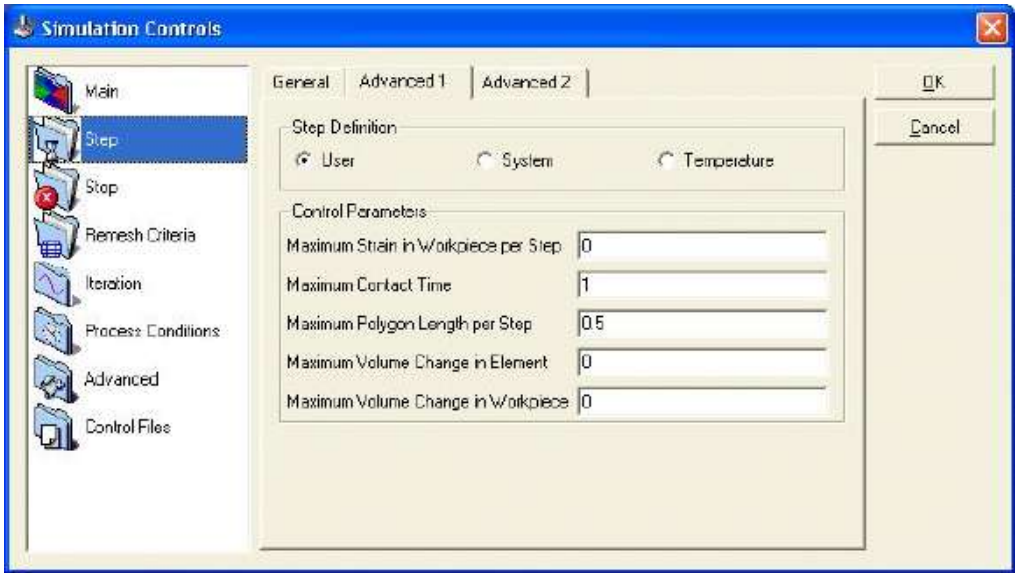


Figure 2.1.4 : Advanced stepping menu 1.

Step definition (STPDEF)
There are three modes for defining steps

- **User** In user defined steps mode, the steps correspond to the NSTEP value. This is the default which does not have to be changed in almost all cases.

- **System** In the system defined steps mode each sub step is saved to the database and is treated as a step. This option is primarily used for debugging purposes.

- **Temperature** In temperature based sub stepping, the DTPMAX settings control the time stepping. The purpose for these controls is to specify the time stepping of a simulation that is driven by thermal-induced deformation.

Strain per step (DEMAX)
The maximum element strain increment limits the amount of strain that can accumulate in any individual element during one time step. If a non-zero value is assigned to DEMAX, a new sub step will be initiated when the strain increment in any element reaches the value of DEMAX.

Contact Time (DTSUB)
Contact time controls whether or not sub stepping is performed when nodes contact a master surface. By default (DTSUB = 0), if a node contacts a master surface a fraction of the way through a time step, the time step is subdivided, and that step is run again at the fraction of the time increment. This will place the node on the surface at the end of the time step. For 3D problems with a large number of nodes contacting master surfaces, this can cause huge increases in execution time.
If DTSUB is set to 1, contact time sub stepping is disabled. Nodes will be allowed to penetrate the master surface, but then will be artificially moved back to surface at the end of the time step. This will allow significantly faster execution time. However, if the defined time step is too large, some volume loss and mesh distortion may occur.
In general, it is recommended that DTSUB be set to 1, and that the time step guidelines described above be followed carefully. Use of polygon length sub stepping, DPLEN, will also control volume loss and mesh distortion, without severe execution time increases.

Polygon length substep (DPLEN)
Polygon length sub stepping places an upper limit on the absolute distance a surface node can move in a given time step. The largest distance a given node can move is defined by

$$\Delta t_{\max} = \frac{(L)(dplen)}{u}$$

Where,

L = the distance from a given node to the nearest adjacent surface on the same object
dplen = the coefficient controlling the relative maximum time step allowed
u = the magnitude of the velocity of the node
Δt_{max} = the maximum time step size allowed

Legal values of DPLEN are from 0 to 1. A value of 0 will disable sub stepping. Recommended values are 0.2 to 0.5, with 0.2 being more conservative, and hence slower, and 0.49 being more aggressive, and faster, but less accurate. Values larger than 0.5 can be used, but may allow unacceptable mesh degeneration.

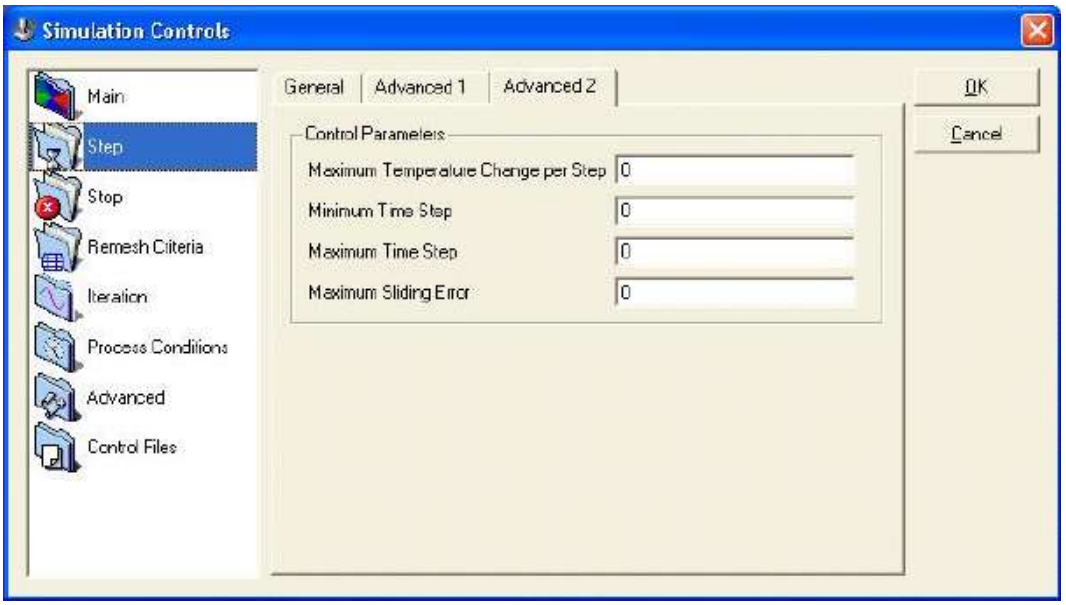


Figure 2.1.5 : Advanced stepping menu 2.

Temperature change per step (DTPMAX)
The maximum temperature change increment limits the amount that the temperature of any node can change during one time step. If a non-zero value is assigned, a new sub step will be initiated when the temperature change at any node reaches the value of DTPMAX. The maximum/minimum time step are the largest and smallest time step allowable with the temperature based sub-stepping.

Maximum Sliding Error
This stepping control is not generally recommended. Please contact SFTC for more information.

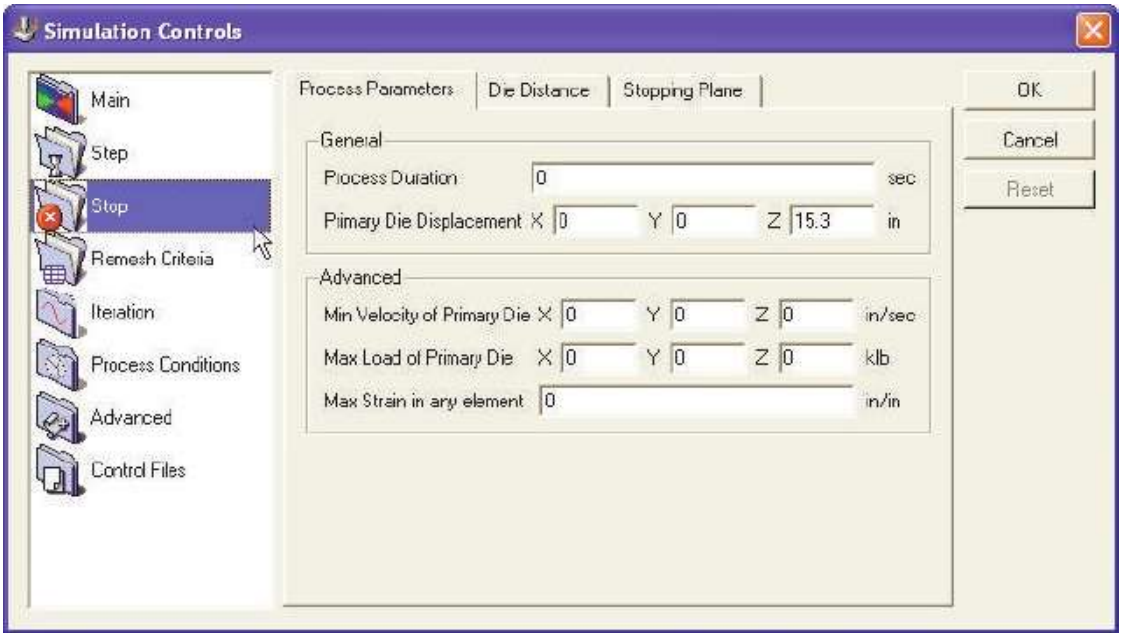


Figure 2.1.6 : Process parameters for stopping a simulation.

2.1.4. Stopping Controls

The stopping parameters determine the process time at which the simulation terminates. A simulation can be terminated based on the maximum number of time steps simulated; the maximum accumulated elemental strain, the maximum process time, or maximum stroke, minimum velocity, or maximum load of the primary object. A simulation will be stopped when the condition of any of the stopping parameters are met. If a zero value is assigned to any of the termination parameters other than number of steps (NSTEP), the parameter will not be used. If no other stopping parameters are specified, the simulation will run until it has utilized all of the specified steps. (See Figure 2.1.6)

Process Duration (TMAX)
Terminates a simulation when the global process time reaches the value specified.

Primary Die Displacement (SMAX)
Terminates a simulation when the total displacement of the primary die reaches the specified value. The stroke value for the object is specified in the *Object, Movement* menu.

Minimum velocity of Primary Die (VMIN)
Terminates a simulation when the X or Y component of the primary die velocity reaches the X or Y values of the VMIN. This parameter is typically used when the primary object movement is under load control, or when the SPDLMT parameter is enforced for a hydraulic press.

Maximum load of Primary Die (LMAX)
Terminates a simulation when the X or Y load component of the primary die reaches the X or Y value of LMAX. Typically used when the movement control of the primary object is velocity or user specified.

Maximum strain in any Element (EMAX)
Terminates a simulation when the accumulated strain of any element reaches the specified value

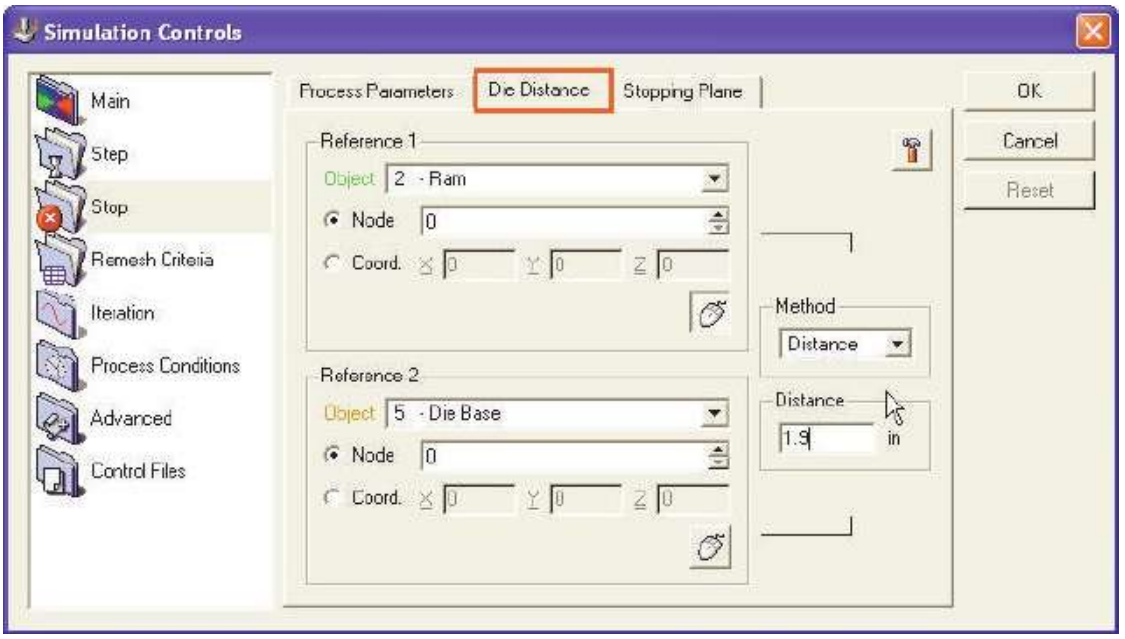


Figure 2.1.7 : Stopping distance based on die distance.

Stopping distance (MDSOBJ)
Terminates a simulation when the distance between reference points on two objects reaches the specified distance. Stopping distance must be used in conjunction with the reference point (REFPOS) definition Die Distance window (See Figure 2.1.7).

Stopping Plane (REFPOS)
Typically used in the models like transient rolling process, user can define a plane in space, and have the simulation terminate once the work piece completely crosses this plane. (See Figure 2.1.8)

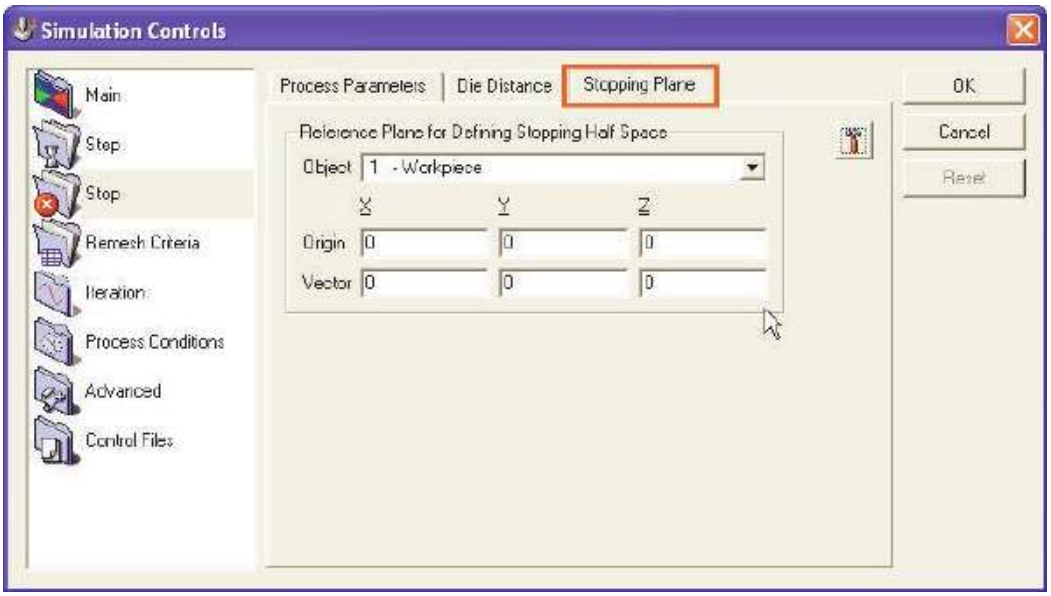


Figure 2.1.8 : Stopping distance based on stopping plane.

2.1.5. Remesh Criteria

Please refer to the section on meshing for a description of this window.

2.1.6. Iteration Controls

The iteration controls specify criteria the FEM solver uses to find a solution at each step of the problem simulation. For most problems, the default values should be acceptable. It may be necessary to change the values if non-convergence occurs (See Figure 2.1.9).

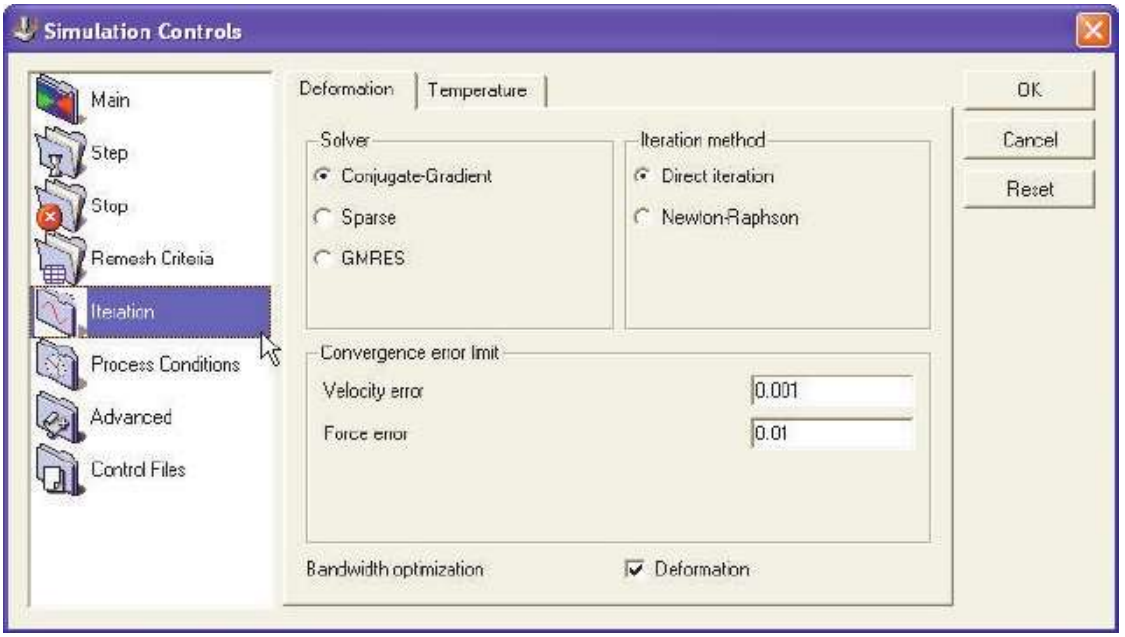


Figure 2.1.9 : Iteration controls for the deformation solver.

Deformation solver (SOLMTD)

The sparse solver is a direct solution that makes use of the sparseness of FEM formulation to improve solution speed. The conjugate-gradient solver tries to solve the FEM problem by iteratively approximating to the solution. For certain problems, this solver offers tremendous advantages over the Sparse solver.

The advantages of the iterative solver include:

- Up to 5:1 improvements in overall solving time, particularly in very large problems
- Ability to handle very large numbers of elements in reasonable time and with reasonable memory demands. (The largest problem to date is 380,000 elements, using 1GB of RAM).
- Much smaller memory requirements for smaller problems - makes 3D practical on inexpensive computers or laptops.

Limitations:

- In certain situations, convergence may be slower, or the simulation may not converge, when the sparse solver will converge. This is particularly a problem for simulations with large "rigid body motion" such as occurs when a part is settling into a die, undergoing light deformation, or bending.

When the conjugate-gradient solver cannot successfully converge toward the solution, DEFORM-3D will fall back to the sparse solver. From 3DV61, a new solver GMRES has been added to the available solvers, to take advantage of multiple CPU environments. The GMRES option can only be used in multi CPU mode.

When to use the iterative solver

The solver is generally very good for problems with a lot of contact with the dies. If a work piece is not well positioned in the dies, or if it will be sliding a bit before it starts deforming, you should start the simulation with the sparse solver. Once there is some substantial deformation in the work piece, stop the simulation, load the final step into the preprocessor, change to "Conjugate Gradient" and "Direct", and write the database.

Keep an eye on the message file for the first few steps. The first step may be a bit slow converging. If the second step is still struggling to converge, or if the simulation stops, you may need to switch back to the sparse solver for a few more steps.

In general, simulations in which you might expect convergence problems using the Sparse solver are not well suited for Conjugate Gradient. Most problems, particularly thin parts or flash parts, will do well after the first 20-30 steps, if not sooner.

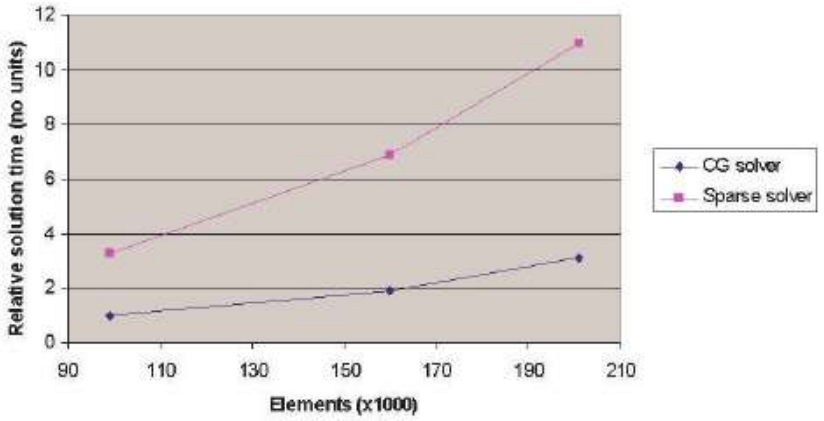


Figure 2.1.10 : Plot of relative time versus elements for different solvers for elastic objects.

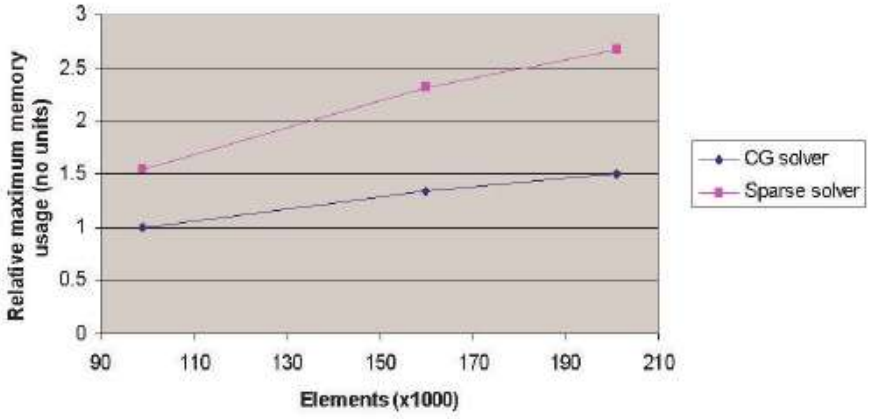


Figure 2.1.11 : Plot of relative memory versus elements for different solvers for elastic objects.

Iteration methods (ITRMTH)
An iteration method is the manner in which the simulation solution is updated (or iterated upon) to try to approach the converged step solution.

- **Newton-Raphson** The Newton-Raphson method is recommended for most problems because it generally converges in fewer iterations than the other available methods. However, solutions are more likely to fail to converge with this method than with other methods.
- **Direct** The direct method is more likely to converge than Newton-Raphson, but will generally require more iteration to do so. In the case of Porous materials, the direct method is the only method currently available.

.
Solver recommendations for 3D

NR : Newton Raphson iterations
DI : Direct iterations
SP : Sparse Solver
CG : Conjugate Gradient Solver
STD : Elasto-Plastic Standard Formulations
MIX : Elasto-Plastic Mixed Formulations
CC : Conformal Coupling (CC) for Contact constraints
PEN : Penalty based contact constraints

Model Data	Recommended	Can be used	Should not be used
General Forming models with Plastic objects (well constrained models)	CG, DI	NR,SP	
General Forming with Elasto-Plastic objects	SP, NR,STD		DI
Spring Loaded Dies	SP		CG
Force Controlled Dies	SP		CG
Heat Treatment with Tet. Mesh Elasto-Plastic	SP, NR, MIX	CG,NR	
Heat Treatment with Brick Mesh Elasto-Plastic	SP, NR	CG,NR	
Multiple Deforming Objects Plastic + Plastic (Large deformation)	SP,DI,CC	CG	
Multiple Deforming Objects	SP,NR,PEN	DI	

Plastic + Plastic (Small deformation)			
Multiple Deforming Objects Elasto-Plastic objects	SP, NR, PEN		DI,CC
Die Stress models Elastic + Elastic Objects	SP, NR	CG	
Rotational Symmetry models (Elasto-Plastic objects)	SP,NR,PEN		CG,CC
Rotational Symmetry models (plastic objects)	SP,DI,CC	CG,NR	
Pure Heat Transfer models	CG	NR	

Convergence error limits (CVGERR)
A deformation iteration is assumed to have converged when the velocity and force error limits have been satisfied. This means that the change in both the nodal velocity norm and the nodal force norm is below the value specified here. The error norm values for each iteration step are displayed in the message file.
If the message file shows that the force or velocity error norms are getting small, but not dropping below the error limits, the simulation may be continued by increasing the appropriate error limit to the smallest value in the message file. This will decrease the solution accuracy, so the simulation should be allowed to run a few steps, then the values should be reduced again. When doing this, extreme care should be exercised. For die stress or press load calculations where extremely accurate force or load values are required, the load accuracy may be improved by decreasing the force error limit. This will increase simulation time, but give more accurate results.

***Note**: It should be noted that the accuracy of the flow stress data will have great impact on the accuracy of die stress and press load predictions.*

Bandwidth optimization (DEFBWD, TMPBWD)
Bandwidth optimization improves solution time by optimizing the structure of the matrix equation being solved. It should be used for almost all problems.

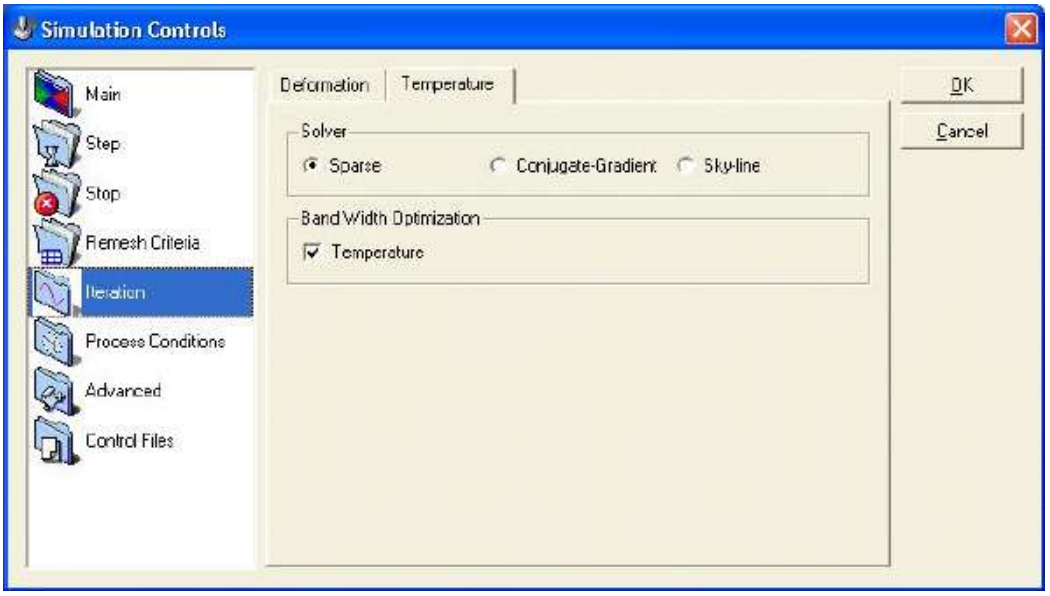


Figure 2.1.12 : Temperature iteration settings.

Temperature solver (SOLMTT)
The sparse solver is a direct solution that makes use of the sparseness of FEM formulation to solve for the temperature. Currently, this is the only solver available for solving thermal problems.

Initial guess (INIGES)
Initial guess generation improves the convergence behavior of the first step of the solution. It should be used for almost all problems.

Bandwidth optimization (DEFBWD, TMPBWD)
Bandwidth optimization improves solution time by optimizing the structure of the matrix equation being solved. It should be used for almost all problems.

2.1.7. Processing Conditions

The processing conditions menu contains information about the process environment, and constants related to general solution behavior.

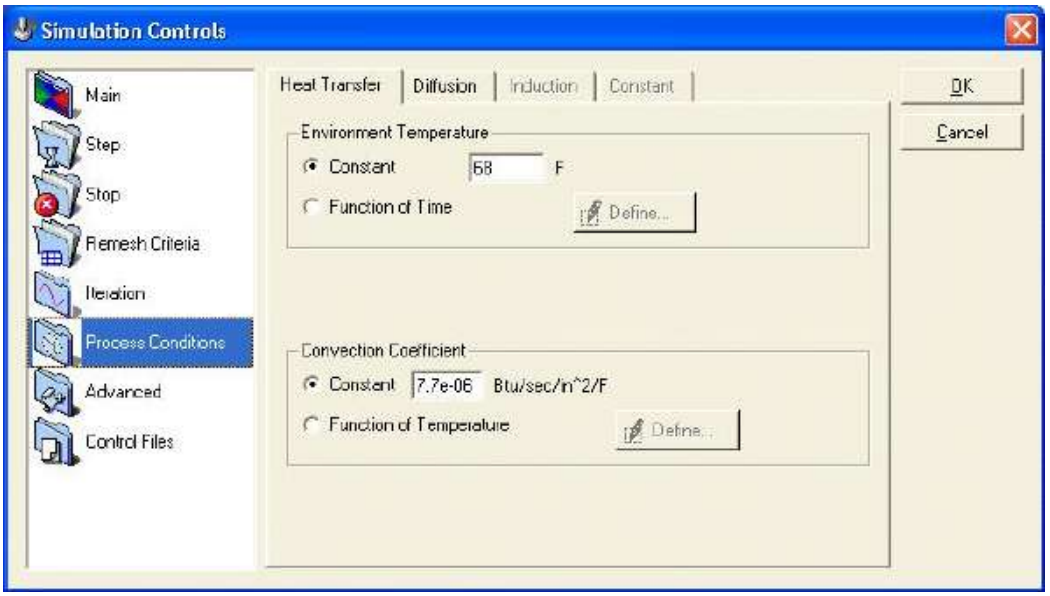


Figure 2.1.13 : Heat transfer processing conditions.

Environment temperature (ENVTMP)

Environment temperature is used in radiation and convection heat transfer calculations and represents the temperature of the area in which the modeled process is taking place. The environment temperature may be specified as a constant or as a function of time. Heat transfer to this temperature is considered to occur from any nodes not in contact with another object. (unless heat exchange windows are used). No radiation view factors are accounted for unless this option is activated. Adding the file DEF_VIEW.DAT to the directory where the simulation is run will activate this. The contents of the file are unimportant.

Convection coefficient (CNVCOF)

The convection coefficient is required for convection heat transfer calculations. The convection coefficient may be specified as a constant or as a function of temperature.

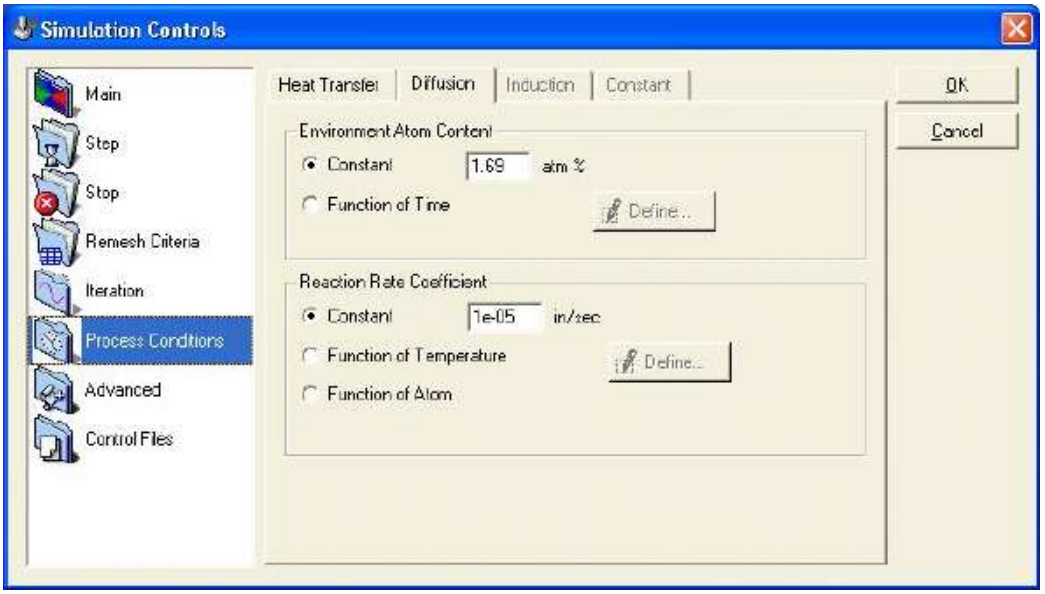


Figure 2.1.14 : Diffusion processing conditions.

Environment atom content (ENVATM) [MIC]

The percentage atom content of the dominant atom (usually carbon) for diffusion calculations.

Reaction rate coefficient (ACVCOF) [DIF]

The surface reaction rate with the atmospheric atom content for diffusion calculations.

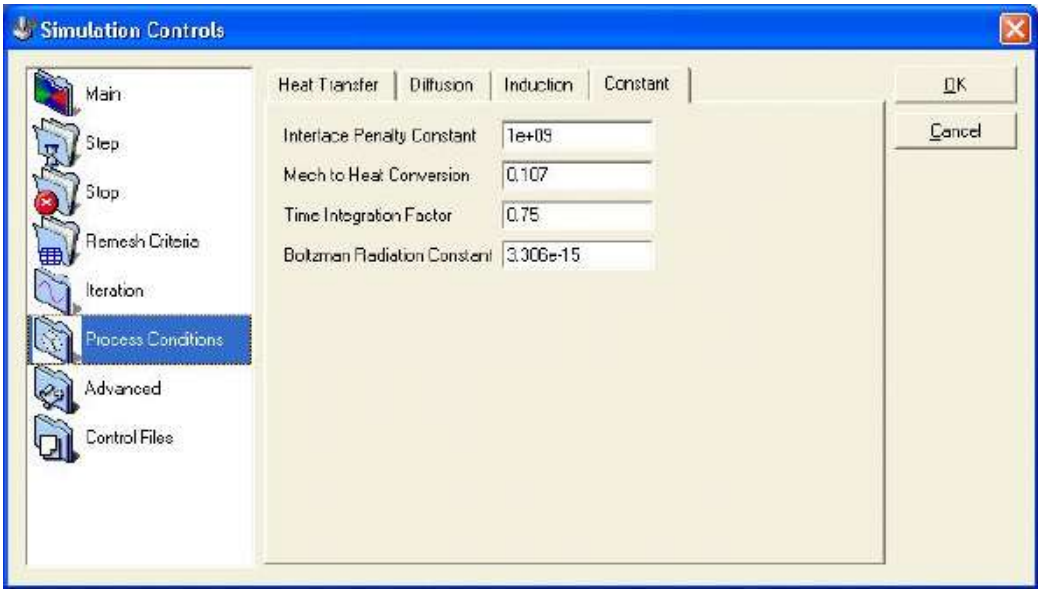


Figure 2.1.15 : Advanced constants.

Interface penalty constant (PENINF)
A large positive number used to penalize the penetration velocity of a node through a master surface. The default value is adequate for most simulations. It should be at least two to three orders higher than the volume penalty constant (PENVOL). For objects of very small size (e.g. fasteners), it is recommended to reduce this number on order of magnitude or two to improve convergence. This will only aid convergence if the sparse solver is used.

Mechanical to heat conversion (UNTE2F)
A constant coefficient to relate units of heat energy(eg BTU) to mechanical energy (eg klb-in). Appropriate constant values are automatically set for English and SI units.

Time integration factor (TINTGF)
The time integration factor is the forward integration coefficient for temperature integration over time. Its value should be between 0.0 and 1.0. The value of 0.75 is adequate for most simulations.

Boltzman constant (BLZMAN)
The Boltzman constant is required for radiation heat transfer calculations. Default values for English and SI are set automatically. In radiation heat calculations the nodal temperature will be automatically converted to absolute temperature (Rankine, Kelvin) based on the selected English or SI units.

2.1.8. Advanced Controls

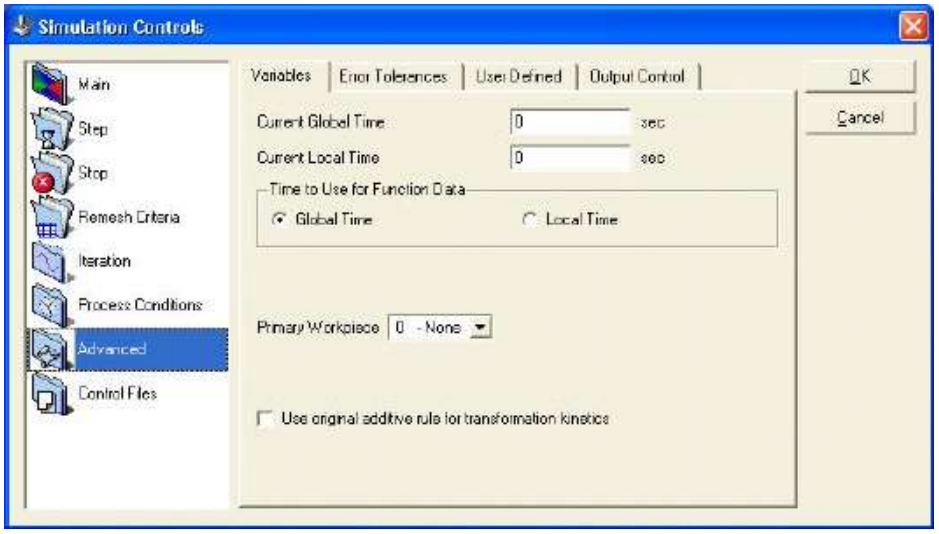


Figure 2.1.16 : Advanced variables.

Current Global Time/Current Local Time (TNOW)

These values specify the global process time and the local process time. The global time is the time since the beginning of the problem, and should never be reset. Local time is a parameter that can be reset by the user. The global time should not be reset during a simulation as the post-processor uses this time for many post-processing operations. Below the local and global time definitions is a selector box that determines which time is to be used for time dependent functions such as movement controls. The default is global time; however, the time dependent functions can also be made a function of local time.

Primary Work piece

This parameter allows the user to specify the work piece as an object that must not possess rigid body motion. If the body does not deform, the simulation will stop. One purpose of this function is to prevent a rolling simulation from continuing past the rolled length of material.

Use original additive rule for transformation kinetics

We have improved the transformation kinetics rule from version 6.0. With the new version, multiple transformations can occur at the same time and temperature for a given material. If the user does not want to use this new rule and wants to use the previous one, checking this box will allow this.

Error Tolerances

Geometry error (GEOERR)

This value is an estimate of the error between discretized objects. The default value for this is sufficient for most of the general applications. (see Figure 2.1.17

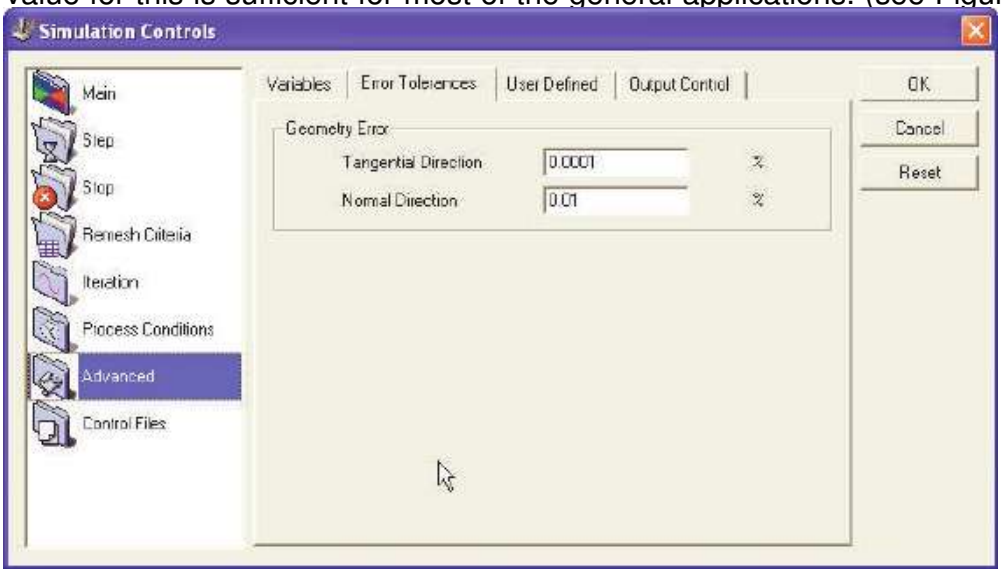


Figure 2.1.17 : Error tolerances window.

User defined variables (USRDEF)
User defined variables are 80 character string variables which are passed to user defined subroutines. Refer to the chapter on User Routines for more information on how to use these variables. (See Figure 2.1.18)

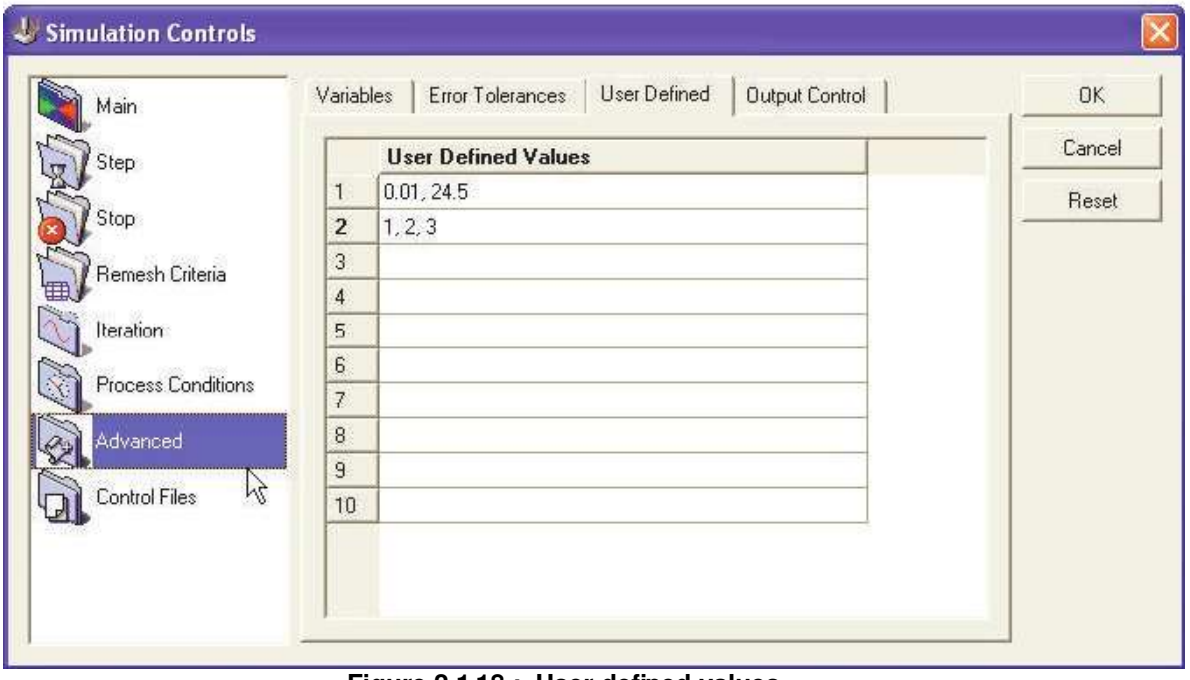


Figure 2.1.18 : User defined values

Output Control
Starting from version 6.0, the simulation control options are further enhanced with two important features.

1. The first among these is to include a wide selection of strain components that can be stored by the user depending upon the analysis and object type. These options for a typical elasto-plastic object enable user to store plastic, elastic and total strains. For non-isothermal models with elasto-plastic objects additionally thermal volumetric strains can also be stored for each stored step of the simulation. When transformation is turned on, the strain components that are produced due to phase transformation can be stored as well. Once set in the Pre-Processor, (Figure 2.1.19) each of these strain components are available in post processing for point tracking, contour plots and other normal display options (Figure 2.1.20).

2. The second option in the output control that is available to the user is intended to improve the state variable representation in the analysis domain and minimize the interpolation error involved in the remeshing procedures. Such representation can also better maintain the local gradients of the state variables compared to the existing the element based representation. In the current version, the user

can choose to represent damage, strain and stress state as Element+Nodal data. This means in addition to the currently available element data, the user can store these variables as nodal variables. In the future versions more state variables will be made available with nodal representation.

These additional nodal and element variables can be accessed from the corresponding nodal and element dialogs (Figure 2.1.21).

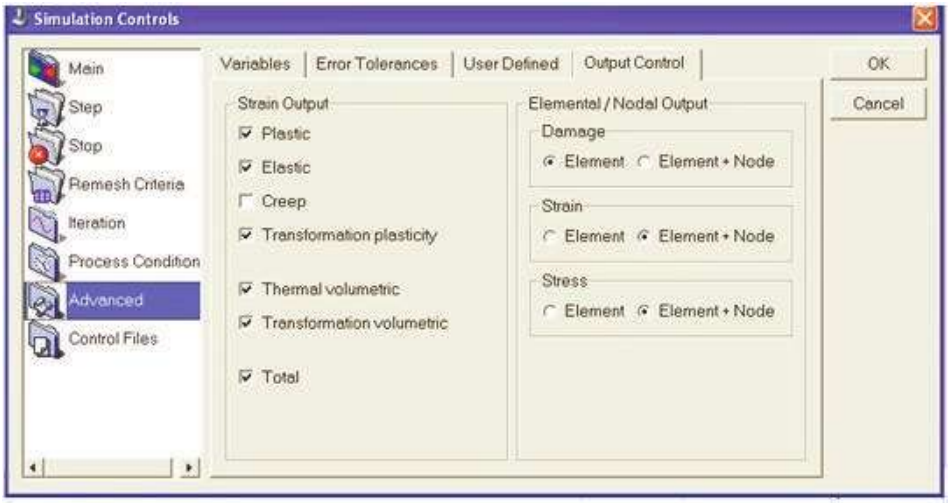


Figure 2.1.19 : Setting the additional strain components and element+node data.

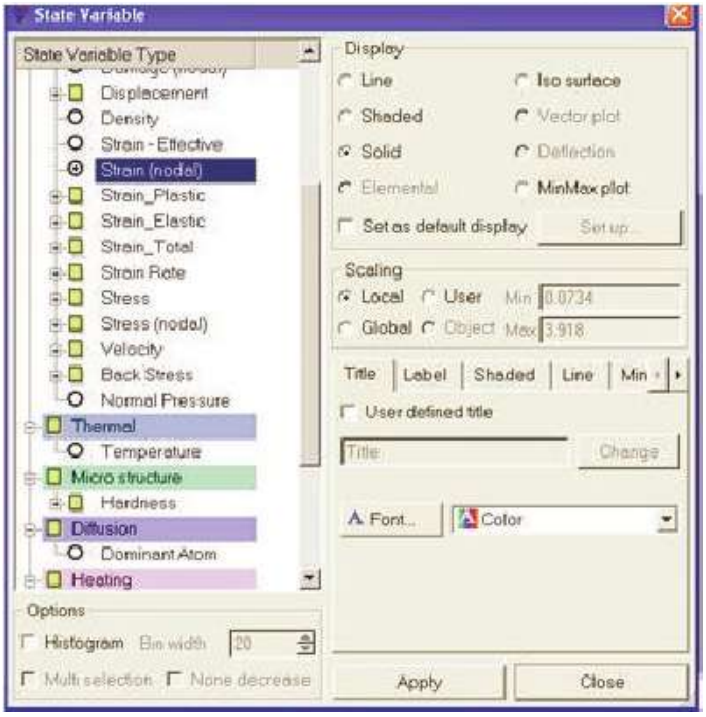


Figure 2.1.20 : State variable list for additional strain components and element+nodal data.

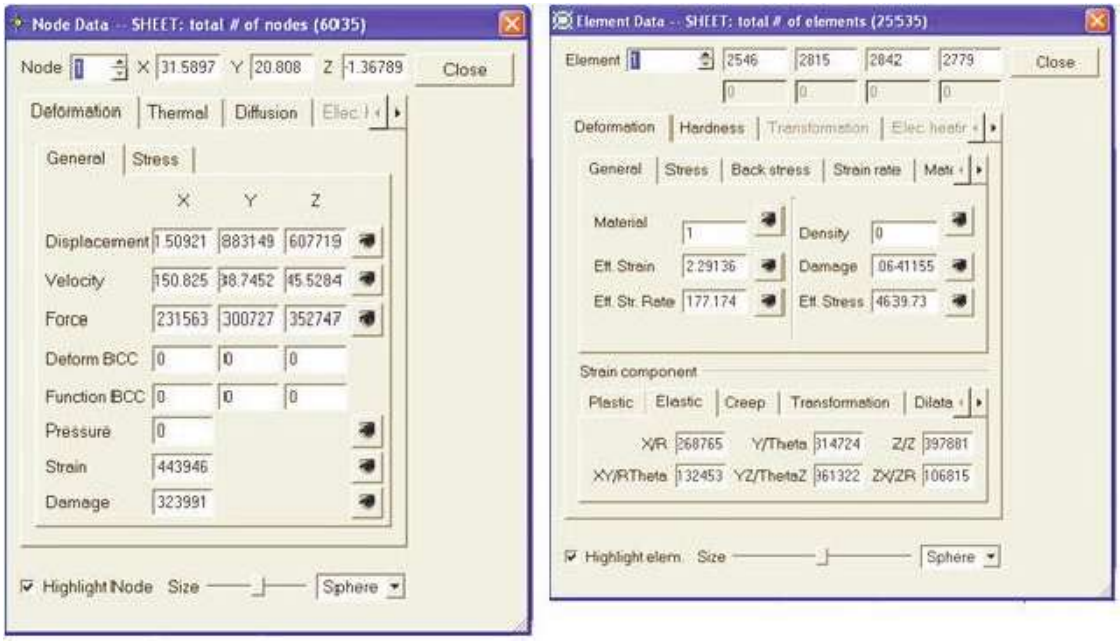


Figure 2.1.21 : Enhanced node and element dialogs including additional nodal variables and strain components.

2.1.9. Control Files

There are many different specialized features within DEFORM-3D that are controlled through data files. The purpose for this type of implementation is that these functions are used in only a few rare instances and if they find popular use, they can be incorporated into DEFORM keywords. When these data files are used, the functionality is available if the data file is located in the same directory as the current problem running. Since they are not contained within neither the database nor the keyword files, the control file has to be moved with the database or the keyword to run the problem with the same functionality if a different directory or computer is used to run a simulation. When one of these control files are used, a warning is automatically posted in the message file heading letting the user know that one of these files exists.

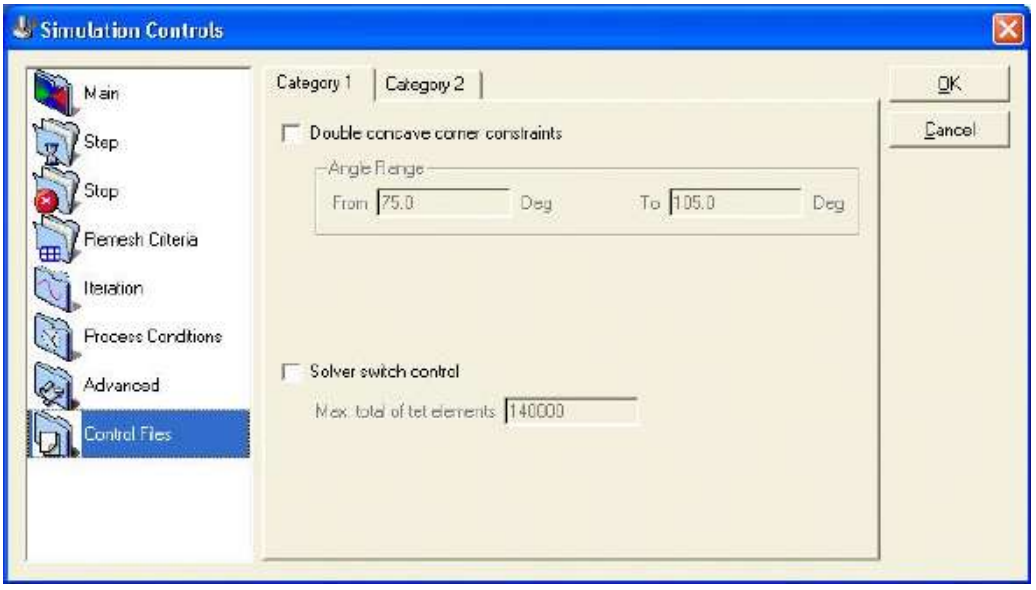


Figure 2.1.22 : Control files selections.

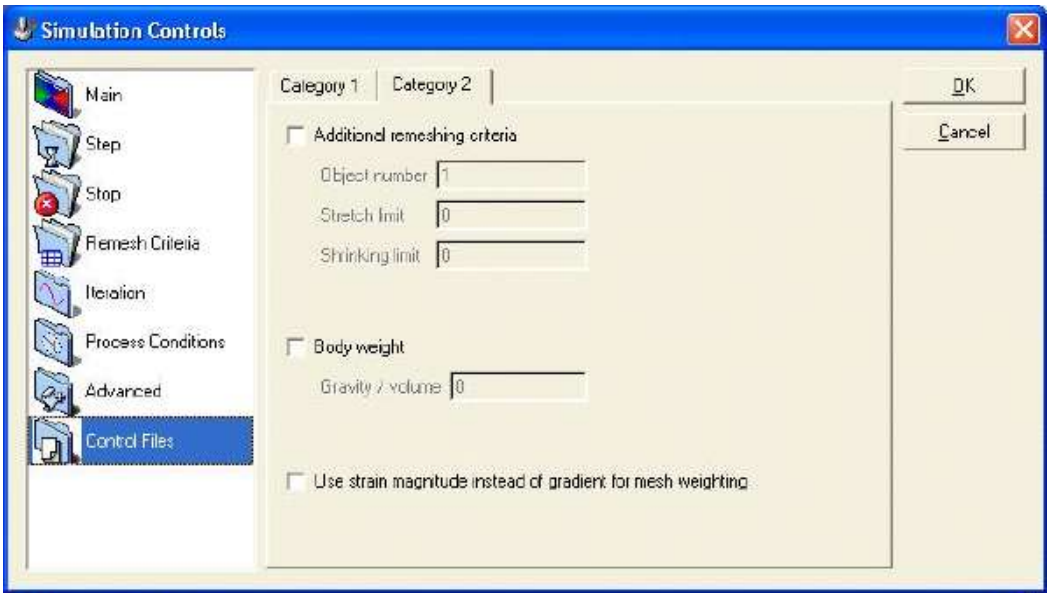


Figure 2.1.23 : Control files dialog (Category 2).

From version 5.0, these data files can be specified through the graphical interface in the Control File window (See Figure 2.1.23).The various categories have different functionalities as follows:

- **Category 1**
  - Double corner constraints – This defines two angles where if a node is contacting a die corner an angle between these values, the node will be given a double contact condition. This is further explained in the appendix.
  - Solver switch control – This defines a number of elements where the switch to sparse solver is blocked. The purpose of this is to prevent the sparse solver from being activated in cases where the problem is too large.

- **Category 2**
  - Additional remeshing criteria – The activation of this feature allows the user to have a finer control on the remeshing criteria.
  - Body weight – This will allow the user to specify the amount of body force per volume of the material. It is not recommended to be used in cases where the body force may be neglected such as times where the material is far from the melting temperature.

2.2 Material Data

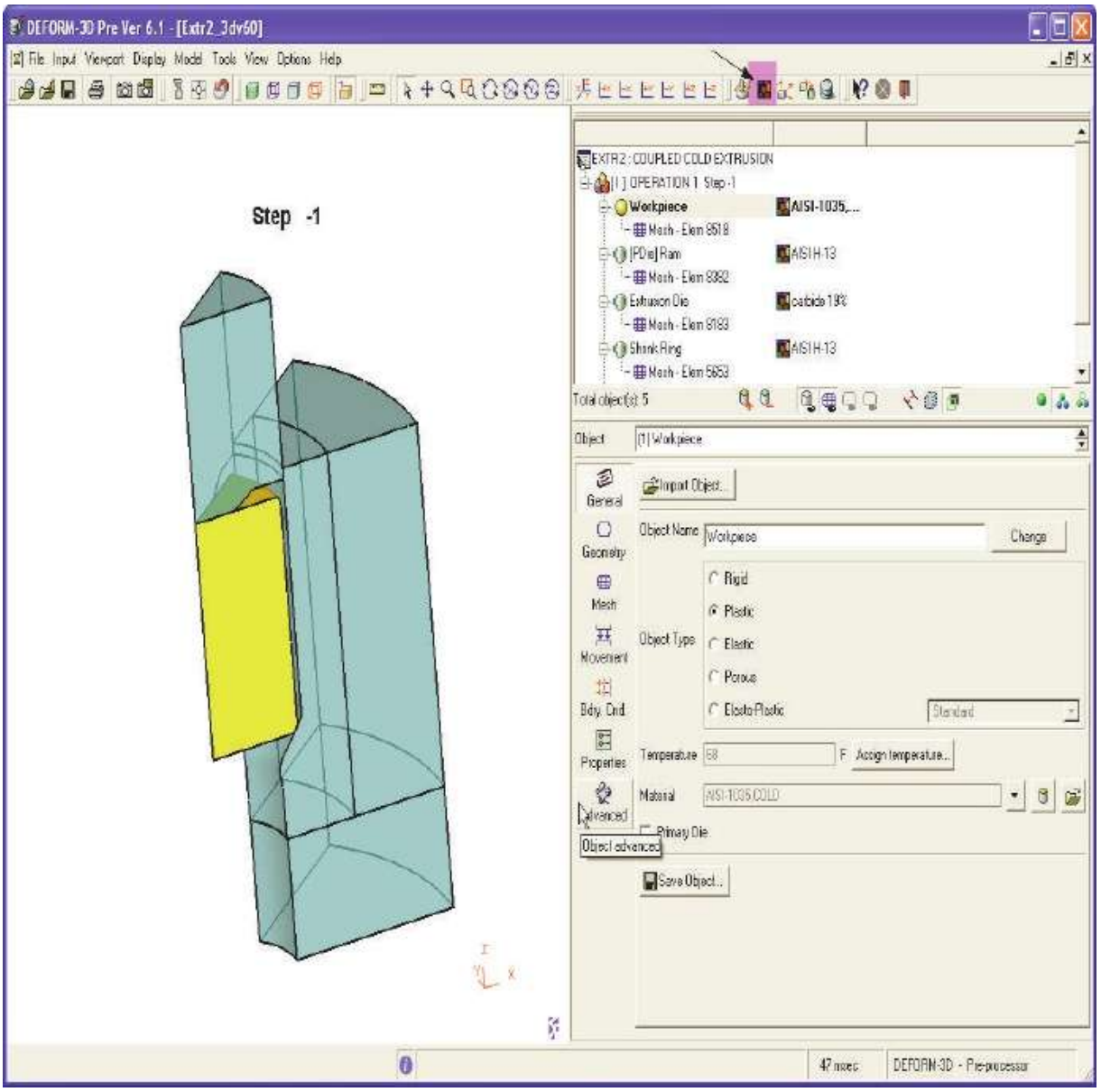


Figure 2.2.1: The material data button highlighted with a red box in the preprocessor.

The material properties window can be accessed by pressing the material properties in the material properties window (See Figure 2.2.1). The material properties dialog in shown in See Figure 2.2.2. In order for a simulation to achieve a high level of accuracy it is important to have an understanding of the material properties required to specify a material in DEFORM. The material

properties that the user is required to specify is a function of the material types that the user is utilizing in the simulation. This section describes the material data that may be specified for a DEFORM simulation.

The different data sets are:

- Elastic data
- Thermal data
- Plastic data
- Diffusion data
- Grain growth/recrystallization data
- Hardness estimation data
- Fracture data

This section discusses the manner in which to define each of these sets of data and which type of simulation each of these are required for.

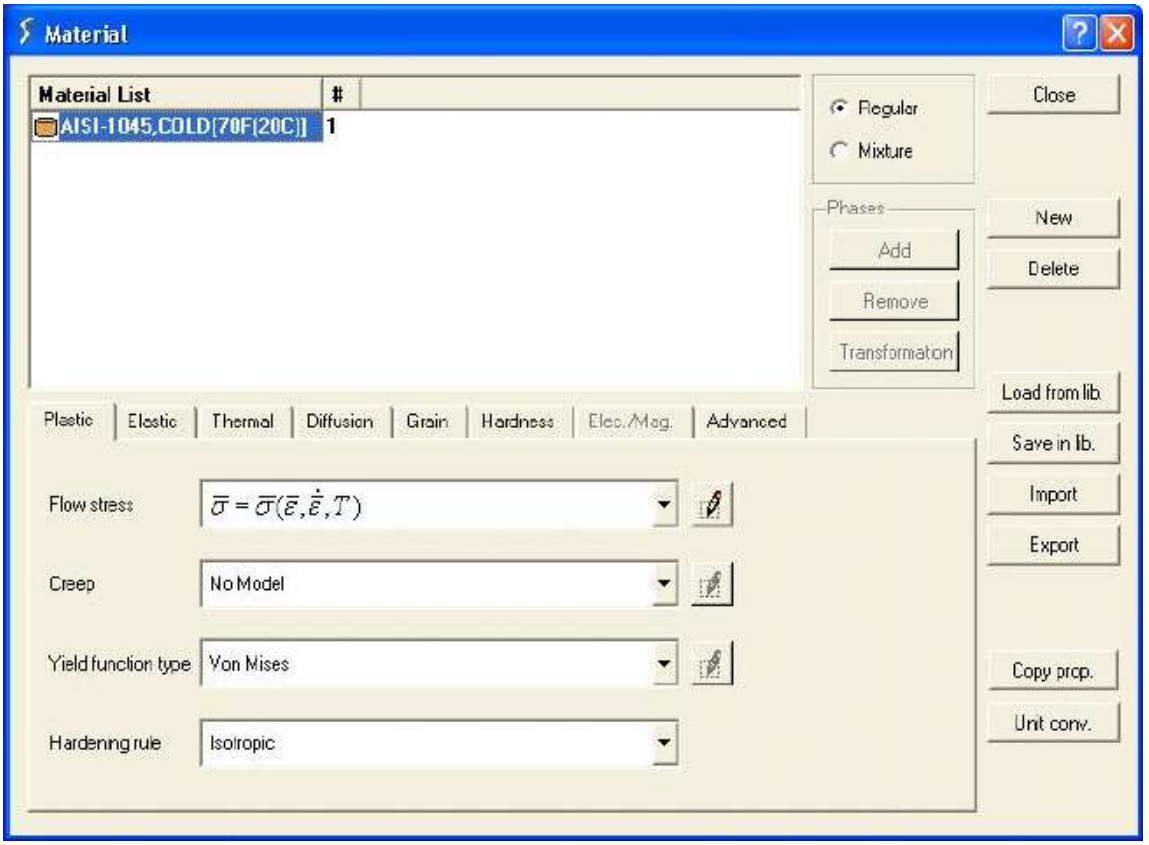


Figure 2.2.2: Defining phases and mixtures within DEFORM-3D.

2.2.1. Phases and mixtures

Material groups can be classified into two categories, phase materials and mixture materials (See Figure 2.2.3). For example a generic steel can exist as Austenite, Bainite, Martensite, etc. During heat-treatment each of the above phases can transform to another phase. So any material group that can

transform to another phase should be categorized as a phase material. The mixture material is the set of all phases for an alloy system and an object can be assigned this mixture material if volume fraction data is calculated.

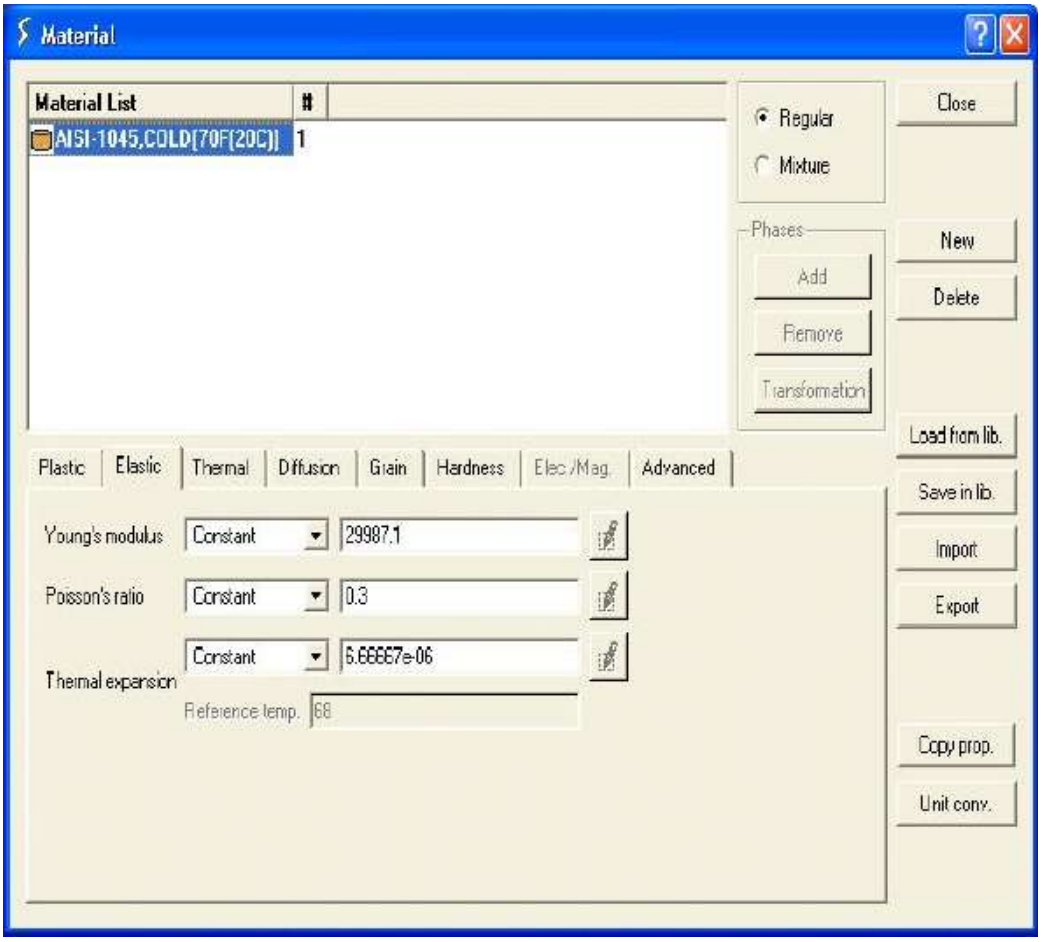


Figure 2.2.3: Defining elastic material data.

2.2.2. Elastic data

Elastic data is required for the deformation analysis of elastic and elasto-plastic materials. The three variables used to describe the properties for elastic deformation are Young's modulus, Poisson's ratio and thermal expansion.

Young's modulus (YOUNG)
Young's Modulus is used for elastic materials and elastic-plastic materials below the yield point. It can be defined as a constant or as a function of temperature, density (for powder metals), dominant atom content (for example, carbon content), or a function of temperature and atom content.

Poisson's ratio (POISON)
Poisson's Ratio is the ratio between axial and transverse strains. It is required for elastic and elasto-plastic materials. It can be defined as a constant or as a

function of temperature, density (for powder metals), dominant atom content (for example, carbon content), or a function of temperature and atom content.

Thermal expansion (EXPAND)

The coefficient of thermal expansion defines volumetric strain due to changes in temperature. It can be defined as a constant or as a function of temperature.
For elastic bodies temperature change is defined as the difference between nodal temperatures and the specified reference temperature (REFTMP):

$$\varepsilon^{th} = \alpha\,(T - T_0)$$

Where,

α is the coefficient of thermal expansion,
T_0 is the reference temperature and
T is the material temperature.

For elasto-plastic bodies the thermal expansion input in the pre-processor is the average value of thermal expansion and the FEM calculates the instantaneous (tangential) value from the average value.

$$\Delta\varepsilon^{th} = \alpha^{*}\Delta T$$

Where,

α^{*} is the tangential coefficient of thermal expansion, and
T is the material temperature.

Experimental data for thermal expansion and conversion tools available

The user interface now allows either direct entry of the tangent thermal expansion coefficient as a function of temperature. The user can also import instantaneous values if available from the experimental data (See Figure 2.2.4). When importing the instantaneous values, the user needs to indicate if the recordings are based on heating or cooling tests and the reference temperature. This instantaneous thermal expansion data can be converted to average data (also called secant, which is the data requirement from the model perspective). At any point the user can see either native data as imported or converted data or both. This data can also be imported and exported as text files. This table data can also be cut and pasted from and to Excel (on PC systems) data table.

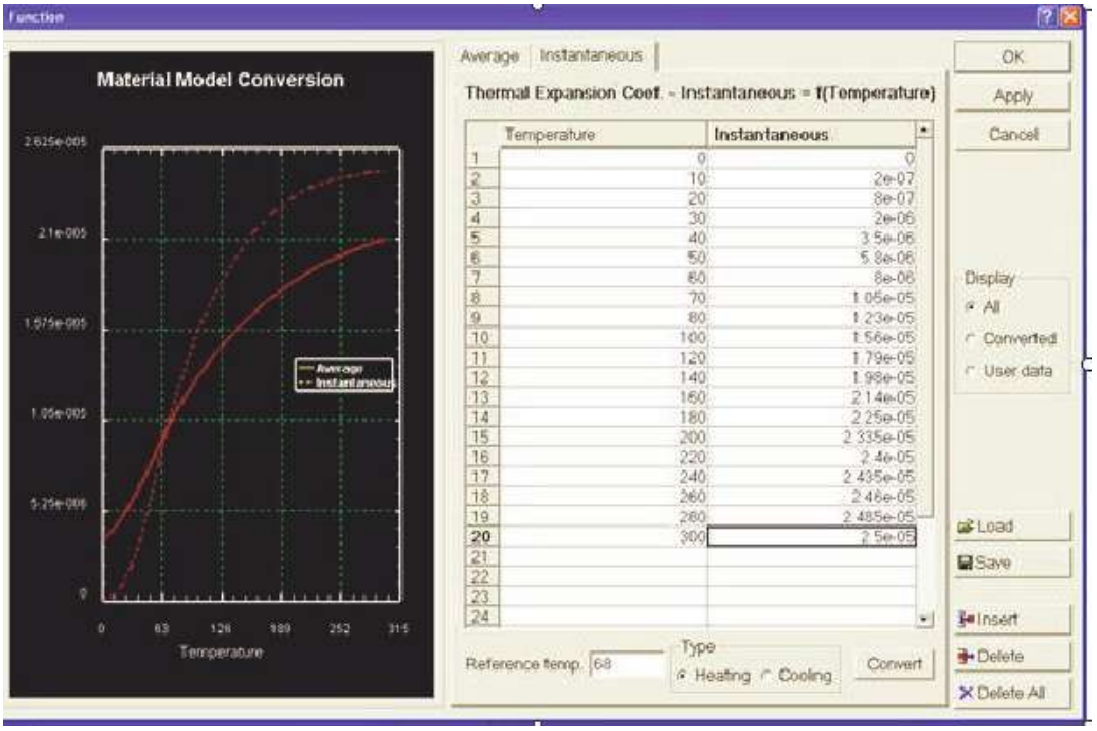


Figure 2.2.4: Data conversion facilities for thermal expansion function data.

Note: To activate the reference temperature option, the thermal expansion coefficient must be made a function of temperature.

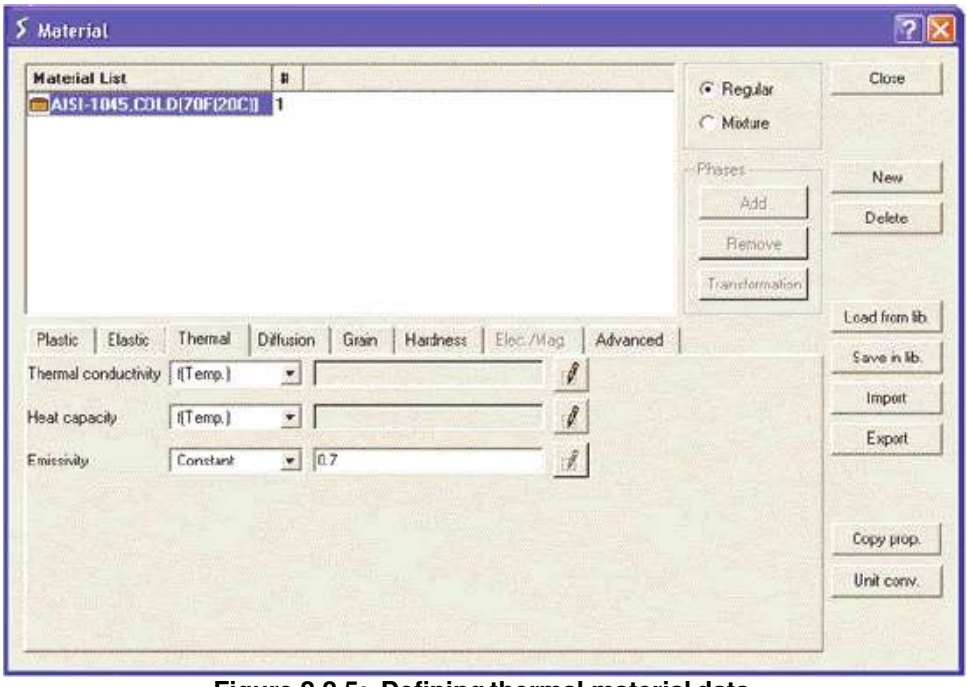


Figure 2.2.5: Defining thermal material data.

2.2.3. Thermal data

Thermal data is required for any object in the heat transfer mode. (See Figure 2.2.5)

Thermal conductivity (THRCND)
Conduction is the process by which heat flows from a region of higher temperature to a region of lower temperature within a medium. The thermal conductivity in this case is the ability of the material in question to conduct heat within an object. The value can be a constant or a function of temperature, a function of atom content, or a function of temperature and atom content.

Heat capacity (HEATCP)
The heat capacity for a given material is the measure of the change in internal energy per degree of temperature change per unit volume. This value is specific heat per unit mass density. The value can be a constant or a function of temperature, a function of atom content, or a function of temperature and atom content.

Emissivity (EMSVTY)
The emissive power, E, of a body is the total amount of radiation emitted by a body per unit area and time. The emissivity, $\mathcal{E}$, of a body is the ratio of E/E_b where E_b is the emissive power of a perfect blackbody. For a more complete description of the properties of emissivity, consult any source dealing with heat transfer. The value can be a constant or a function of temperature.

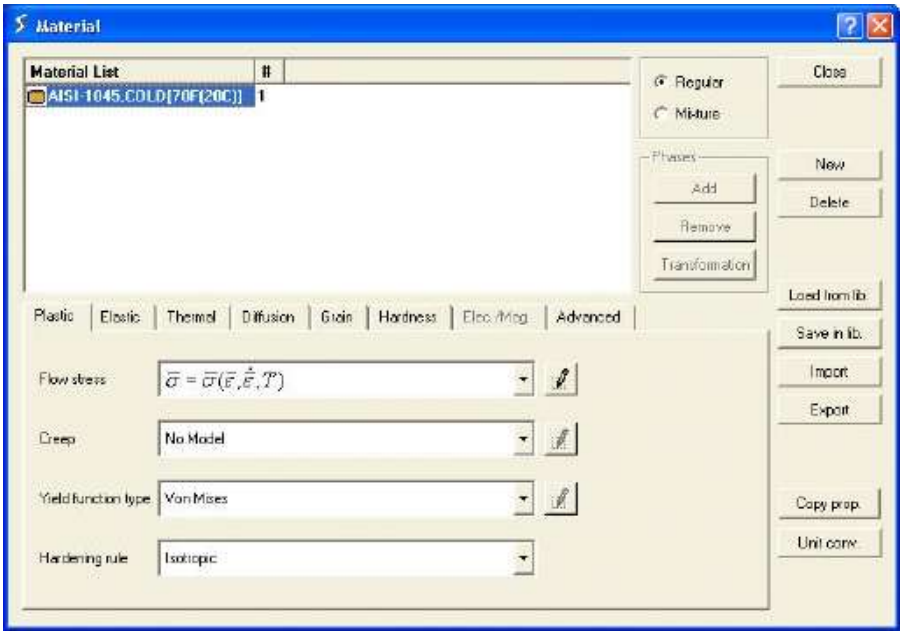


Figure 2.2.6: Defining plastic material data.

2.2.4. Plastic Data

For studying the plastic deformation behavior of a given metal it is appropriate to consider uniform or homogeneous deformation conditions. The yield stress of a metal under uniaxial conditions as a function of strain ($\bar{\epsilon}$), strain rate ($\dot{\bar{\epsilon}}$), and temperature (T) can also be considered as flow stress (See Figure 2.2.6). The metal starts flowing or deforming plastically when the applied stress reaches the value of yield stress or flow stress.
The DEFORM material database has been implemented with around 145 material flow stress data sets. Additional materials will be included as they are available. The material database contains only flow stress data (data for a material in the plastic region). Thermal and elastic properties are not included in the material database.

Note: Flow stress data is compiled from a variety of sources and it is only provided as a reference data set. Material testing should be performed to obtain flow stress data for critical applications.

Material model data conversion utilities
When the material flow stress data is available in the form of data table (Figure 2.2.7), user can convert this data in to a close form model equation using the 'Conversion' utilities. User can select material model from the available list, and fit the model parameters to match the table data points using the curve fitting techniques (Figure 2.2.8). Once this is done, the system displays both forms of the data for the users to proceed with. Typically solid lines in the graph indicate the original data, and the dashed lines from the flow curve computed based on the fitted model parameters.

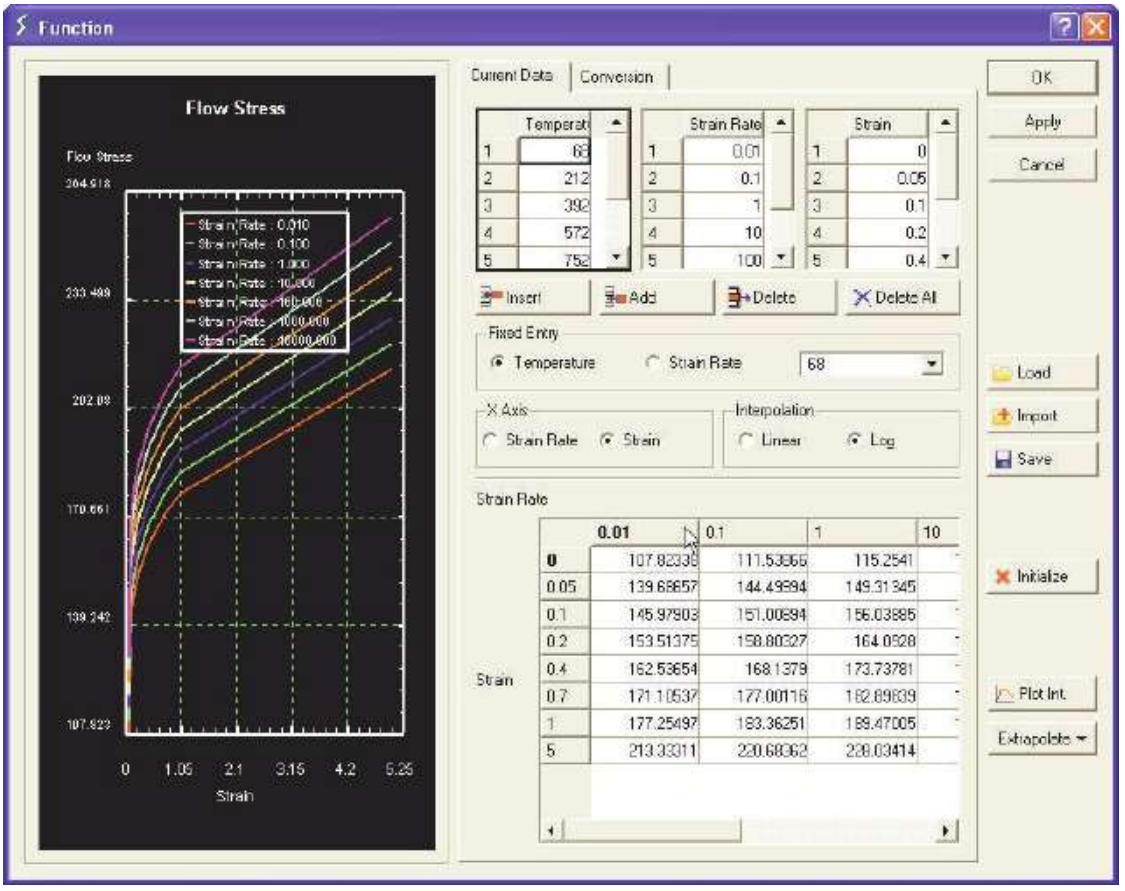


Figure 2.2.7: Material flow stress data in table form in temperature, strain rate and strain dimensions.

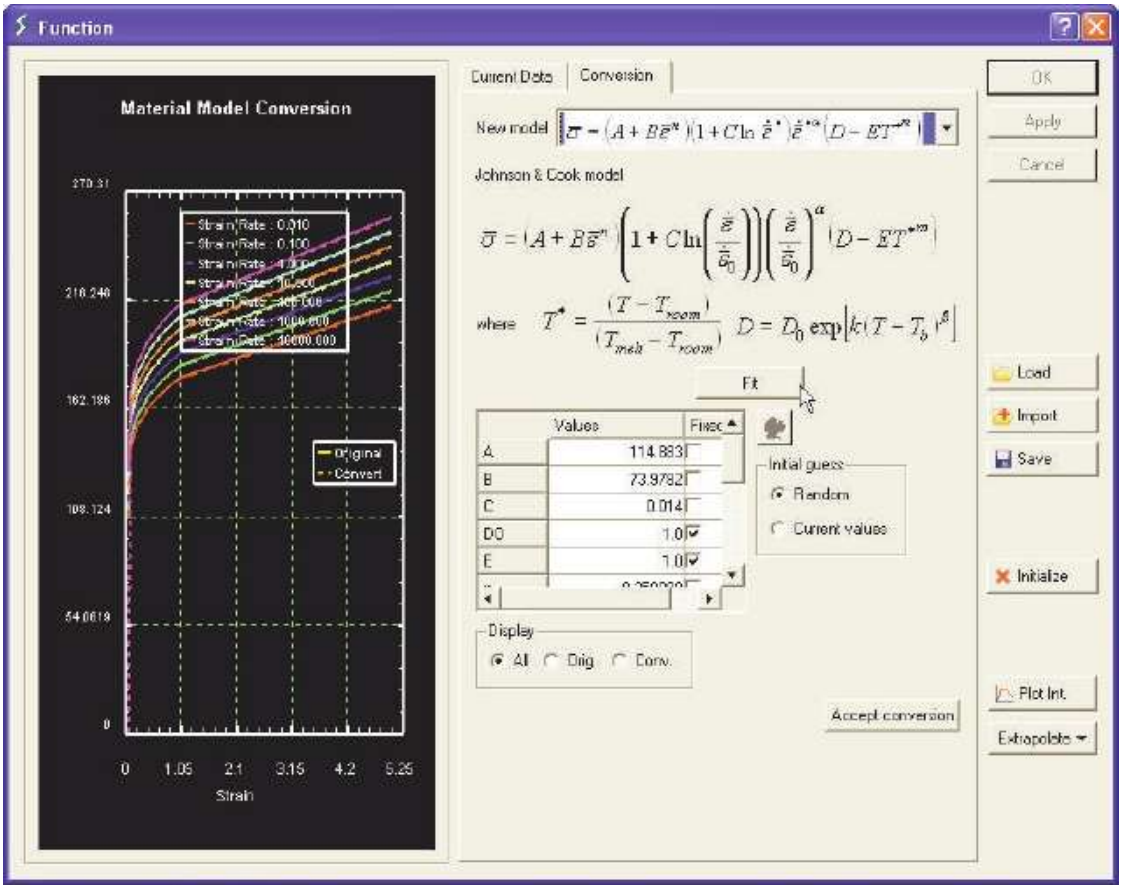


Figure 2.2.8: Results from material model data conversion.

The user should make note that, like any other curve fitting techniques, the nature of original data and initial guess (if user can make one) on the model parameters will greatly influence the quality of the conversion results. This tool also provides options to selectively carryout the curve fitting needs with control over the individual model parameters. Once user accepts the conversion, the converted model data replaces the original table data.
From 3DV61 additional functionality has been added to allow users to import multiple measured flow stress data files, each set at a given temperature and strain rate as shown in Figure 2.2.9



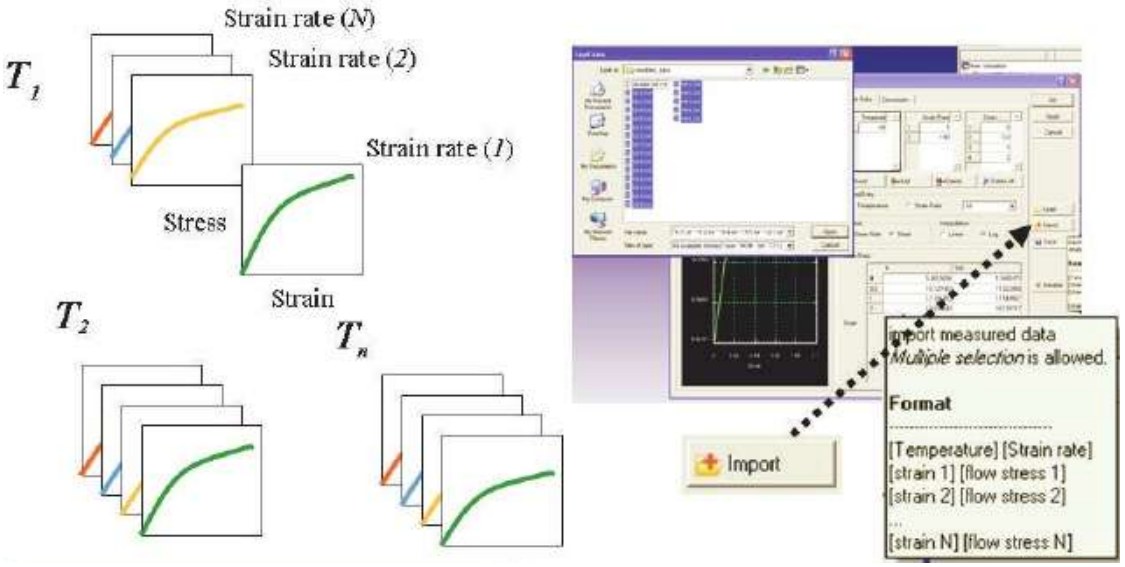


Figure 2.2.9: Utilities to upload the measured flow stress data in to DEFORM system

Flow Stress (FSTRES)

DEFORM provides different methods of defining the flow stress. These forms are shown below:

Power law

$$\bar{\sigma} = c\bar{\epsilon}^{n}\dot{\bar{\epsilon}}^{m} + y$$

Where

$\bar{\sigma}$ = Flow stress

$\bar{\epsilon}$ = Effective plastic strain

$\dot{\bar{\epsilon}}$ = Effective strain rate

c = Material constant

n = Strain exponent

m = Strain rate exponent

y = Initial yield value

Tabular data format

$$\bar{\sigma} = \bar{\sigma}(\bar{\epsilon}, \dot{\bar{\epsilon}}, T)$$

Where

$\bar{\sigma}$ = Flow stress

$\bar{\epsilon}$ = Effective plastic strain

$\dot{\bar{\epsilon}}$ = Effective strain rate

T = Temperature

This method is most highly recommended due to its ability to follow the true behavior of a material. The user is required to enter the values of effective strain, effective strain rate and temperature for which the user has flow stress values.

Interpolation methods:

Linear interpolation

This method takes a linear weighted average between tabular flow stress data points.

Linear interpolation in log-log space

This method takes a linear weighted average between tabular flow stress values in log-log space. If the user inputs a value at zero strain, a linear average between the flow stress value at the zero strain value and the flow stress value at the next highest strain value is linearly interpolated. Using this method the initial yield stress can be defined at a plastic strain of zero. The flow stress values are always interpolated linearly with respect to temperature.

Warning: If simulation conditions of the material exceeds the bounds of the strain, strain rate or temperature defined in the tabular data, the program will extrapolate based on the last two data points which may lead to loss of accuracy.

Flow stress for aluminum alloys (Type 1)

$$\dot{\bar{\epsilon}} = A[sinh(\alpha\bar{\sigma})]^n e^{-\triangle H/RT_{abs}}$$

Where
A = Constant
α = Constant
n = Strain rate exponent
$\triangle H$ = Activity energy
R = Gas constant
T_{abs} = Absolute temperature
$\bar{\sigma}$ = Flow stress
$\dot{\bar{\epsilon}}$ = Effective strain rate

Flow stress for aluminum alloys (Type 2)

$$\dot{\bar{\epsilon}} = A\bar{\sigma}^{n} e^{-\triangle H/RT_{abs}}$$

Where
A = Constant
n = Strain rate exponent
$\triangle H$ = Activity energy
R = Gas constant
T_{abs} = Absolute temperature $\bar{\sigma}$ = Flow stress
$\dot{\bar{\epsilon}}$ = Effective strain rate

Linear hardening

$$\bar{\sigma} = Y(T, A) + H(T, A)\bar{\epsilon}$$

Where
A = Atom content
T = Temperature
$\bar{\epsilon}$ = Effective plastic strain
$\bar{\sigma}$ = Flow stress
Y = Initial yield stress
H = Strain hardening constant

User defined flow stress routine
Please refer to Chapter 13 for a description of how to implement user defined flow stress routines.

Flow stress database
The DEFORM material database contains flow stress data for around 145 different materials. The flow stress data provided by the material database has a limited range in terms of temperature range and effective strain.

Warning: If a simulation condition of the material exceeds the bounds of the strain, strain rate or temperature defined in the tabular data, the program will extrapolate based on the last two data points which may lead to loss of accuracy.

Yield function type
This functionality supports anisotropy. There are three different types of yield functions available.

Von Mises

This is the default setting. This specifies an isotropic material model.

Hill's quadratic (FGHLMN)

This allows the user to specify anisotropic settings using the FGHLMN format of the Hill's quadratic method. (See Figure 2.2.10)

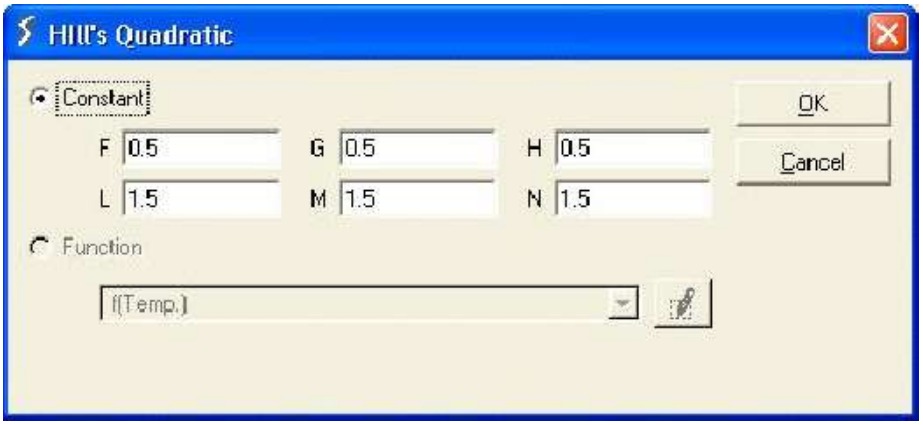


Figure 2.2.10: Hill's quadratic (FGHLMN) input screen.

Hill's quadratic (R value)

This allows the user to specify anisotropic settings using the R-value format of the Hill's quadratic method. (See Figure 2.2.11)

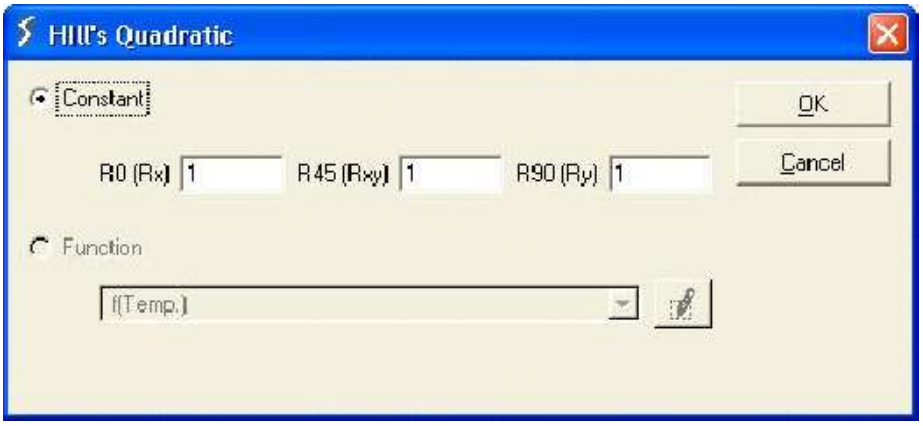


Figure 2.2.11: Hill's quadratic (R value) input screen.

Hardening rule (HDNRUL) [MIC]

Currently, two models for hardening are supported, kinematic and isotropic. For an isotropic model, as a material yields and plastically deforms, the yield surface expands uniformly or isotropically. Thus, the yield strain in all directions is the same. However, for a kinematic model, the yield surface shifts as the material

yields. The kinematic hardening model is required if the Bauschinger effect is to be modeled. This is valid only for the elasto-plastic objects under small deformation.

Creep (CREEP) [MIC]
Creep is defined as the time-dependent permanent deformation under stress that usually occurs at high temperatures. It is common in applications where the material undergoes cyclic loading or where stress relief is of interest. DEFORM only supports creep calculations for elasto-plastic objects.
The methods for defining creep in DEFORM are given below:

Perzyna's model

$$\dot{\bar{\epsilon}} = \gamma \left[\frac{\bar{\sigma}}{S} - 1 \right]^m$$

Where
γ = fluidity
$\bar{\sigma}$ = effective stress
S = Flow stress
m = Material parameter
$\dot{\bar{\epsilon}}$ = Effective strain rate
This model is known as Perzyna's model. It is a formulation for elastic-viscoplastic flow. In this method creep will not occur until the effective stress exceeds the yield strength of the material. If the effective stress is less than the flow stress, the resulting strain rate is zero.

Power law

$$\dot{\bar{\epsilon}} = \gamma \left[\frac{\bar{\sigma}}{S} \right]^m$$

Where
γ = fluidity
$\bar{\sigma}$ = effective stress
S = Flow stress
m = Material parameter
$\dot{\bar{\epsilon}}$ = Effective strain rate
This model is known as the power law. It is a very classical method for describing steady state or secondary creep.

Baily-Norton's model

$$\dot{\bar{\epsilon}} = Km\bar{\sigma}^n t^{m-1} + Q\bar{\sigma}^r$$

Where
$\bar{\sigma}$ = Effective stress
T_{abs} = Absolute temperature
K, n, m, Q, r = Constants
γ = fluidity
S = Flow stress
$\dot{\bar{\epsilon}}$ = Effective strain rate
This model is known as Baily-Norton's model. The user should make sure that K and Q are in the proper units so that the strain rate is defined as *second*$^{-1}$. The nodal temperature will be converted to absolute temperature inside the FEM engine.

Soderburg's model

$$\dot{\bar{\epsilon}} = K\bar{\sigma}^n e^{-C/T_{abs}}$$

Where
$\bar{\sigma}$ = Effective stress
T_{abs} = Absolute temperature
K ,n ,C = Constants
$\dot{\bar{\epsilon}}$ = Effective strain rate

Tabular data

This method is not currently available for this release

User Routines
This method is not currently available for this release.

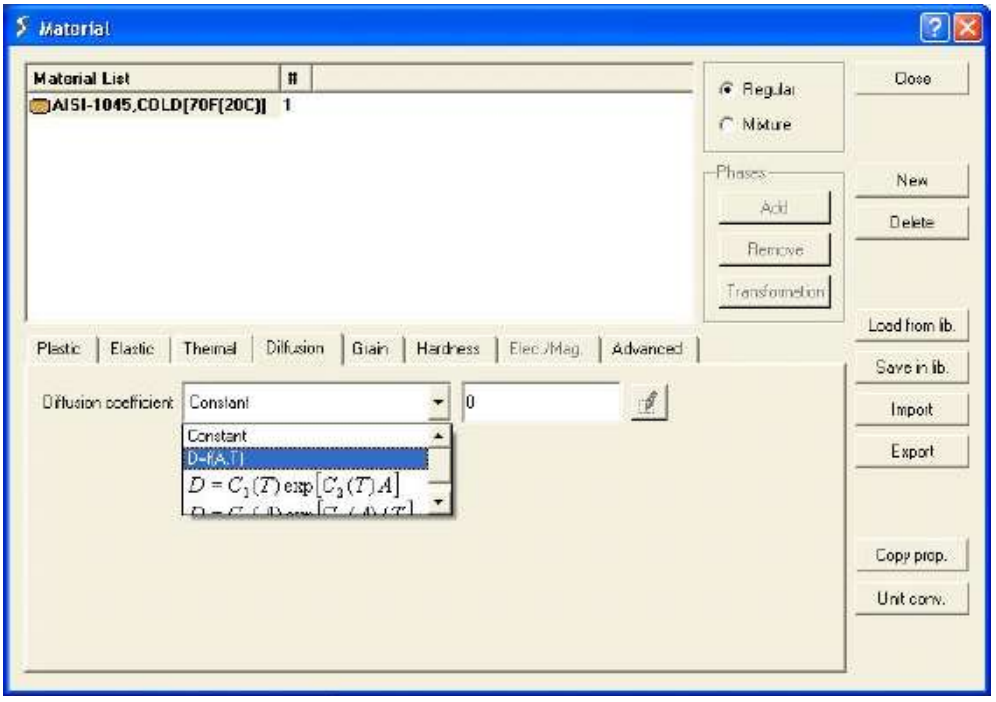


Figure 2.2.12: Diffusion data window.

2.2.5. Diffusion data

DEFORM allows the user to model the diffusion of the dominant atom (at this point carbon) in an object. The window for this is seen in Figure 2.2.12. The user only needs to specify the diffusion coefficient for the diffusion. For the simulation of carburizing process, normally performed before quenching, the Laplace equation is used for the diffusion model:

$$\frac{\partial C}{\partial t} = \frac{\partial}{\partial X}\left(D\frac{\partial C}{\partial X}\right)$$

Where
C is the carbon content, and D is the diffusion coefficient.

Note: Brick elements tend to produce nicer looking results than the tetrahedral elements since the mean diffusion distance is normally much smaller than the average element edge length. This will tend to make the tetrahedral results look somewhat patchy due to their generally uneven edge length.

Diffusion coefficient (DIFCOE)
The diffusion coefficient can be defined by the following methods:

Method 1
Constant value for diffusion coefficient.

Method 2
Diffusion coefficient is a function of atom content and temperature (Matrix format).

$$D=f(A,t)$$

Where
A is the atom content,
D is the diffusion coefficient and t is time.

Method 3
Diffusion coefficient is a function of temperature and atom content (Tabular format).

$$D=C_1(T)exp(C_2(T)A)$$

Where
D = Diffusion coefficient
A = Atom content
C_1 = Coefficient 1 which is a function of temperature
C_2 = Coefficient 2 which is a function of temperature
T = Absolute temperature

Method 4
Diffusion coefficient is a function of temperature and atom content (Tabular format).

$$D=C_1(A)exp(C_2(A)/T)$$

Where
D = Diffusion coefficient
A = Atom content
C_1 = Coefficient 1 which is a function of atom content
C_2 = Coefficient 2 which is a function of atom content
T = Absolute temperature

Hardening rule

This allows the user to specify whether an isotropic or kinematic hardening model is used.

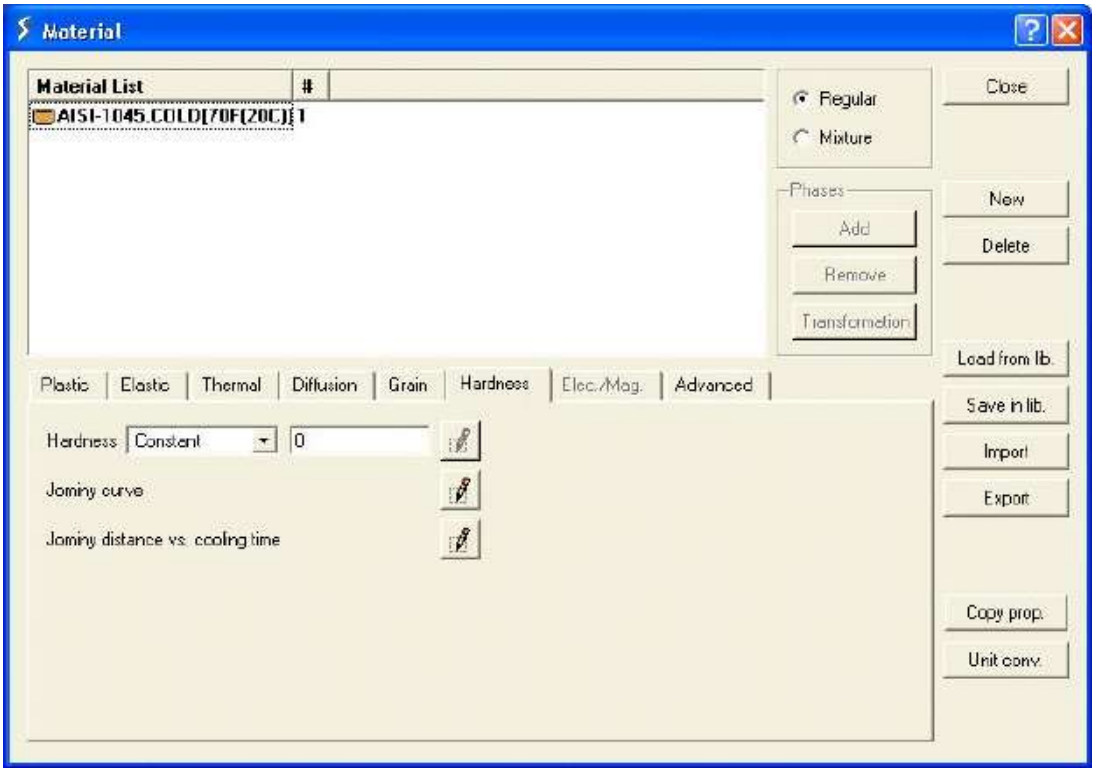


Figure 2.2.13: Hardness modeling data specification.

2.2.6. Hardness data [MIC]

There are two methods in which the hardness of a object may be determined after a cooling operation. The screen where this data is set is seen in Figure 2.2.13. The first method is by specifying the hardness of each phase (HDNPHA) in a mixture and DEFORM will use the mixture law to determine the hardness of each element. The second method is to use experimental results from the Jominy curve and cooling time vs. distance to determine the hardness during cooling.

The method of calculating the hardness can be specified for each object in the *Object, Properties* menu.

Hardness of each phase (HDNPHA)

The hardness of each phase (material group) can be specified. The hardness of each phase may be a constant or may vary with respect to atom content. The hardness of the object will be calculated based on the volume fraction of each phase in the element and on the hardness of each phase.

Jominy curve (JOMINY)
The hardness vs. distance for the Jominy test specimen must be specified here. This data is to be defined only for the mixture material (MSTMTR).

Cooling time (HDNTIM)
The cooling time vs. distance for the Jominy test specimen must be specified here. Using the JOMINY data and the HDNTIM data, DEFORM will calculate the hardness of the object during cooling.

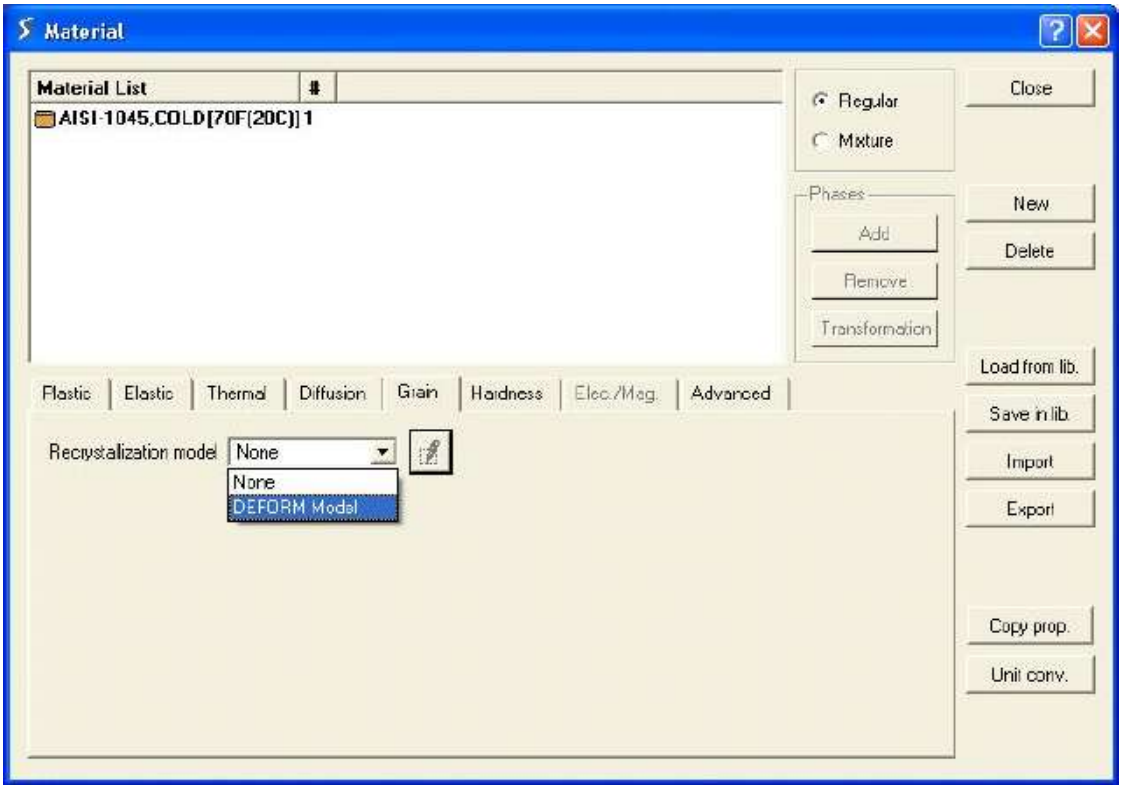


Figure 2.2.14: Recrystallization model setting window.

2.2.7. Grain growth/recrystallization model

Numerous phenomenological models have been published in the area of grain modeling, and controversies exist on the definitions of various recrystallization mechanisms. (See Figure 2.2.14)To accommodate these models, DEFORM has chosen the most popular definitions and generalized equation forms. In each time step, based on the time, local temperature, strain, strain rate, and evolution history, the mechanism of evolution is determined, and then the corresponding grain variables are computed and updated. (see Figure 2.2.13)

Definitions

Dynamic recrystallization: occurring during deformation and when the strain exceeds critical strain. The driving force is removal of dislocations.

Static recrystallization: occurring after deformation and when strain is less than critical strain. The driving force is removal of dislocations. The recrystallization begins in a nuclei-free environment.

Meta-dynamic recrystallization: occurring after deformation and when strain is greater than critical strain. The driving force is removal of dislocations. Because the strain has exceeded critical strain, recrystallization nuclei have formed in the material, so the recrystallization behaviors are different from without nuclei (static recrystallization).

Grain Growth: occurring before recrystallization begins or after recrystallization is completed. The driving force is the reduction of grain boundary energy.

Numerous phenomenological models have been published in the area of grain modeling, and controversies exist on the definitions of various recrystallization mechanisms. To accommodate these models, DEFORM has chosen the most popular definitions and generalized equation forms. In each time step, based on the time, local temperature, strain, strain rate, and evolution history, the mechanism of evolution is determined, and then the corresponding grain variables are computed and updated.

Dynamic Recrystallization

The dynamic recrystallization is a function of strain, strain rate, temperature, and initial grain size, which change in time. It is very difficult to model dynamic recrystallization concurrently during forming. Instead, the dynamic recrystallization is computed in the step immediately after the deformation stops. Average temperatures, strain rate of the deformation period are used as inputs of the equations.

1. Activation Criteria

   The onset of DRX usually occurs at a critical stain ε_c.

   $$\varepsilon_c = a_2 \varepsilon_p$$

   Where ε_p denotes the stain corresponding to the flow stress maximum:

   $$\varepsilon_p = a_1 d_o^{n_1} \dot{\varepsilon}^{m_1} \exp\left(Q_1 / RT\right) + c_1$$

2. Kinetics

The Avrami equation is used to describe the relation between the dynamically recrystallized fraction *X* and the effective strain.

$$X_{drex} = 1 - \exp\left[-\beta_d \left(\frac{\varepsilon - a_{10}\varepsilon_p}{\varepsilon_{0.5}}\right)^{k_d}\right]$$

Where $\varepsilon_{0.5}$ denotes the strain for 50% recrystallization:

$$\varepsilon_{0.5} = a_5 d_o^{n_5} \dot{\varepsilon}^{m_5} \exp(Q_5 / RT) + c_5$$

3. Grain Size

The recrystallized grain size is expressed as a function of initial grain size, strain, strain rate, and temperature

$$d_{rex} = a_8 d_0^{h_8} \varepsilon^{n_8} \dot{\varepsilon}^{m_8} \exp(Q_8 / RT) + c_8$$ (if $d_{rex} \geq d_0$ then $d_{rex}=d_0$)

Static Recrystallization

When deformation stops, the strain rate and critical strain are used to determine whether static or meta-dynamic recrystallization should be activated. The static and metal-dynamic recrystallization is terminated when this element starts to deform again.

1) Activation Criteria

When strain rate is less than $\dot{\varepsilon}_{ss}$, static recrystallization occurs after deformation.

$$\dot{\varepsilon}_{ss} = A \exp(b_1 - b_2 d_o - Q_2 / T)$$

2) Kinetics

The model for recrystallization kinetics is based on the modified Avrami equation.

$$X_{srex} = 1 - \exp\left[-\beta_s \left(\frac{t}{t_{0.5}}\right)^{k_s}\right]$$

Where $t_{0.5}$ is an empirical time constant for 50% recrystallization:

$$t_{0.5} = a_3 d^{h_3} \varepsilon^{n_3} \dot{\varepsilon}^{m_3} \exp(Q_3 / RT)$$

3) Grain Size

The recrystallized grain size is expressed as a function of initial grain size, strain, strain rate, and temperature

$$d_{rex} = a_6 d_0^{h_6} \varepsilon^{n_6} \dot{\varepsilon}^{m_6} \exp(Q_6 / RT) + c_6$$ (if $d_{rex} \geq d_0$ then $d_{rex}=d_0$)

Meta-dynamic Recrystallization

Meta-dynamic recrystallization is similar to static recrystallization but with different activation criteria and material constants.

1. Activation Criteria

   When strain rate is greater than $\dot{\varepsilon}_{ss}$ (see equation (1)), meta-dynamic recrystallization occurs after deformation.

2. Kinetics

   The model for recrystallization kinetics is based on the modified Avrami equation.

   $$X_{mdrex} = 1 - \exp\left[-\beta_{md}\left(\frac{t}{t_{0.5}}\right)^{k_{md}}\right]$$

   Where $t_{0.5}$ is an empirical time constant for 50% recrystallization:

   $$t_{0.5} = a_4 d^{k_4} \varepsilon^{n_4} \dot{\varepsilon}^{m_4} \exp(Q_4 / RT)$$

3. Grain Size

The recrystallized grain size is expressed as a function of initial grain size, strain, strain rate, and temperature

$$d_{rex} = a_7 d_0^{k_7} \varepsilon^{n_7} \dot{\varepsilon}^{m_7} \exp(Q_7 / RT) + c_7 \quad (\text{if } d_{rex} \geq d_0 \text{ then } d_{rex}=d_0)$$

Grain Growth

Grain growth takes place before recrystallization start or after recrystallization finishes.

The kinetics is described by equation:

$$d_\gamma = \left[d_{rex}^{\ m} + a_9 t \exp\left(-\frac{Q_9}{RT}\right)\right]^{1/m}$$

Where d_{gg} denotes the grain size after growth, a_9 and m are materials constant, and Q_8 is activation energy.

Retained Strain

When recrystallization of a certain type is incomplete, the retained strain available for following another type of recrystallization can described by a uniform softening method

$$\varepsilon_i = (1.0 - \lambda X_{rex})\varepsilon_{i-1}$$

Temperature Limit

The temperature limit is the lower bound of all grain evolution mechanisms. Below this temperature, no grain evolution occurs.

Average Grain Size

The mixture law was employed to calculate the recrystallized grain size for incomplete recrystallization,

$$d = X_{rex} \cdot d_{rex} + (1 - X_{rex}) \cdot d_0$$

Model Dependency on Temperature and Strain Rate

DEFORM allows different constants and coefficients for the equations at different temperatures or strain rates. The data are linearly interpolated.

NOMENCLATURE

T - Temperature, K

R - Gas constant

t - Time

d_0 - Initial grain size

d_{rex} - Recrystallized grain size

ε - Effective strain

ε_c - Critical strain

ε_p - Peak strain

$\varepsilon_{0.5}$ - Strain for 50% recrystallization

$\dot{\bar{\varepsilon}}$ - Effective strain rate

$t_{0.5}$ - Time for 50% recrystallization

Z - Zener Holloman parameter

a_{1-10} - Material data

b_{1-2} - Material data

c_{1-8} - Material data

n_{1-8} - Material data

m_{1-8} - Material data

Q_{1-8} - Material data

β_d, β_{md}, β_s - Material data

k_d, k_{md}, k_s - Material data

λ - Material data

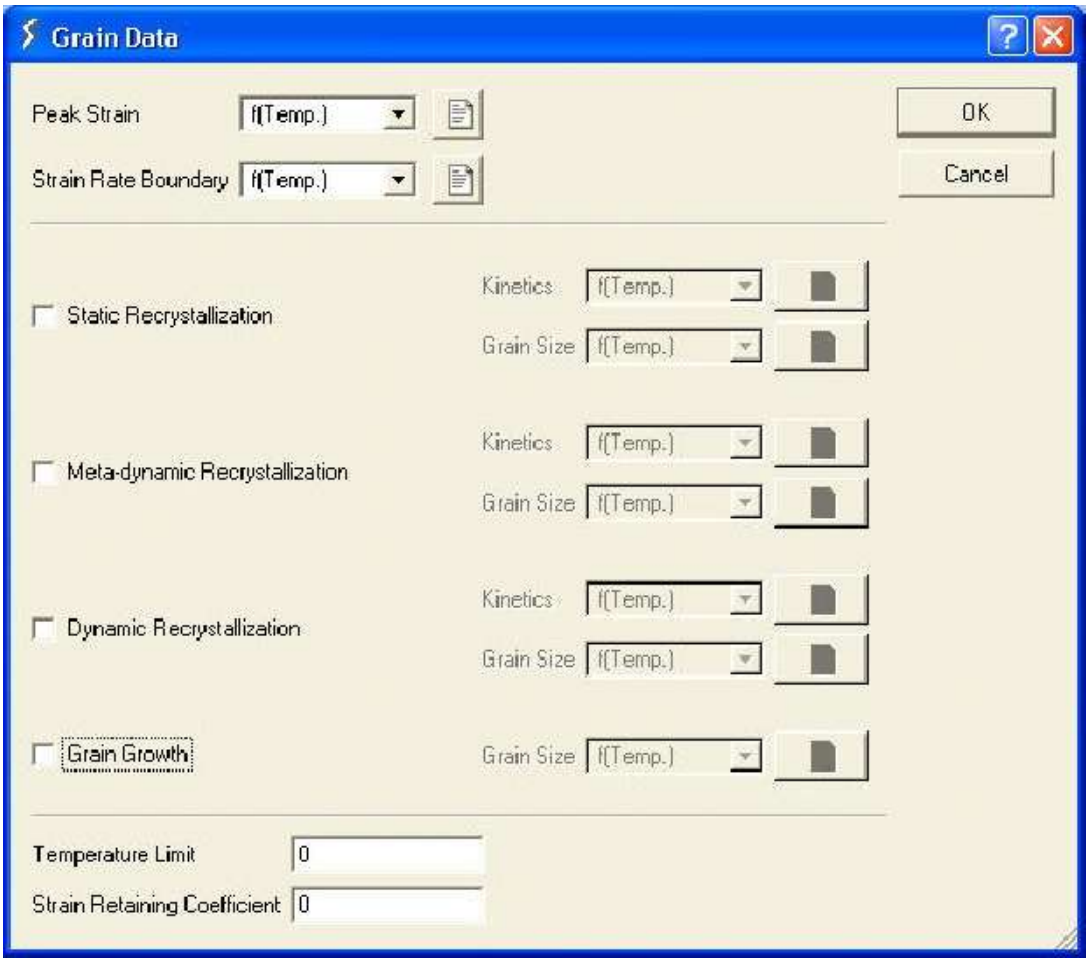


Figure 2.2.15: Grain growth and recrystallization settings window.

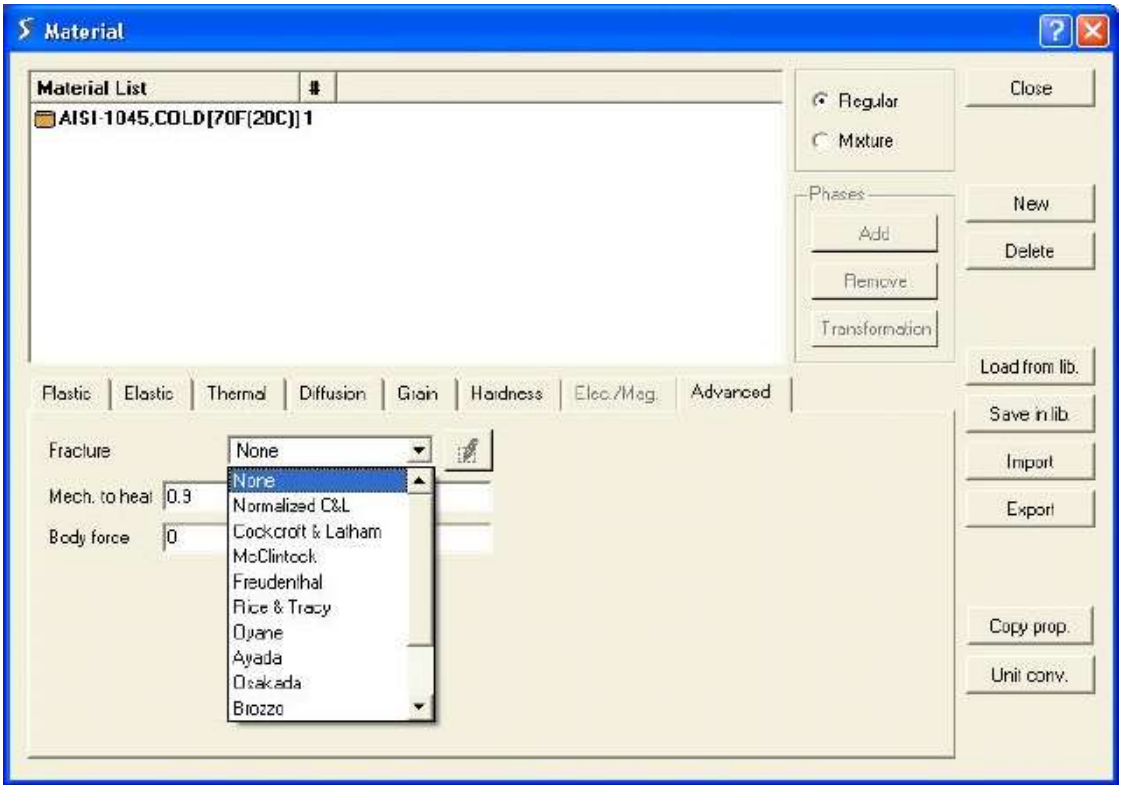


Figure 2.2.16: Advanced material settings.

2.2.8. Advanced material properties

Figure 2.2.16 shows the advanced material properties available for special applications.

Fracture Data (FRCMOD)
FRCMOD specifies the damage model that one wishes to use for damage calculation. The Normalized Cockcroft & Latham damage model is the only model currently supported in DEFORM3D

Mechanical Work to Heat (FRAE2H)
Mechanical work to heat specifies the fraction of mechanical work converted to heat. The conversion fraction would typically be 0.9 to 0.95. The default value is 0.9 and unless the user has a good feel for this value, this value should not be changed.

2.2.9. Material data requirements

Guidelines

1. An isothermal forming problem with rigid dies and a rigid-viscoplastic work piece

	Work piece	Dies
Material Data		
Flow Stress	X	
Young's Modulus		
Poisons Ratio		
Thermal Expansion		
Heat Capacity		
Conductivity		
Emissivity		

2. A Non-isothermal forming problem with rigid dies and a rigid-viscoplastic work piece

	Work piece	Dies
Material Data		
Flow Stress	X	
Young's Modulus		
Poisons Ratio		
Thermal Expansion		
Heat Capacity	X	X
Conductivity	X	X
Emissivity	X	X

3. Heat transfer analysis

	Work piece	Dies
Material Data		
Flow Stress		
Young's Modulus		
Poisons Ratio		
Thermal Expansion		
Heat Capacity	X	X
Conductivity	X	X
Emissivity	X	X

4. Coupled analysis non-isothermal with thermal expansion Elastic dies and elasto-plastic work piece

	Work piece	Dies
Material Data		
Flow Stress	X	
Young's Modulus	X	X
Poisons Ratio	X	X
Thermal Expansion	X	X
Heat Capacity	X	X

Conductivity	X	X
Emissivity	X	X

5. Decoupled die stress analysis isothermal

	Work piece	Dies
Material Data		
Flow Stress	X	
Young's Modulus		X
Poisons Ratio		X
Thermal Expansion		
Heat Capacity		
Conductivity		
Emissivity		

6. Decoupled die stress analysis non-isothermal

	Work piece	Dies
Material Data		
Flow Stress	X	
Young's Modulus		X
Poisons Ratio		X
Thermal Expansion		X
Heat Capacity	X	X
Conductivity	X	X
Emissivity	X	X

2.3. Inter Material Data

The purpose of inter-material data is to define the relationships between the phases of a mixture. As defined in the material data, a mixture is defined by the user as a set of phases. The relationships between the phases are defined in terms of the following transformation characteristics:

- Transformation kinetics model
- Latent heat of transformation
- Transformation induced volume change
- Transformation plasticity

The purpose of this section is to give the user an understanding of how to properly define a transformation relationship between two phases. This section will explain in how DEFORM handles the above concepts.

Transformation is a crucial concept in metal forming and heat treatment. Figure 2.3.1 illustrates the coupling between temperature, deformation, transformation, and carbon content. Transformation is modeled by defining the volume fraction for each possible phase in each element of a meshed object. For low carbon steel object, each element may contain different volume fraction of martensite, bainite, pearlite, or austenite. Each phase is defined by its own set of material properties. These material properties define the plastic behavior of the phase, the thermal properties of the phase, and possibly (if using an elastic-plastic material) the elastic properties of the phase.

The relationship between the transformations from one phase to another is defined by the inter-material data. This relationship is defined in terms of a kinetics model (in order to determine rate of phase transformation) and a few relational properties such as latent heat and volume change.

2.3.1. Transformation relation (PHASTF)

In DEFORM, the manner in which transformation is defined is in terms of transformation relationships. The basic unit for transformation relationships is phases. Phases can be grouped together to define a mixture. A mixture corresponds to a material such as AISI-1045 or Ti-6Al-4V. The phases correspond to Austenite, Bainite, Pearlite, Martensite, or *alpha* and to any other phase that is defined. For example, in the case of low carbon steel, austenite has a relationship to pearlite since austenite may form pearlite upon the proper cooling conditions. Also, upon proper heating conditions, pearlite can convert to austenite. Thus, in DEFORM, to specify the relationship of austenite converting to pearlite, one need select the austenite as Material 1 and the pearlite as Material 2 and then click the Phase 1 $\Rightarrow$ Phase 2 relationship. This will then allow the user to define the kinetics of the transformation, the latent heat of the transformation, volume change of the transformation, etc.

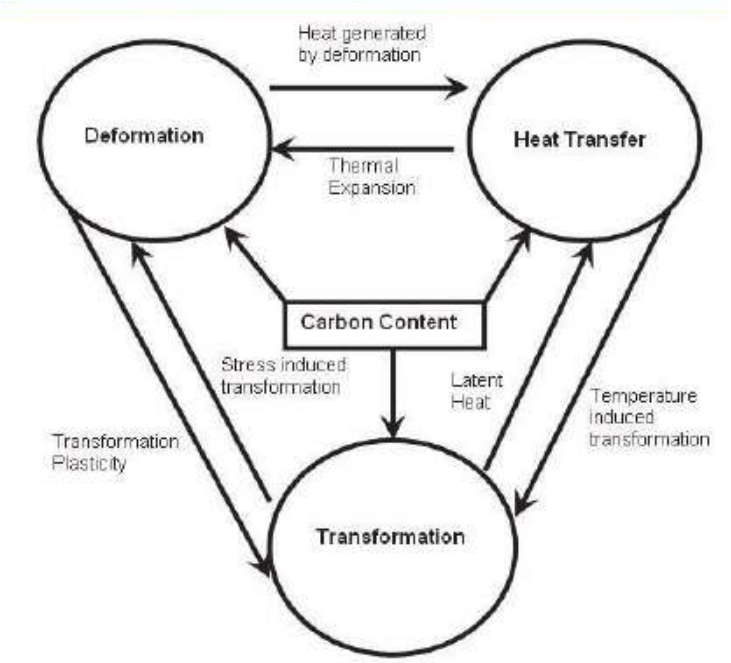


Figure 2.3.1: Relationships between various modules within DEFORM.

2.3.2. Kinetics model (TTTD)

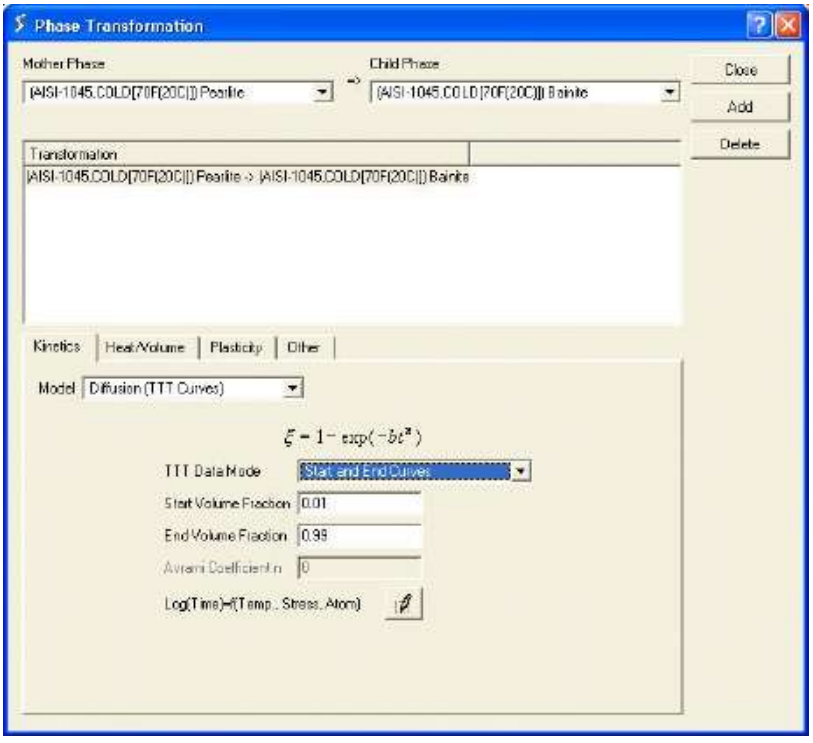


Figure 2.3.2: Transformation kinetics models.

A kinetics model is a function that defines the conditions and manner in which one phase may transform into another. The amount of data required is often considerable as in the case for a full TTT diagram, so unless necessary, often using coefficients for a function can suffice. A model defines one relationship only so many relationships are required for such cases as steel where many phases can be produced. (see Figure 2.3.2)There are two classifications for kinetics models, diffusion-type transformations and diffusion less-type transformations. The system is designed for both ferrous and nonferrous metals. Using carbon steel as an example, the austenite-ferrite and austenite-pearlite structure changes and vice versa are governed by the diffusion type transformation. The transformation is driven by the diffusion processes depending n the temperature, stress history, and carbon content. The diffusion less transformation from austenite to martensite involves a shear process which depends on the temperature, stress, and carbon content.

Diffusion type TTT table form (Temp, Stress, Atom)

This type defines a TTT diagram whose independent variables are average element temperature, effective stress and dominant atom content. In the case of steel, dominant atom content is the weight percentage of carbon in the metal at each element. Using tabular data, DEFORM solves for the coefficients of an Avrami equation, which has the form

$$\xi = 1 - exp\ (-kt^n)$$

Where ξ is the volume fraction transformed,
t is the time and
k and n are constants (n being the Avrami number).
In terms of TTT data, two curves are required in order to solve for k and n. If only one curve is input to DEFORM, the user must provide the Avrami exponent n.

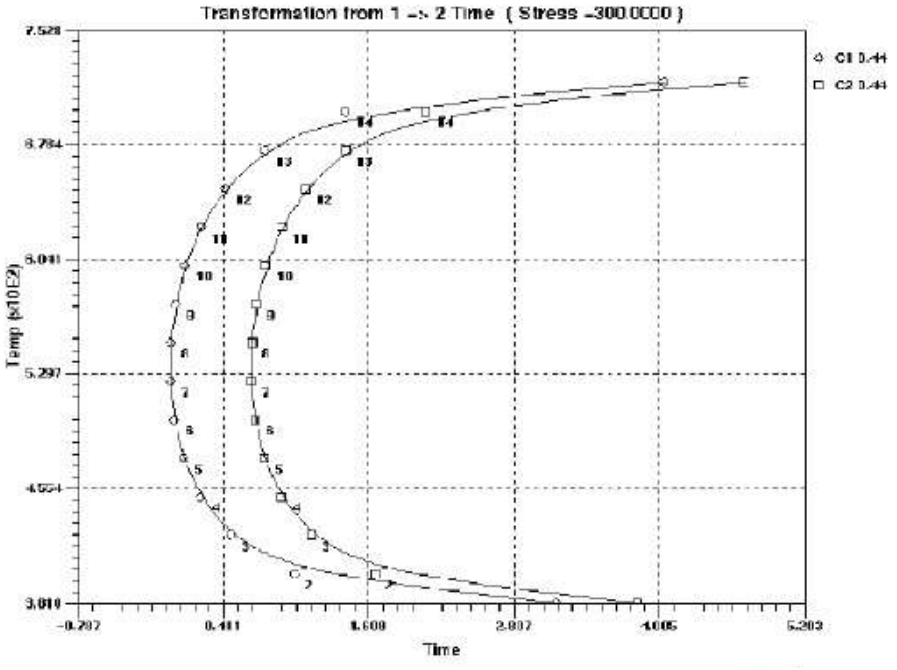


Figure 2.3.3: TTT curve data.

Figure 2.3.3 shows an example TTT diagram in DEFORM. In the case below, two curves are used to define the volume fraction transformed for a given dominant atom content. Each curve corresponds to the appropriate specified volume fraction (Curve1 VF or Curve2 VF).

Martensitic type (Tms, Tm 50 Table form)

The transformation start and 50 % level temperature are entered in table format as a function of carbon content and stress levels.

Diffusion type (function)
Volume fraction is represented by the Avrami equation as follows:

$$\xi_I = 1 - exp(-f_T(T) f_s(\sigma_m) f_c(C) t^n)$$

Where

$f_T(T)$, $f_S(\sigma_m)$ and $f_C(C)$ are the functions of temperature T, σ_m is the mean stress and C is the carbon content. The exponent n depends on the transformation and $f_T(T)$ can be expressed the following simplified formula.

$$f_T(T) = A_{T1}\left(\frac{T - A_{T2}}{A_{T3}}\right)^{A_{T4}}\left(\frac{A_{T5} - T}{A_{T6}}\right)^{A_{T7}}$$

Where the coefficients from A_{T1} to A_{T7} are determined by using 50% transformed line of TTT diagram. $f_S(\sigma_m)$ and $f_C(C)$ describes the stress and carbon content dependency of transformation, respectively as follows:

$$f_S(\sigma_m) = exp(A_s\sigma_m)$$

$f_C(C)=exp(A_{C1}(C-A_{C2}))$

The coefficients A_S is specified according to the stress dependency of TTT curves, A_{C1} and A_{C2} are determined by carbon content dependency.

Diffusion type (function and table)
This model is not available in the current release of DEFORM.

Martensitic type (function)
The volume fraction of diffusionless-type (martensite) transformation depended on temperature, stress and carbon content is governed by Magee's equation as follows:

$$\xi_M = 1 - exp(\psi_1 T + \psi_2(C - C_0) + \psi_{31}\sigma_m + \psi_{32}\bar{\sigma} + \psi_4)$$

Here, σ_m is the mean stress, $\bar{\sigma}$ is the effective stress. When the martensite transformation start temperatures under carburized conditions and applied stress are given, ψ_2/ψ_1, ψ_{31}/ψ_1 and ψ_{32}/ψ_1 can be determined, ψ_1 and ψ_4 and are identified, if temperatures for martensite-start T_{MS} and for 50% martensite T_{M50} at $\xi_M = 0$ and $\xi_M = 0.5$ are provided respectively.

Diffusion type (simplified)

SCREEN: Inter Material
A simplified Diffusion function is defined by a function of the following form:

$$\xi_J = 1 - exp\left[A\left(\frac{T - T_S}{T_E - T_S}\right)^D\right]$$

Where
T = average element temperature.
T_S = starting temperature of the transformation.
T_E = ending temperature of the transformation.
This formula is a good first approximation for a diffusion-based transformation. The coefficients can be obtained using dilatation-temperature diagrams.

Diffusion type (recrystallization)

The volume fraction of recrystallization is usually defined by the equation including the time for 50% recrystallization as follows:

$$\xi_J = 1 - exp\left[-b\left(\frac{t}{t_{0.5}}\right)^n\right]$$

Where, b is material constant and n is the exponent whose value depends upon the underlying mechanisms and $t_{0.5}$ is the time for 50% recrystallization;

$$t_{0.5} = a\epsilon^m d_0^n exp(\frac{Q}{RT})$$

where a, m, and n are material constants, Q is activation energy, R gas constant, T absolute temperature, and ϵ is a prior plastic strain obtained after an operation of forming and d_0 is an initial grain diameter specified as object data. This model is not currently available for the current release of DEFORM.

Melting and solidification type
This model is not available for the current release of DEFORM.

User Routine
This model is not available for the current release of DEFORM.

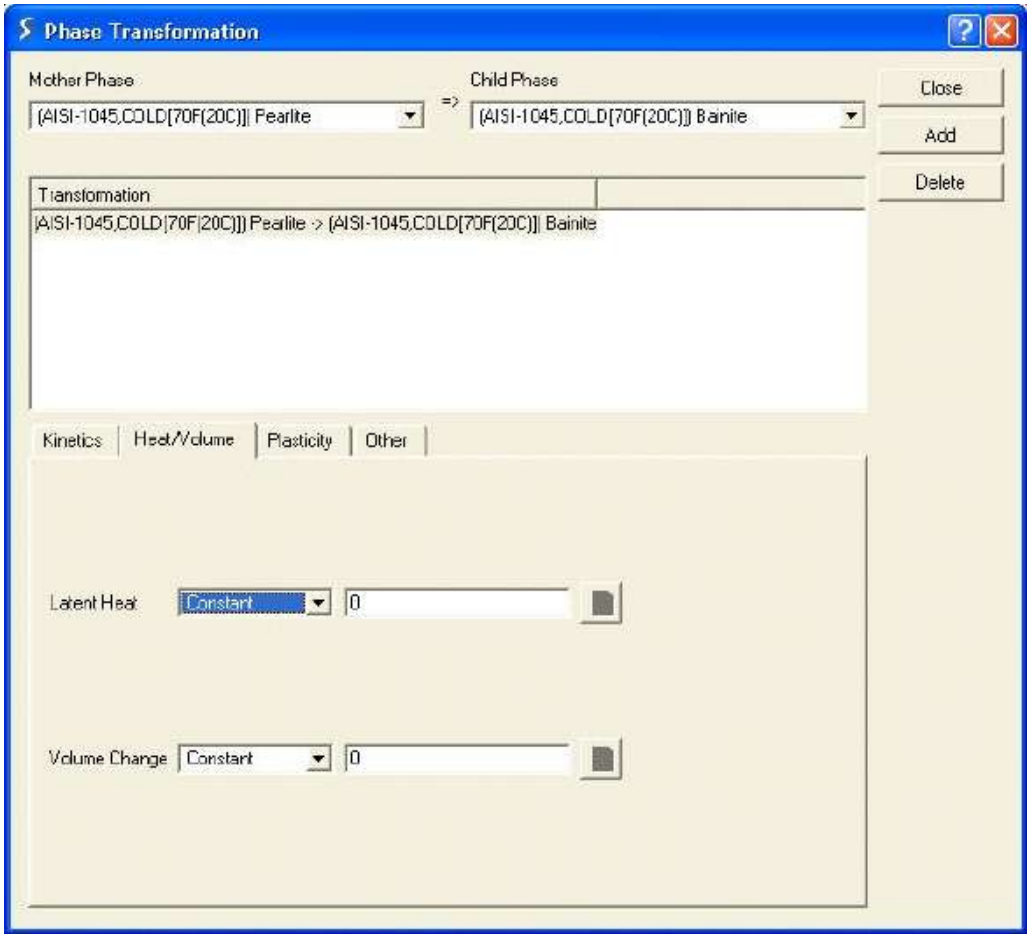


Figure 2.3.4: Latent heat and transformation-induced volume change data.

2.3.3. Latent heat (PHASLH)

Latent heat accounts for the net energy gain or loss when a phase change occurs from one phase to another. Latent heat may be a constant value, a function of either temperature or a function of the dominant atom content. The energy release due to the latent heat can prolong the time of transformation. A positive sign on the latent heat value means that the transformation acts as a heat source and a negative sign means that the transformation acts as a heat sink. (See Figure 2.3.4)

2.3.4. Transformation induced volume change (PHASVL)

Volume change may be the result of a phase transformation. This volume change may induce stresses in the transforming object and will certainly affect the final dimensions after processing. The volume change due to transformation is induced by a change in the lattice structure of a metal. The transformation strain is used mainly to account for the structure change during the transformation and is in the form of:

$$\dot{\epsilon}_{ij}^{Tr} = \Sigma \beta_{IJ} \dot{\xi}_J \delta_{ij}$$

Where β_{IJ} is the fractional length change due to transformation from phase I to phase J, $\dot{\xi}_J$ is transformation volume fraction rate, δ_{ij} is the Kronecker delta and $\dot{\epsilon}_{ij}^{Tr}$ is the transformation strain rate. A positive sign in the volume change means there is an increase in volume change and a negative sign means there is a decrease in volume over the transformation.

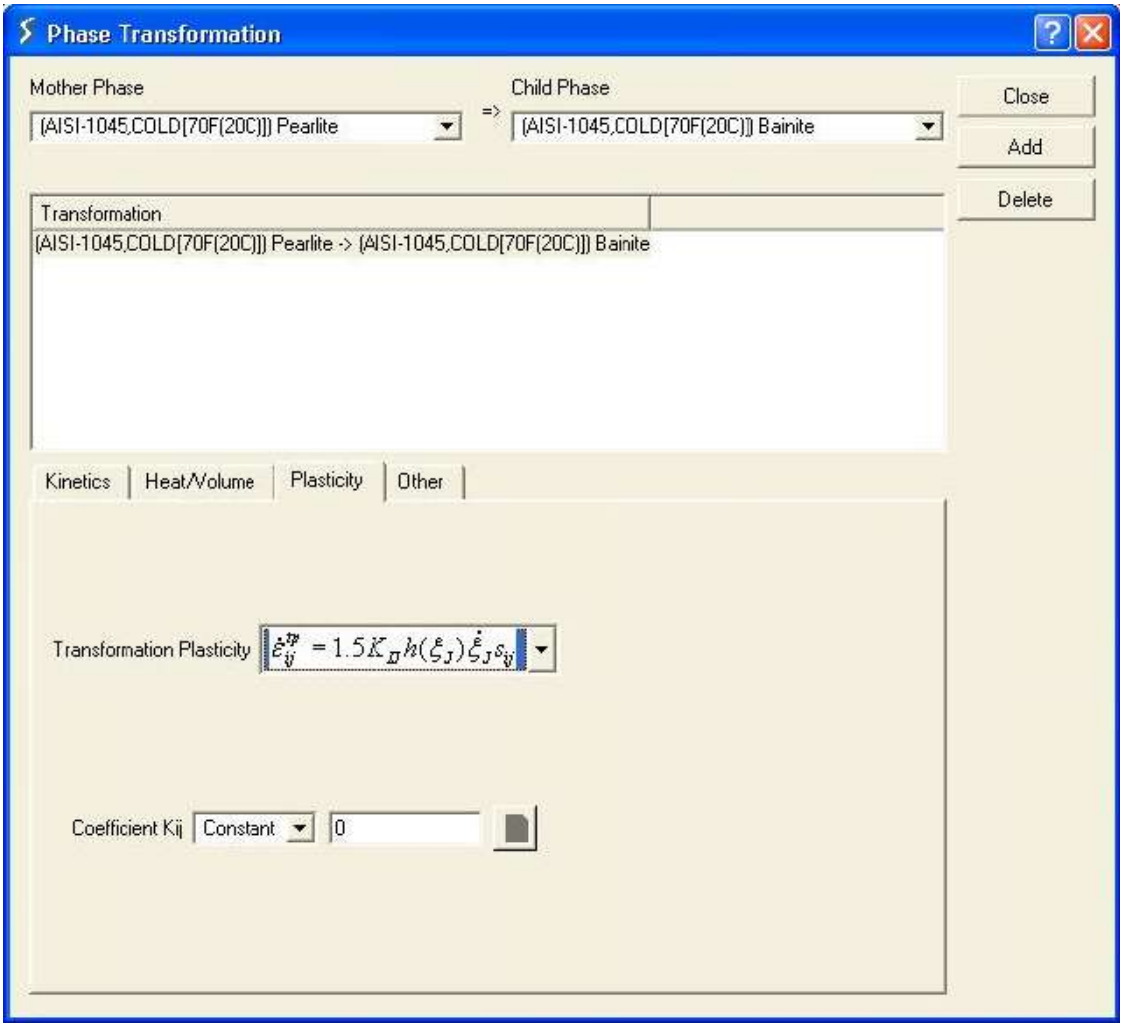


Figure 2.3.5: Transformation-induced plasticity data.

2.3.5. Transformation plasticity (TRNSFP)

As a material undergoes transformation, it will plastically deform at a stress lower than the flow stress. This phenomenon is known as Transformation Plasticity. The change of the dimensions of a part due to transformation plasticity occur in combination with the dimension changes due to transformation induced volume change. In DEFORM, the equation for transformation plasticity is as follows: (see Figure 2.3.5)

$$\dot{\epsilon}_{ij}^{Tp} = \frac{3}{2} K_{IJ} h(\xi_J) \dot{\xi}_J s_{ij}$$

Where

$\dot{\epsilon}_{ij}^{Tp}$ = Transformation plasticity strain tensor.

K_{IJ} = Transformation plasticity coefficient from phase I to phase J

$\dot{\xi}_J$ = Volume fraction rate

$h(\xi_J) = 2(1 - \xi_J)$

s_{ij} = Deviatoric stress tensor.

The only data that the user needs to provide for this relationship is the transformation plasticity coefficient. The other terms are automatically calculated by DEFORM. The transformation plasticity coefficient may be a function of temperature.

A general range for K_{IJ} for steel is given below,

- austenite - ferrite, pearlite or bainite (4 - 13 *10^{-5} /MPa)
- austenite - martensite (5 - 21 * 10^{-5} /MPa)
- ferrite & pearlite - austenite (6 - 21 *10^{-5} /MPa)

2.3.6. Other Transformation Data

Thermal direction gives the simulation a bit more information so transformation does not errantly generate volume fraction. For example, when heating steel from room temperature to austenizing temperature, any bainite will be converted, over time, to austenite. During the heating, austenite may be converted back to bainite since it may be defined as a possibility. This definition prevents this. It is recommended to use this sparingly. (see Figure 2.3.6

Equilibrium volume fraction defines the maximum amount of a phase volume fraction generation during an isothermal condition.

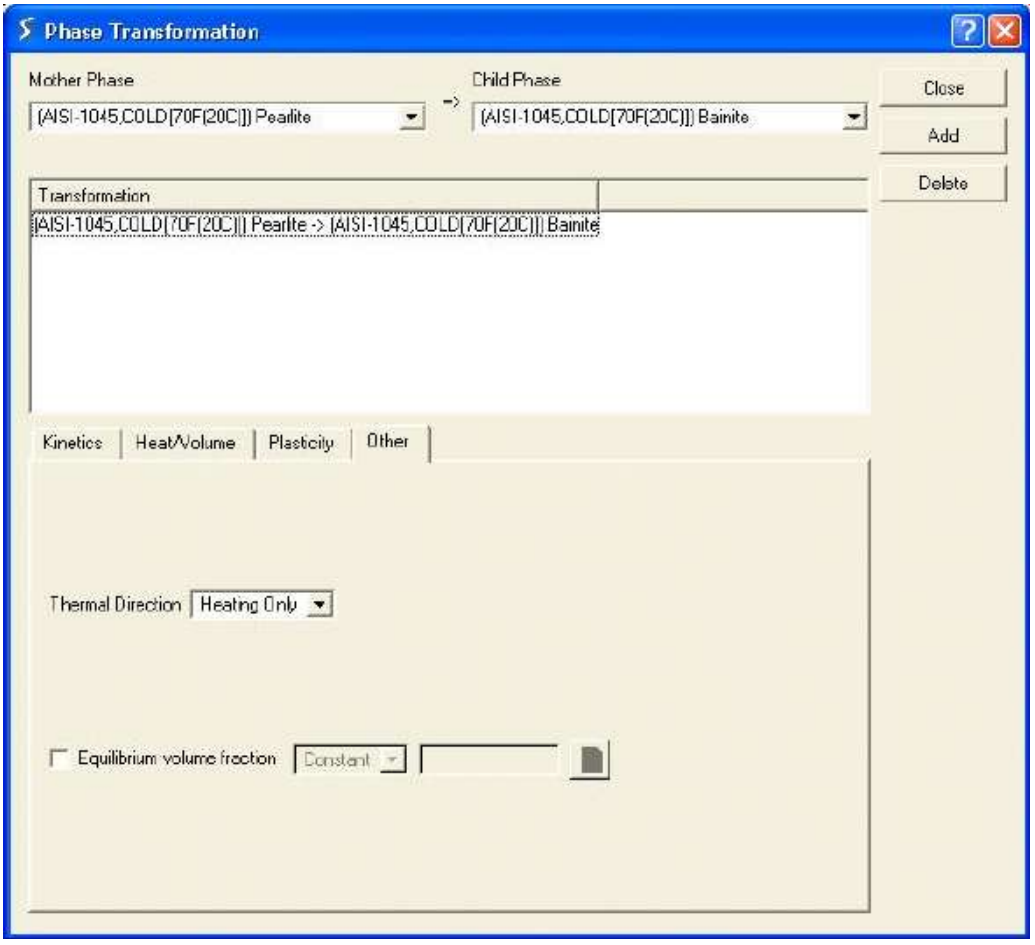


Figure 2.3.6: Other phase transformation data

2.4 Object Definition

The objects display list in the preprocessor shows all the currently active objects (See Figure 2.4.1). The “active object” can be controlled by selecting an object in the objects display list. Once an object is selected, the object properties window contains all object specific data such as the geometry, mesh, boundary conditions, movement, initial conditions, and object specific numerical properties for the object “active object”.

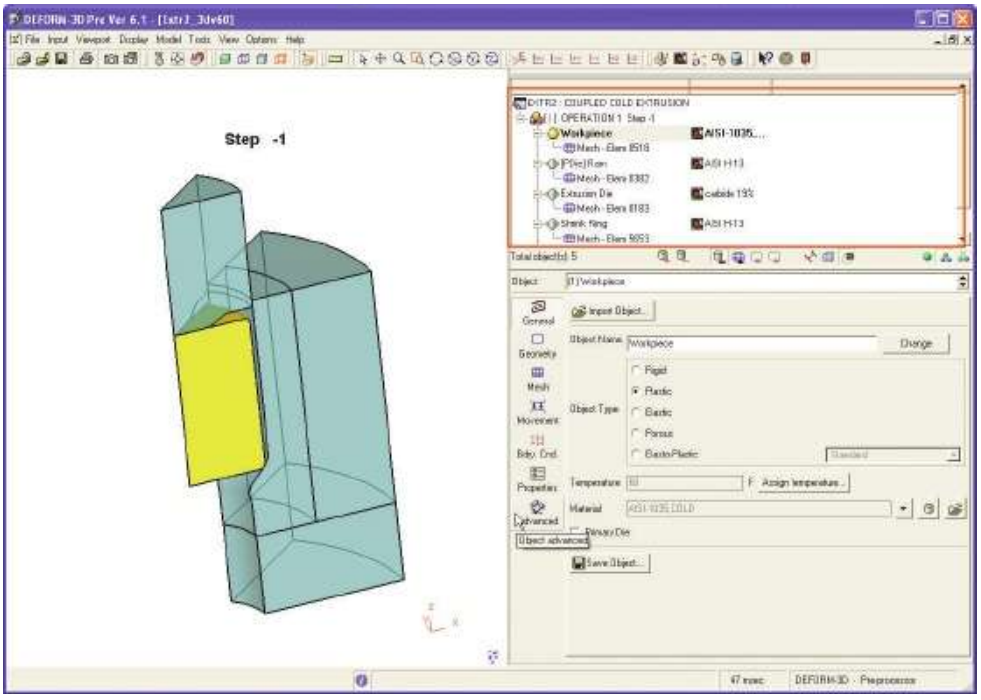


Figure 2.4.1: Preprocessor with the object list with a red box.

2.4.1. Adding, deleting objects

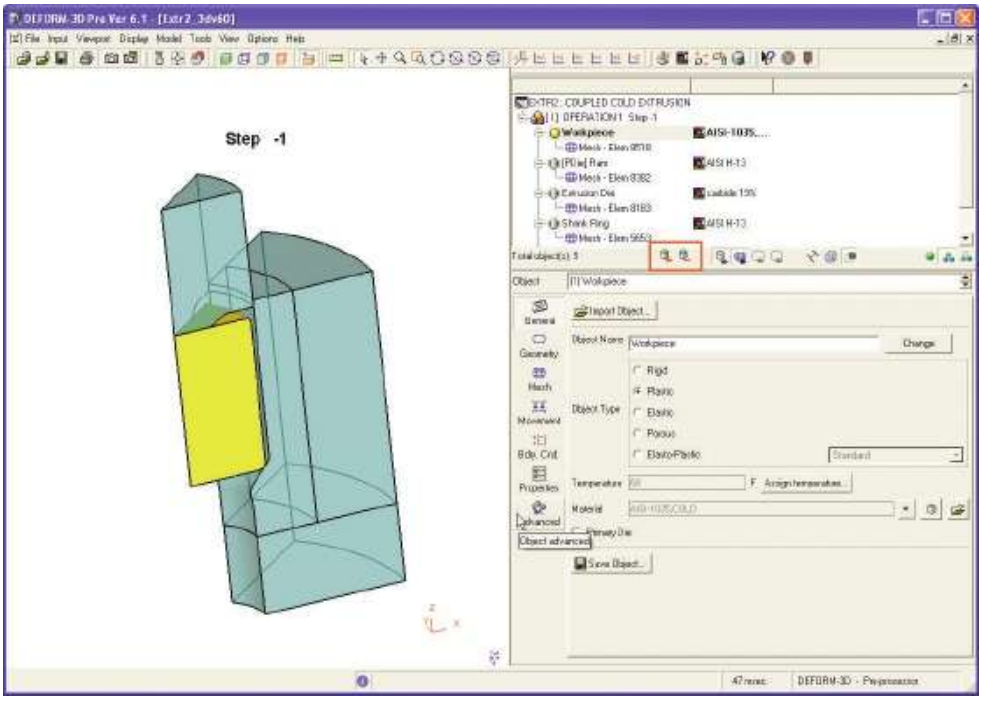


Figure 2.4.2: Insert and Delete object buttons in a red rectangle.

To add an object to the list of objects, click on the *Insert Object* button. This will insert a new object into the first available object number.
To delete an object, select the appropriate object and press the *Delete Object* button (See Figure 2.4.2). This will delete all entries associated with the object, (see Figure 2.4.3 including movement controls, inter-object boundary conditions, friction and heat transfer data, etc.
Note: To replace an object geometry definition without deleting movement controls and inter-object relationships, it is possible to overwrite the object geometry from the geometry window. This is useful for changing die geometries when performing two or more deformation operations on the same work piece. When redefining an object in this manner, it is extremely important to initialize and regenerate inter-object boundary conditions. It may also be necessary to reset the stroke definition in Movement controls.

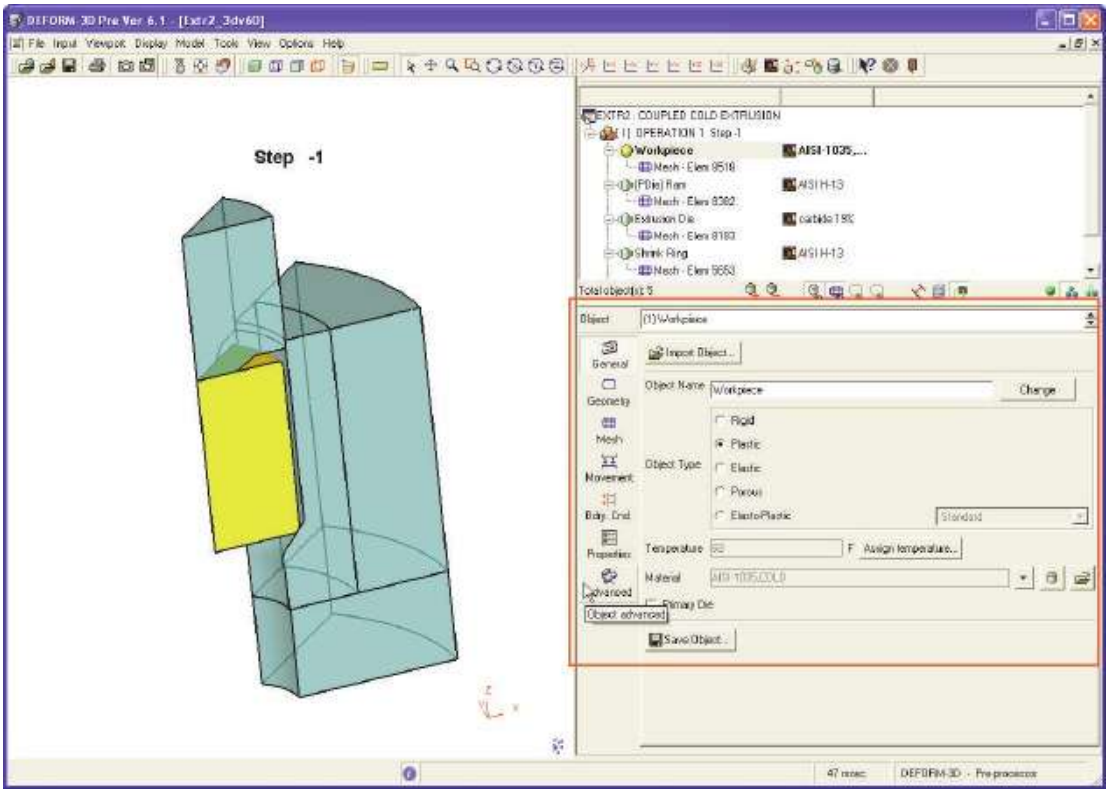


Figure 2.4.3: Object general properties in a red box.

2.4.2. Object name (OBJNAM)

The work piece and each piece of tooling must be identified as a unique object and assigned an object number and name. The object name is a string of up to 64 characters. It is highly recommended that it be set to something meaningful (e.g. punch, die, work piece). (See Figure 2.4.3).

2.4.3. Primary Die (PDIE)

The primary die specifies the primary object for the simulation. The primary object is usually assigned to the object most closely controlled by the forming machinery. For example, the die attached to the ram of a mechanical press would be designated as the primary die. Characteristics of the primary die can be used to control various aspects of a simulation including:

1. Simulation time step size (DSMAX)
2. Object movement (MOVCTL)
3. Simulation termination criteria (SMAX, VMIN, LMAX)

The primary die is defined in using a checkbox (See Figure 2.4.3). Only one object can be defined as the primary die.

2.4.4. Object type (OBJTYP)

The object type defines if and how deformation is modeled for each individual object in a DEFORM problem.

Rigid

Rigid objects are modeled as non-deformable materials. In the deformation analysis, the object is represented by the geometric profile (DIEGEO). Deformation solution data available for rigid objects include object stroke, load, and velocity. The mesh for the rigid object is used only for thermal, transformation, and diffusion calculations.

Applications:

When used to model tooling, increases simulation speed (over elastic tooling) by reducing the number of deformable objects, and hence the number of equations which must be solved. Negligible loss of accuracy for typical simulations where the tools have a much higher yield stress than the work piece.

Limitations:

Stress and deflection data for the dies is not available during deformation. This data can be obtained at selected single steps by performing a single step die stress analysis

Elastic

The elastic material behavior is specified with Young's modulus (YOUNG) and Poisson's ratio (POISON). Elastic objects are used if the knowledge of the tooling stress and deflection are important throughout the process. If maximum stress or deflection information is required for die stress, it is recommended that rigid dies be used for the deformation simulation, and then a single step die stress simulation be used. Refer to the die stress tutorials in the online help for more

information. At this time a fully coupled elastic tool, plastic work piece analysis is not recommended.

Applications:

When used to model tooling, the elastic model can provide information on tool stress and deflection. Useful in rare situations when tooling deflection can have a significant influence on the shape of the part.

Limitations:

If yield stress for the tooling is exceeded, stress and deflection results will be incorrect. However, in most cases, if tooling yield stress is exceeded, this represents an unacceptable situation, and tooling deformation beyond yield is not useful. It is good practice to check stresses in simulations with elastic tooling to ensure that this situation is not violated.

Plastic

Plastic objects are modeled as rigid-plastic or rigid-viscoplastic material depending on characteristics of materials. The formulation assumes that the material stress increases linearly with strain rate until a threshold strain rate, referred to as the limiting strain rate (LMTSTR). The material deforms plastically beyond the limiting strain rate. The plastic material behavior of the object is specified with a material flow stress function or flow stress data (FSTRES).

Applications:

When used to model work piece, provides very good simulation of real material behavior. Accurately captures strain rate sensitivity.

Limitations:

Does not model elastic recovery (spring back), and is therefore inappropriate for bending or other operations where spring back has a significant effect on the final part geometry. Does not model strains due to thermal expansion / contraction. Cannot capture residual stresses.

Elasto-plastic (Ela-Pla)

Elasto-plastic objects are treated as elastic objects until the yield point is reached. Then, any portions of the object that reach the yield point are treated as plastic, while the remainder of the object is treated as elastic. In the elasto-plastic deformation the total strain in the object is a combination of elastic strain and non-elastic strain. The non-elastic strain consists of plastic strain, creep strain, thermal strain and transformation strain depending on the characteristics of materials. Detail of different material models can be found later. In the case of brick elements, the elasto-plastic model is valid for all levels of strain.

Applications:

Provides a realistic simulation of elastic recovery (spring back), and strains due to the thermal expansion. Useful for problems such as bending where spring back has a significant effect on the final part geometry. Also useful for residual stress calculations. Object type must be elastic-plastic for creep calculations.

Limitations:

Does not model strain rate sensitivity, and as such is inappropriate for hot materials undergoing large deformations. Requires more solution time than rigid-plastic, and may have difficulties with convergence.

NOTE: If flow stress is defined for multiple strain rates, the flow stress of an elasto-plastic material is evaluated at the strain rate value specified in limiting strain rate under object->properties

Porous

Porous objects are treated the same as plastic objects (compressible rigid-viscoplastic materials) except that the material density is calculated and updated as part of the simulation. The material behavior is modeled similar to plastic objects but the model includes the compressibility of the material in the formulation. The limiting strain rate (LMTSTR) and the flow stress (FSTRES) must be specified at the fully dense state. The material density is specified at each element (DENSTY). Objects with changing material densities such as materials used in powder forming, should be modeled as Porous objects. The only iteration method currently available for the porous material is the direct solution method. This method does not have fast convergence capabilities; subsequently a porous simulation may take longer than a comparable plastic simulation.

Applications

Appropriate for compacted, sintered powders, beyond around 70 % accurately models consolidation and densification during forging.

Limitations

Cannot model loose powder or compaction processes.

2.4.5. Object geometry

In DEFORM the object geometry plays two roles. (See Figure 2.4.4, for Geometry data options)

- For deformable objects, the imported geometry is used to construct the finite element mesh. Since the object geometry changes, the original geometry is not stored.
- For rigid objects, the imported geometry defines the surface of the tool. If a mesh is generated for heat transfer, the original geometry definition is still used for the rigid surface definition. The original geometry can be

displayed in the *object/geometry* window.

- For deformable objects, the imported geometry is used to generate a mesh. Once a database is generated and the preprocessor is exited, the object geometry will be defined by the surface of the FEM mesh, and the original surface is no longer stored.

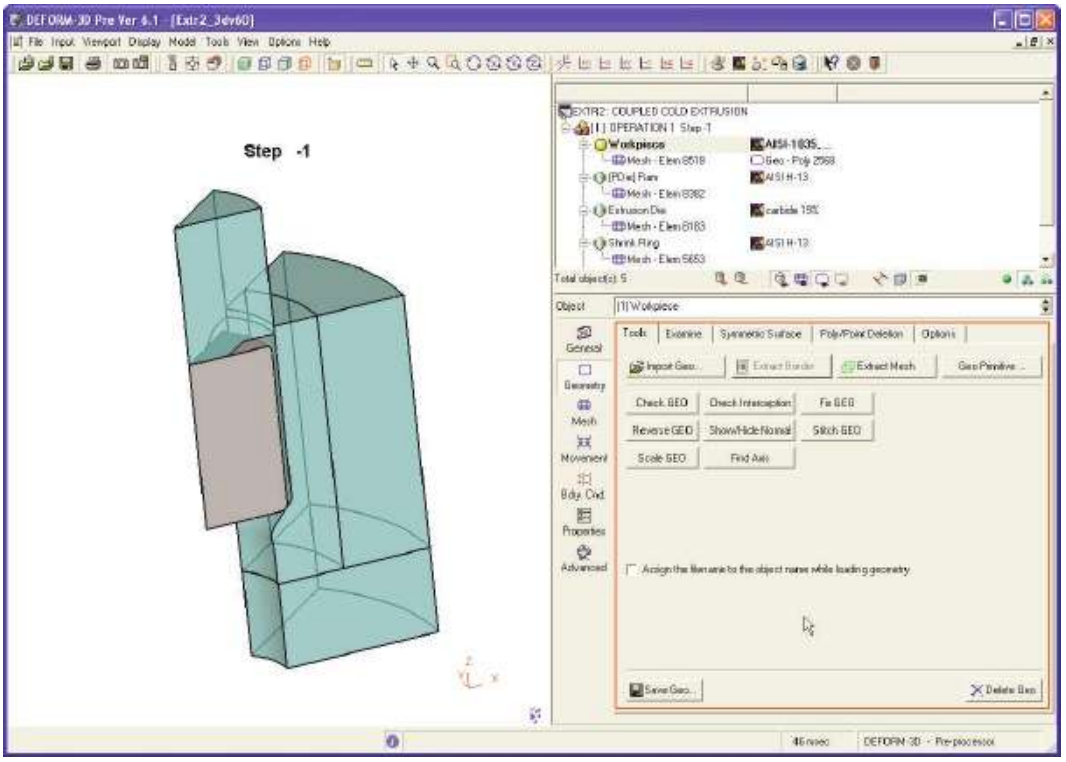


Figure 2.4.4: Geometry options window.

Geometry formats

STL format input (DIEGEO)

The STL format represents a surface by a series of three sided facets. This format may be created from almost all commercial solid modeling packages from a either a solid model or a surface model. For very simple shapes, such as a cube, very few facets may be used to provide an excellent representation of the shape. In the case of an extrusion die where the facets are used to model a curved surface, many facets may be required in order to give the object a smooth representation or to render small details in geometry. An economy of the number of facets used to represent geometry is recommended in order to minimize the size of the database file. As more facets are used, the size of each step in the database file will increase. The increase in the time for the contact calculations is negligible with the increase of the number of facets in the die geometries.
Upon inputting an STL file into the Pre-processor, the user is immediately prompted for a error tolerance value. This value is the snapping distance

between the points in the STL file. Since the facets are not dependent on each other, the points in which adjacent facets share may not be represented exactly the same way in an STL file. Since they were meant to be the same point, the Preprocessor assumes some error tolerance where the points are merged into representing the same point. The default value of 1e-005 is usually a good starting value. If there are small cracks in the die geometry, they may closed by increasing the error tolerance value and hoping that the cracks are snapped closed. This is not a very controlled manner in which to close any cracks and should be used with extreme caution. After using this method, the geometry should be carefully checked to ensure that no holes are introduced or important features are lost.

The file format for STL files may be either ASCII or binary format. DEFORM can both read and write ASCII and binary versions of the STL file. The facets are all defined independently of each other, so the danger of there being folds, holes, overlapping facets, or invalid facet orientations are possible. After reading an STL file, it is strongly recommended to check the geometry to make sure that there are no folds, holes or other problems. (see Figure 2.4.5) If there are geometry problems in a deforming body, problems may occur upon meshing the object. If there are geometry problems in a rigid die, problems may occur during the simulation where nodes get trapped and severely compromise the integrity of the deforming body. This can be very problematic since problems in die geometries may not occur until well into a simulation. The manner in which to best determine if a die geometry is well defined or not is to try to apply a mesh to it. If a mesh can be generated on geometry, then it is a well defined geometry, however, if the meshing fails, then it is possible that there is a problem with the geometry definition.

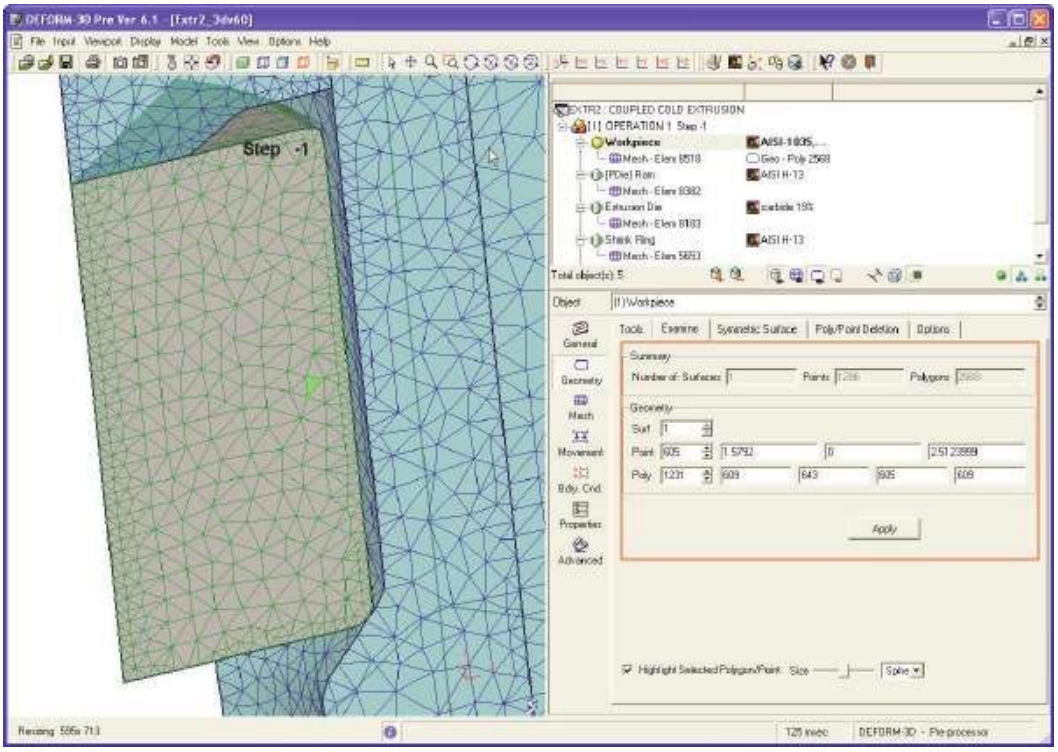


Figure 2.4.5: Examine information for object geometry.

AMGGEO format input

The AMGGEO format is a DEFORM internal format for handling geometries. This format can specify a surface as a set of connecting triangles or quadrilaterals. If quadrilaterals are used, degenerate elements (ie triangles) are not permitted. The patch normals must be out of the element. That is, the points should be numbered counterclockwise when viewed from the outside of the object. The file can be created and edited using a text editor such as vi, emacs, or Notepad.
The file format is

```
NUMBER OF VERTEX POINTS
1   X1 Y1 Z1
2   X2 Y2 Z2
 ......
N   XN YN ZN
NUMBER OF SURFACE PATCHES
1   first patch connectivity  1 2 3 (1/4)
2   second patch connectivity  1 2 3 (1/4)
 ......
N   Nth patch connectivity   1 2 3 (1/4)
```

where (1/4) in the connectivity indicates that point 1 is repeated in the 4th position in a triangular patch. All 4 points are used for a quadrilateral patch.
A 1" by 1" square patch in the xy plane with normal pointing along the z axis would be defined as follows:

```
4
1 0. 0. 0.
2 1. 0. 0.
3 1. 1. 0.
4 0. 1. 0.
```

```
1
1 1 2 3 4
```
The square would be defined using two triangles as follows:
```
4
1 0. 0. 0.
2 1. 0. 0.
3 1. 1. 0.
4 0. 1. 0.
2
1 1 2 3 1
2 1 3 4 1
```

PATRAN format input
The PATRAN neutral file format is an output format from PATRAN. This format specifies a either a surface mesh or a solid mesh which can be used to either represent a geometry. Upon loading a PATRAN neutral file, the user is first prompted whether the neutral file is either a surface mesh or a solid mesh. After the user provides the information on whether the file is a surface or a solid mesh, the user is prompted to provide a conversion factor. This is merely a scaling variable and the user is recommended to just use the default value of 1.

IDEAS format input
The IDEAS neutral file format is an output format from IDEAS. This format specifies a surface mesh which can be used to either represent a geometry in DEFORM or as a basis for a solid mesh. Upon loading an IDEAS universal file, the user is first prompted to provide a conversion factor. This is merely a scaling variable and the user is recommended to just use the default value of 1.

Geometry rules
There are several conventions that must be followed when defining object surfaces in DEFORM.

Orientation of Surface Normals
In DEFORM the surface normals of the closed geometries should point outwards from the geometry. This is how DEFORM defines the exterior of an object. In the case of a surface that isn't closed, the surface normals should point toward the deformable objects and great care should be taken that no nodes see the back of the object. In the case where a rigid plane is used to constrain a work piece, it is recommended to make the plane sufficiently large such that the nodes cannot see around the plane. In the Geometry window, the direction of the surface normals can be viewed by clicking on the surface normal button in the lower left part of the screen. Failure to follow this convention may cause any of the following problems:

- object won't mesh
- mesh distorts when boundary conditions are applied
- object positioning error using interference positioning

Surface Patches
In DEFORM a surface patch is defined by a section of a surface that is separated

from other portions of the same surface by a 30 degree or greater bend in the surface. For example, a cube would have six surface patches due to the edges between each side having a 90 degree bend in the surface. In order to view the surface patches in DEFORM the user may click on the surface patches button in the geometry window at the lower left section of the screen. Any bend in the surface greater than 30 degrees with appear as a thick red line. The benefit of this feature is that folds in the surface will appear as red slivers in the middle of the geometry. This provides a method for finding where folds may exist.

Border Extraction

Border extraction is the process of identifying the deformed part surface geometry from the surface of the finite element mesh. Geometric reasoning is used to identify critical features such as edges, corners, and symmetry planes which should be maintained during remeshing.
Border extraction can fail for the following reasons:

Folds or crossed elements

If a closed forming lap develops in the process, the surface geometry will be ill-defined. If an excessively large time step is used without polygon length sub stepping, element faces can become crossed, also causing an ill-defined surface. Both of these cases can frequently be identified using the *surface patches* display in the geometry window.
If a legitimate forming lap is developing, the process should be redesigned to eliminate the lap. If the lap is in a region where it is acceptable, it may be necessary to use a CAD system to edit the geometry, then remesh the part and interpolate data.
If element faces are crossed, it is generally necessary to revert to the last good step in the database. The situation can be avoided by using a smaller time step, using polygon length sub stepping, using smaller elements around tight corners, and forcing remeshings on a fixed step or stroke interval (under *remeshing criteria*)

Parallel symmetry planes

When using symmetry, the user should not specify parallel fixed velocity boundary conditions. (For a comprehensive discussion on symmetry planes, refer to the appendix section on the use of symmetry planes in 3D) In the case where two parallel symmetry planes are necessary, the user can specify one fixed velocity boundary condition and one rigid plane with no friction and a non-separable contact condition (To see how to implement this, please refer to the appendix section on the use of symmetry planes in 3D). If two fixed velocity boundary conditions are set parallel to one another, border extraction will surely fail, causing any remeshing to fail.

If the symmetry plane is not sufficiently large to cover the entire area where symmetry needs to be defined, it is possible that nodes may move around the plane of symmetry and this will also cause border extraction to fail. Since rigid planes, when used to define symmetry planes, need not have relations

to any objects other than the work piece, they may be arbitrarily large. For display reasons, the user is not recommended to make the rigid planes unreasonably large.

Geometry checking
Always check geometry. DEFORM has a checking algorithm that checks for number of invalid edges, invalid orientation, polygons with small area and number of surfaces. Every type of error can not be detected. Below are a few common geometric errors and how they are corrected by DEFORM.

Note: Correct orientation of the surface normals is NOT checked in geometry checking if all the normals are pointed in a consistent direction .

GEOMETRY ERROR	CORRECTION ADVICE
Poly with invalid orientation	Either fix the STL file in the solid modeling package or
	find the problematic poly in the Preprocessor and fix the
	STL file manually.
Poly with small area	Increase the error tolerance slightly
Poly with invalid edge	Fix the geometry in the solid modeling package

Fix geometry
This feature will handle geometric problems where there are either multiple surfaces or open (holes) regions by deleting any extra surfaces and filling holes. Other problems, like unshared edges and

Symmetric Surfaces

DEFORM-3D now supports the ability for the rigid tools to be of the same size as the work piece at symmetric surfaces. This is determined automatically during the simulation but can also be defined manually in the object geometry/symmetric surfaces window (See Figure 2.4.6). Both planar symmetry and rotational symmetry can be defined. In the case of planar symmetry, the simulation will have extra information that allows it to prevent material from flashing around it. In the case of rotational symmetry, meshing will automatically place the proper boundary conditions on the faces. This is meant as a uniform place to apply symmetry boundary conditions for all objects.

Specifying Planar Symmetry

To specify planar symmetry, click on the plane to maintain symmetry and then click the apply button. The planar symmetry condition will be added to the list of currently specified symmetry.

Specifying Rotational Symmetry

To specify rotational symmetry, specify the point and vector of the rotational axis as well as the degree of symmetry available. After this, click on the plane to be applied rotational symmetry. The symmetry condition will be added to the list of currently specified symmetry.

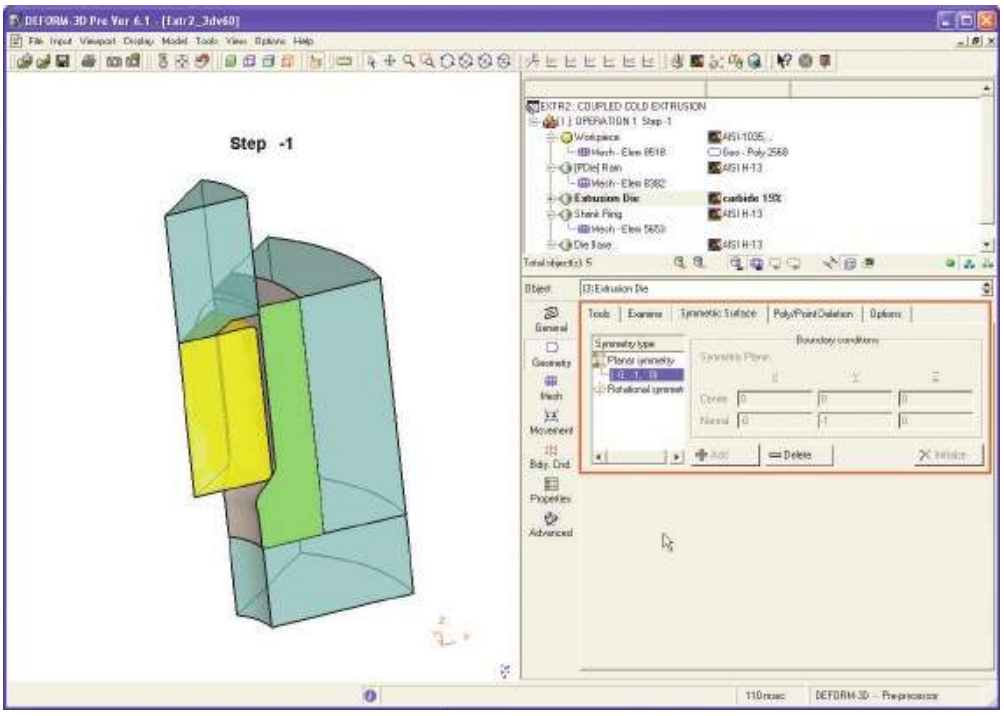


Figure 2.4.6: Symmetry surfaces can be marked in object geometry allowing the user to make a tool the same size as the work piece. This feature is activated automatically during the simulation. Note the symmetry surfaces are marked in bright green.

Geometric polygon deletion

Polygons in object geometry can be manually deleted within the DEFORM Preprocessor. This is not recommended as a manner of performing major changes to geometry rather it is recommended to correct slight mistakes within geometry or to make minor trimming operations. (See Figure 2.4.7).

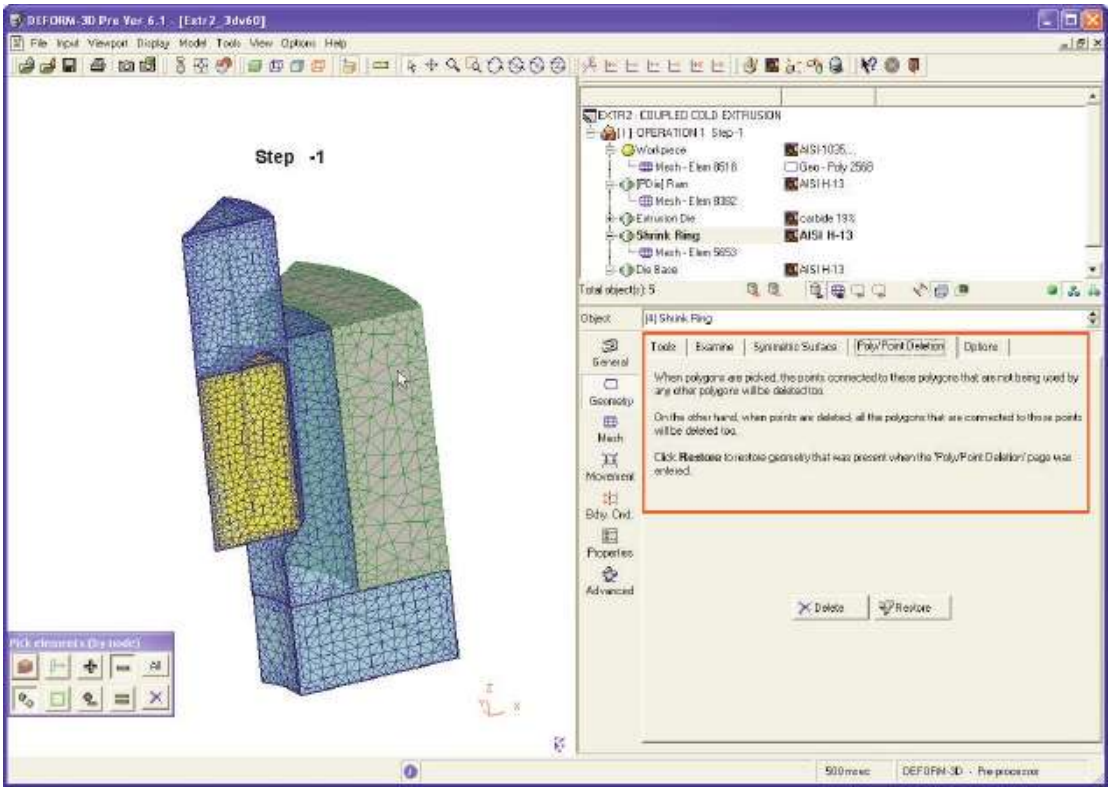


Figure 2.4.7: The geometry polygon deletion window.

2.4.6. Object meshing

The object meshing window can generate a mesh very easily with the first meshing window (See Figure 2.4.8). A mesh can be imported into the preprocessor from a keyword file, database file or any supported mesh formats (PDA or universal file). The number of elements to be generated in an object can be specified merely by adjusting the slider bar and selecting an appropriate value for the current simulation. After this, the surface mesh can be viewed by clicking Preview or the solid mesh can be generated by clicking Generate Mesh. The mesh can be examined for problems using the Check Mesh function. If a simulation is continued from a previously running simulation, the recommended button to click is Manual Remesh. This automatically performs border extraction and will prompt the user for boundary condition interpolation.

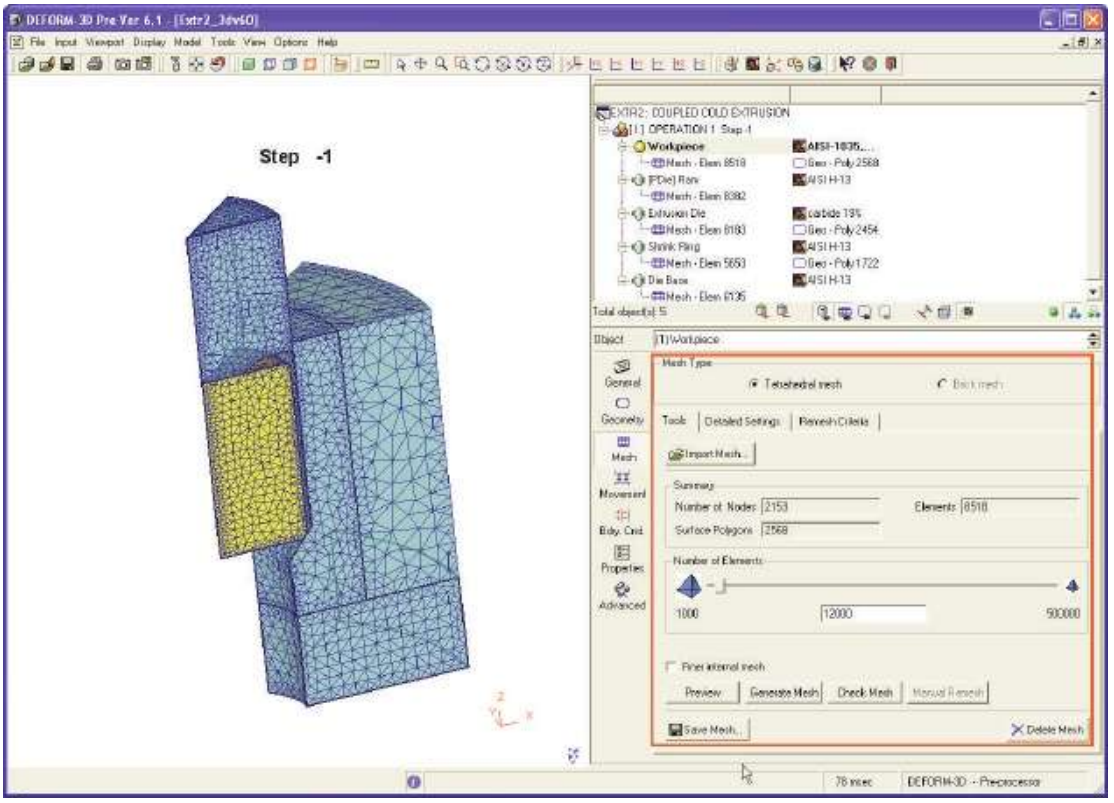


Figure 2.4.8: Meshing window.

The *Mesh Generation* window allows the user to generate a both a surface mesh and a solid mesh for the current object. The user has three methods of controlling surface mesh density:

- **System Defined** method uses a system of weights and assigned windows to control the size of elements during the initial surface mesh generation and subsequent automatic remeshing. This method includes user controlled mesh density windows.

- **User-Defined** allows the user to specify certain areas on the object to have higher element densities relative to other areas of the object during the initial surface mesh generation only (this density specification is not referenced during automatic remeshing).

Mesh density refers to the size of elements that will be generated on a object surface. The mesh density is normally based on the specified total number of surface elements and "Parameter" mesh density controls. The sample grid resolution and the critical point tolerances also affect the mesh density, but to a lesser extent than the other parameters. If mesh density windows are used with absolute mesh density, the number of surface elements will be completely determined by the specified mesh density.
A higher mesh density (more elements per unit area/volume) offers increased accuracy and resolution of geometry and field variables such as strain,

temperature, and damage. However, in general, the time required for the computer to solve the problem increases as number of nodes increases. Thus, it is desirable to have a large number of small elements (high density) in regions where large gradients in strain, temperature, or damage values are present, or in regions with complex geometry. Conversely, to conserve computation resources, it is desirable to have a small number of relatively large elements in regions where very little deformation occurs, or where the gradients are very small. Other issues related to mesh generation:

- Too coarse a mesh around corners may cause mesh degradation, excessive volume loss and remeshing problems
- Too coarse a mesh in regions with localized surface effects (i.e. high damage along a surface) may cause peak values to decline due to interpolation errors during remeshing.

From 3D Version 5.1, a checkbox has been added to allow the user to create a finer internal mesh. This feature can be used when the user would like to see more solid elements throughout the thickness of the part. Note that the total number of elements may be significantly increased when this option is used. A comparison of with and without fine internal mesh is seen in the figure below. (See Figure 2.4.9, and controls as indicated in Figure 2.4.8)

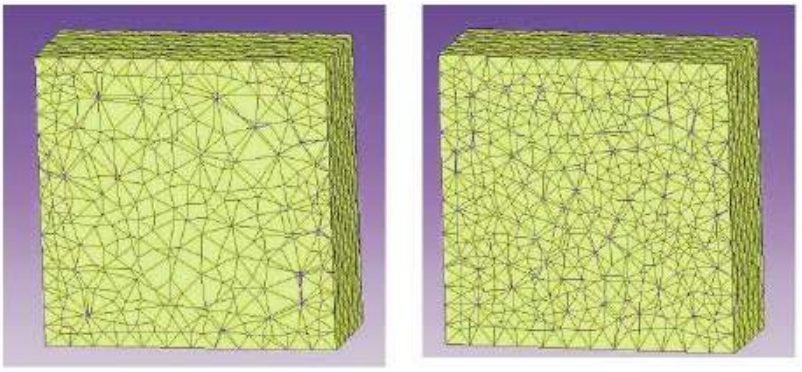


Figure 2.4.9: Comparison of work piece from UPSET.KEY with 10000 elements specified at generation. Left image shows the internal mesh without fine internal with and right side is with fine internal mesh.

Basic mesh controls

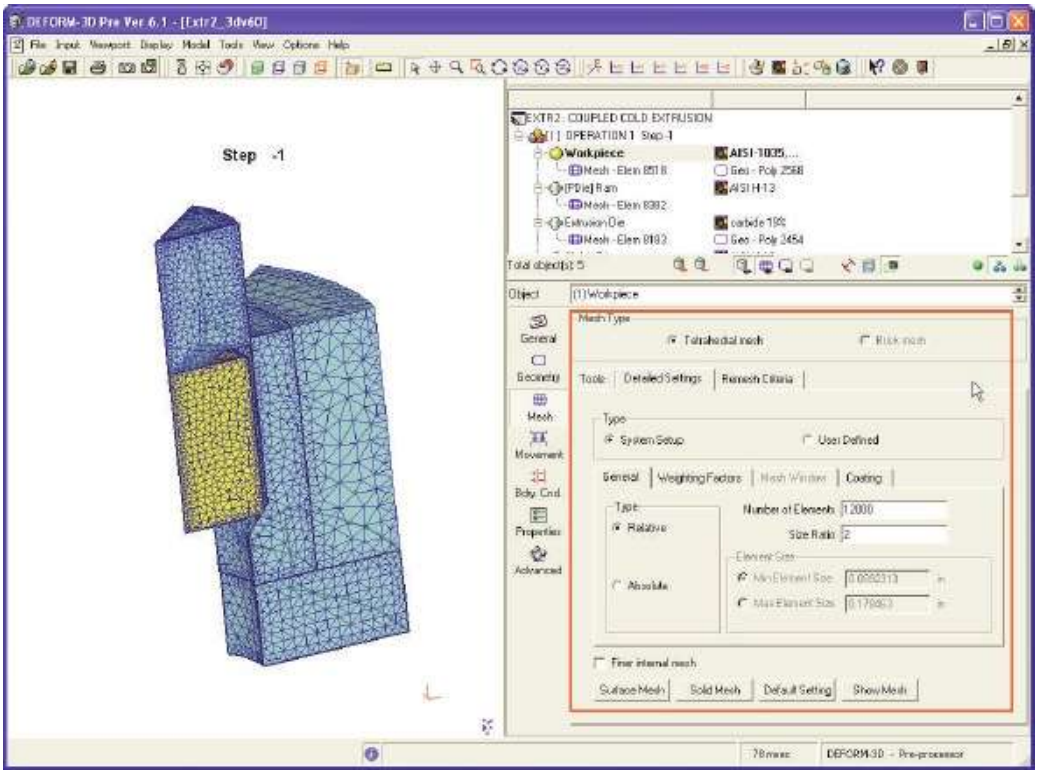


Figure 2.4.10: Detailed settings meshing parameters window.

Surface elements (MGNELS)

The number of surface elements represents the approximate number of surface elements that will be generated by the mesh generator. The Automatic Mesh Generator (AMG) takes the value for MGNELS and generates a mesh that will contain approximately the same number of elements.(see Figure 2.4.10)
This value is ignored if mesh windows are used with absolute mesh density definition.

Element size ratio (MGSIZR)

The maximum size ratio between elements is one of several ways to control the mesh density during automatic mesh generation (AMG) by specifying the ratio of surface element edge lengths. For a value of 3 for MGSIZR, the largest element edge on an object will be roughly 3 times the size of the smallest element edge on the same object. If equal sized elements are desired, then Size Ratio = 1.

Mesh weighting factors

The weighting factors or parameters (system defined mesh density) for boundary curvature, temperature, strain and strain rate specify relative mesh density weights to be assigned to the associated parameter.
Temperature, strain, and strain rate densities are assigned based on gradients in

these parameters, not absolute parameter values. That is, a region with a rapid temperature change in a particular direction will receive more elements that a region with a uniform high temperature.
The values from all the mesh density keywords are combined during the mesh generation process to create a mesh density distribution within the geometric boundary.

Polygon edge length based weighting factor (MGWCUV)
The Polygon edge length weighting will apply a higher mesh density to regions where the geometry edge lengths are smaller. The purpose of this is to apply a finer mesh in regions where smaller facets are used to give a finer mesh in the region of small features. If MGWCUV is greater than 0, the boundary area with curves will receive a higher mesh density in that area. If MGWCUV is set to 0, this weighting criterion is ignored.

Strain base weighting factor (MGWSTN)
This parameter will maintain a high mesh density in regions of high strain gradient.

Strain rate based weighting factor (MGWSTR)
This parameter will maintain a high mesh density in regions of high strain rate gradient

Temperature based weighting factor (MGWTMP)
This parameter will maintain a high mesh density in regions of high temperature gradient.

User defined mesh density
Refer also to Mesh density windows
When a user defined mesh density is selected, the density values are set through the *Display Window*. By selecting a user defined mesh density, the relative mesh density definition screen becomes active. After clicking this button the user will see the Display Window through which the mesh densities can be set. The density value is used to specify a weight to be given to an area. Note that the actual number only has meaning in relation to another density point. For example, a patch with a weighting of 4 will have a mesh that is two times as dense as a patch with a weighting of 2. One can also choose between setting the density values as boundary or interior points. If a mistake is made, Delete will delete the current point (shown in red) and Delete All will delete all density values set.
Note: For more detailed instructions, see DEFORM tutorial Lab 6, Mesh Generation for Dies in SPIKE.

Mesh density windows

The Mesh density window concept is similar to a user defined mesh density. The mesh density specified for a given window is applied to any geometry point (node or STL vertex) inside the window. However, the mesh density window is used during remeshing as well as initial mesh generation, whereas user defined mesh densities are used only during initial generation. It can also be assigned a velocity and it can be defined in an area such as a flash gutter which the work piece has not yet reached.

Mesh density is defined by the number of surface nodes per unit length. The mesh density values may specify a mesh density ratio between two regions in the object or an absolute element edge length in a region on a surface. In the case of a relative mesh density between one or more regions, the unspecified areas will be given the global value. In the case of absolute mesh density, the unspecified regions will also take on the global value. In the case of absolute mesh density specification, the user should be careful what value of density is used.

Important considerations in mesh density window definition:

1. There needs to be only one mesh density window. The global density values will used for any region that is not inside a window. *This differs from 2D*
2. Be aware of large mesh density ratios. A 3:1 density ratio is large, and a 5:1 ratio is extreme. Extremely large density ratios may lead to problems in mesh generation.
3. Absolute mesh density is a powerful tool, in that it allows specific resolution to be defined in various areas of a part. Consider a feature .5" thick. To maintain 3 elements across the thickness of the feature, specify an absolute mesh density of 6 elements/inch in that region. As more material enters this region, the total number of elements will be increased as necessary to maintain the desired resolution.
4. If absolute density is used, the user must be careful of the values that are specified since there is no specified upper bound on the number of elements.

Defining mesh density windows

1. Select a window number under the mesh density window header. Up to 20 different windows can be defined.
2. Click the add bounding point button and click on the part to create a mesh density window.
3. Click on the Drag window button to adjust the size and location of the mesh density window.
4. Once a window is defined, its density is entered in the box under the Parameters header.
5. A velocity for the mesh window can also be defined in the x and/or y direction. These values are also found under the Parameters header.
6. If the window needs to be rotated, the user can use rotation buttons in the display box.

7. After the window is applied in the position that the user desires, pressing the Show button in the display box will highlight the surface nodes that are encapsulated by the mesh window. The Hide button will de-highlight the nodes contained by the mesh density window.

Mesh density windows have the following data associated with them:

Absolute mesh density

Absolute mesh density defines the number of elements per unit length on the surface of the part. For example, an absolute mesh density of 8 will give 8 elements per inch (millimeter in SI units), or each element edge length will be approximately 0.125 long. With absolute mesh density, the total number of elements specified on the Meshing/Remeshing window is disregarded. The number of elements will be adjusted throughout the simulation, maintaining an optimum problem resolution. This is the recommended mesh definition method.

Relative mesh density

Relative mesh density defines the ratio of element edge lengths. The total number of surface elements is determined by the number of elements block on the Meshing/Remeshing window.

Points

The Points represents the total number of points that make up the mesh density window.

Density

The Density is the desired density value for a given window.

Velocity

The Velocity is the velocity of the window. This allows the window to move with the dies. In cases where punch velocity is not known, such as a hammer forging, or a load controlled press, the best estimate of a constant velocity should be made.

Note: If a velocity is assigned to a window, it should be repositioned as necessary before a second or third operation is performed.

Weighting factor defined edge length

From DEFORM-3D Version 3.2, a new automatic mesh density determination feature has been added. This feature is intended to reduce the reliance on mesh density windows to get an optimized mesh.
The mesh weighting is determined by the slider bars on the Mesh window. Polygon Edge Length will put more elements in areas of greater curvature. The other slider bars will weight based on Strain, Strain Rate and Temperature Gradient (not absolute values).
To activate automatic density determination, Select "Absolute Density" on the Mesh Density Windows dialog, but do not define any windows. Enter a global density. This will be the coarsest mesh anywhere in the part.

The smallest element size will be determined by the Maximum Size Ratio block on the main Meshing/Remeshing window. A maximum size ratio of 3 or 4 is probably a good place to start. As always with absolute mesh density, the total number of elements is ignored.

How to select mesh density
Determine the finest mesh density required based on anticipated curvature, die corners, defect size, etc. For example, for a .125 radius, we would like to have elements slightly smaller than this (maybe .100"). This would correspond to an absolute mesh density of 10 (1 / 0.100). Now determine the global density by taking 1/3 of this value, so enter 3 in the global density field in the Mesh Density Windows window.

Surface mesh generation
When all of the Mesh parameters have been set, a surface mesh can be generated by clicking on the Generate Surface Mesh button. When a new mesh is generated for an object that currently has a mesh, the old mesh will be deleted and replaced with the new mesh. If there is a failure in the generation of the surface mesh, please refer to the *Troubleshooting* section.

Solid mesh generation
After the surface mesh is generated, the user should inspect the mesh before generating the solid mesh. Pay particular attention to adequate mesh density in regions with complex geometry. After an acceptable surface mesh has been generated the solid mesh may be generated by clicking the Generate Solid Mesh button. If a surface mesh is imported as the geometry, the user may forgo the surface mesh generation and directly place a solid mesh on the surface mesh. If the solid mesh generation fails, please refer to the *Troubleshooting* section.

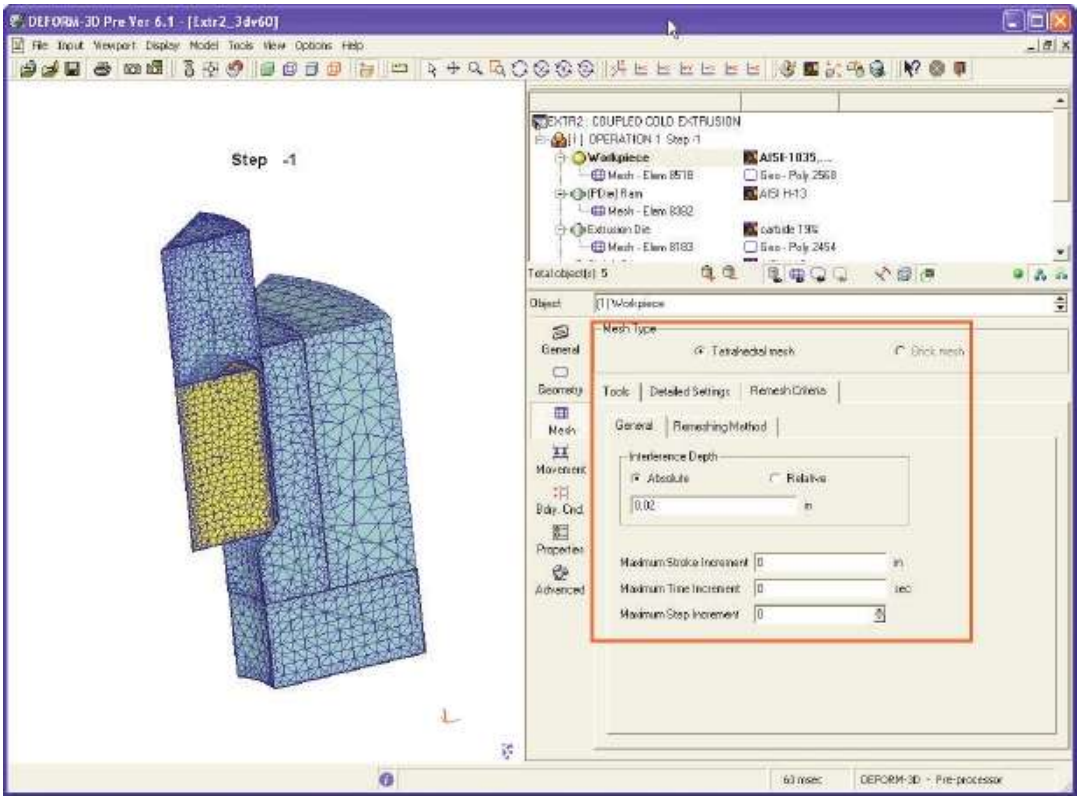


Figure 2.4.11: Remeshing criteria window.

Automatic remeshing criteria

Automatic remeshing (Autoremesh) is the most convenient way to handle the remeshing of objects undergoing large plastic deformation. The Remeshing Criteria Window contains a group of parameters that control when and how often the mesh will be regenerated on a meshed object based on assignment of certain triggers (See Figure 2.4.11). There are four keywords that control the initiation of a remeshing procedure (RMDPTH, RMTIME, RMSTEP, RMSTRK) for an object. When the remeshing criteria of any of these keywords has been fulfilled or the mesh becomes unusable (negative Jacobian), the object will be remeshed. During the simulation, if an object satisfies any of its remeshing criteria, a new mesh is generated, the solution information from the old mesh is interpolated onto the new mesh and the simulation continues. Due to the nature of 3D meshes, a mesh may degrade beyond the point where it is usable if no remeshing triggers are used. For typical meshes, remeshing every 10 to 30 time steps may be appropriate.

Interference Depth (RMDPTH)

Remeshing will be triggered when the an element edge of meshed body has been penetrated by the master object by a specified amount. The penetration distance is determined differently depending on whether the specified distance is positive or negative.

Penetration Distance (absolute):
If a positive number (in the unit of length) is entered, the program will conduct a check on each surface edge that has a contact node on each end. The distance from the middle of the edge to the die surface is calculated. If the maximum penetration depth exceeds the specified limit, remeshing will be triggered

Penetration Distance (relative):
If a negative number (a fraction) is entered, the program will conduct a check on each surface edge that has a contact node on each end. The distance from the middle of the edge to the die surface is calculated and divided by the original length of the edge. If the ratio exceeds the magnitude of the specified value, remeshing will be triggered.

Default Value:
The pre-processor now has a default value 0.8 with a relative setting.

Purpose of Criteria:
When a sharp edge on a tool or die impinges on the work piece, the sharp edge may deeply penetrate an element edge. If this depth is severe the elements may get stretched out and remeshing may become difficult. Before this depth is achieved, remeshing with place nodes about the edge and allow the simulation to continue unhampered.

Maximum stroke increment (RMSTRK)
Remeshing will be triggered when the stroke value is evenly divisible by the stroke remeshing increment.

Maximum time increment (RMTIME)
Remeshing will be triggered when the time value is evenly divisible by the remeshing increment. If remeshing is specified for every 10 seconds, remeshing will occur at 10, 20, 30, etc. If automatic remeshing is triggered by a negative Jacobian on a previous step, the remeshing will still occur.

Maximum step increment (RMSTEP)
Remeshing will be triggered at the end of a step whenever the current step number is evenly divisible by the step increment. If remeshing is specified every 15 steps, remeshing will occur at 15, 30, 45, etc.

Global and Local Remeshing Options
From 3Dv6.1, mesh generation has been enhanced with local meshing functionality. Default settings point to existing global remeshing procedures, where in every element of the old mesh gets replaced with new mesh element, followed by interpolation. New Local meshing functionality allows several options to control the element size and quality. Local remeshing also has options to keep the meshing truly local, to minimize the interpolation related errors (See Figure 2.4.12). In the current version, all the local meshing related settings are stored in

the local files, not in the database. This means when the user copies the database file from one folder to another these local remesh settings will not be carried over unless all the files are copied to the working folder.

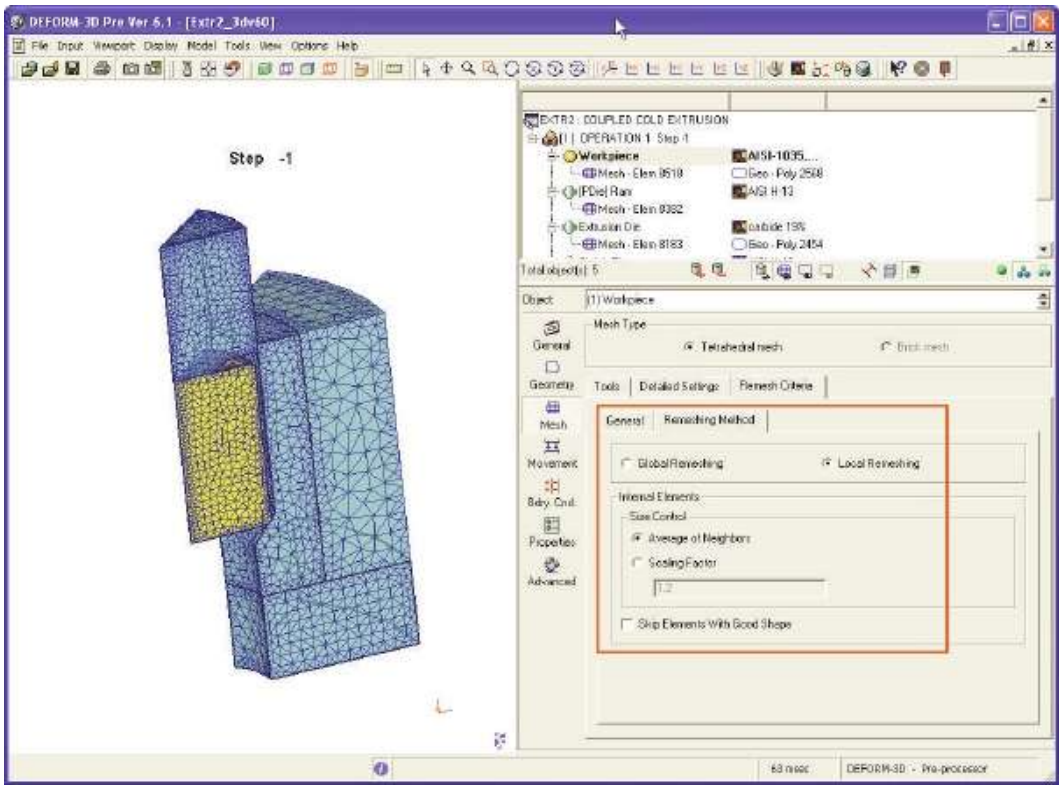


Figure 2.4.12: Local Remeshing criteria window.

Manual remeshing

During the course of a DEFORM simulation, extensive deformation of plastic or porous objects may cause elements in those object meshes to become so distorted that the mesh is no longer usable (negative Jacobian). If this condition occurs, the simulation will abort and an error message will be written to the ProblemID.MSG file. To continue a simulation after a mesh has become unusable, the object must be remeshed. Remeshing is the process of replacing a distorted mesh with a new undistorted mesh and interpolating the field variables (strain, velocity, damage, and temperature etc.) from the old mesh to the new mesh.

In the case of a hexahedral (brick) mesh, 3D cannot currently create a brick mesh so if a remesh is required for a elasto-plastic brick mesh, the user needs to remesh outside of DEFORM and interpolate the state variables and re-apply the boundary conditions to the new mesh.

In most cases, remeshing and interpolation occurs automatically without user intervention.

It is also possible to manually regenerate a mesh on an object and interpolate the

data from the old mesh. The procedure to perform a manual remeshing is as follows:

Procedure

1. Open the preprocessor.
2. Select the step from the database where remeshing is to be performed and load this in the pre-processor. If the object will not remesh at the last step, it may be necessary to remesh at an earlier step.
3. Select the object to be remeshed.
4. Select the Manual Remeshing option in the Objects window.
5. If the part geometry is to be modified (such as trimming flash or punching out a web, it may be done at this point using the geometry editor).
6. Adjust mesh windows or other mesh parameters as necessary.
7. Generate a new surface mesh.
8. Generate a new solid mesh.
9. Interpolate data from the old mesh to the new mesh by clicking on the OK button.
10. Interpolate boundary conditions from the old mesh to the new mesh unless:
    - Dies are being changed at the same time the part is being remeshed
    - The mesh visibly distorts on remeshing.
    - A negative Jacobian error occurs immediately when the problem is restarted
11. Generate a database and start simulation.

If the mesh visibly distorts after remeshing or if you are changing dies at the same time, regenerate the mesh and interpolate data but not boundary conditions. If boundary conditions are not interpolated, it is necessary to recreate all velocity, heat transfer, inter-object, or other boundary conditions. If there are no changes to the geometry (such as trimming the part) then the simplified manual remeshing icon can be used, this extracts the border and shows the mesh generation dialog. After meshing when exiting, interpolation of state variables and boundary conditions is carried out.

Brick Remeshing procedures provided with specific templates

The following are the special modules where in the system allows to create initial brick mesh using the template settings, and the procedures allow users to trigger brick remesh during the process, that require a remesh as a result of either bad element shape or user specified remesh conditions.

1. Cogging
2. Shape Rolling
3. Ring Rolling

2.4.7. Object material

Any object which has a mesh defined must also have a material assigned to it. The material data can be defined in the *Materials data* section of the pre-processor. Assignment is made through the general selection.
Either phase or mixture materials may be assigned to each object. In general, phase materials which are not components of an alloy system will be assigned individually. An alloy system mixture will be assigned to the appropriate objects as a mixture, and relative volume fractions of the constituent phases should be assigned under element data.
For example:

- A tool is made of H-13. H-13 is defined as a phase. It should be assigned to appropriate tooling as a phase.
- A work piece is made of 1040 Steel. The simulation begins with the object composed of 100% volume fraction pearlite. 1040 is defined as a mixture of pearlite, banite, austenite, and martensite. The 1040 mixture properties are assigned to the object, and the volume fraction is set to 100% pearlite under element properties.

2.4.8. Object initial conditions

Initial conditions can be specified for any object related state variable in DEFORM. The most common initial condition specification is object temperature which can be specified in the Objects->General window of DEFORM-3D (See Figure 2.4.13). For heat treatment problems with variable carbon content in the work piece, dominant atom content may also be specified. For meshed objects, initial object temperature and initial dominant atom content are specified by assigning values to all the nodes.
When a mesh is generated, the nodes in that mesh will be assigned values from the *Meshing Defaults* fields under the *Defaults* tab in the Object window. Uniform object temperature can be specified using the *TEMP* button on the object window. Nodal values may also be specified using the *Nodes Data* menu. Values for an entire object can be set using the initialize (i) icon next to the appropriate data field.
For non-meshed rigid tools, a constant object temperature may be set using the reference temperature (REFTMP) under the *Objects, Properties* menu.

Note: *Using this approximation will tend to over-estimate temperature loss as the die surface will not heat up during the simulation. This effect can be compensated for by reducing the inter-object heat transfer coefficient (IHTCOF).*

For any object defined as a mixture, the initial volume fraction (VOLFC) and maximum volume fraction transformed (VOLFS) must be assigned for all volume fractions. In general VOLFC, and VOLFS should be initialized to the same value. The volume fraction initialization is under the *Object Data, Elements* dialog under the *Transformation* tab.

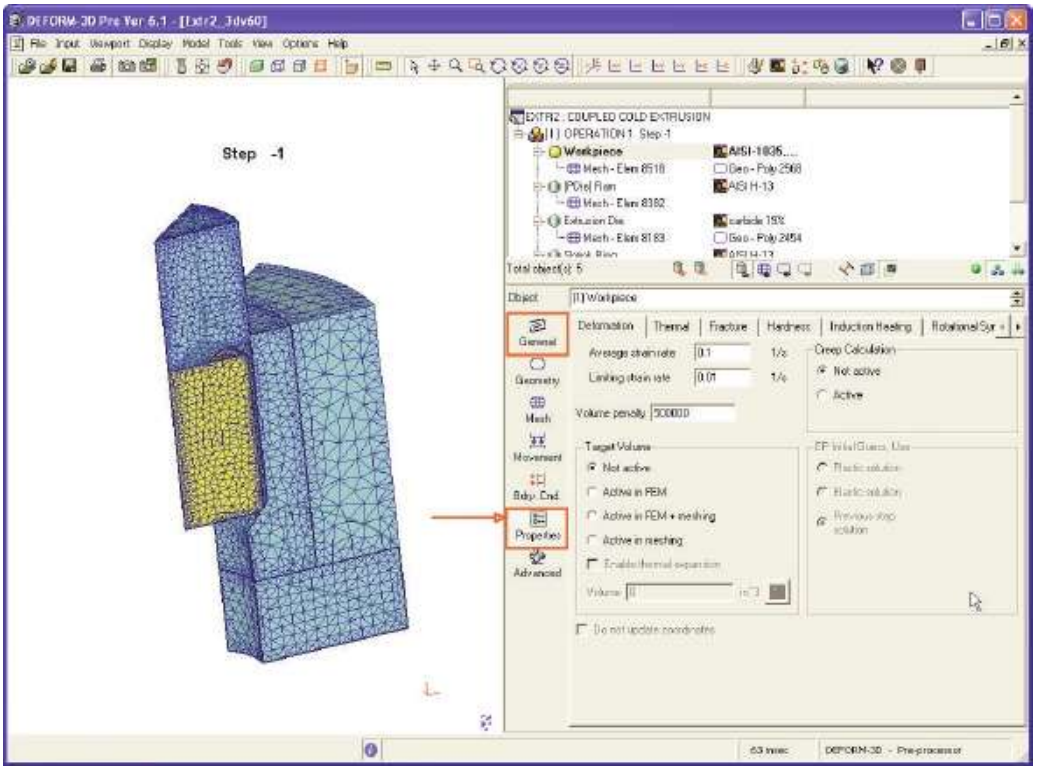


Figure 2.4.13: Object deformation properties.

2.4.9. Object properties

Miscellaneous object parameters which affect either thermo-mechanical behavior of the object, or numerical solution behavior, are specified in the *Object-Properties* window (See Figure 2.4.13).

Deformation properties

Average strain rate (AVGSTR)

The average strain rate is a characteristic average value of the effective strain rate. An approximation of this value should be given at the start of the simulation. A reasonable approximation can be obtained from:

$$\dot{\epsilon}_{avg} = \frac{V}{h}$$

where V is the initial velocity of the primary die, and h is the maximum height of the work piece.

Limiting strain rate (LMTSTR)

The limiting strain rate defines a limiting value of effective strain rate below which a plastic or porous material is considered rigid. The stress-strain-rate relationship in the rigid region is approximated by

$$\dot{\epsilon}_{ij} = \frac{3}{2}\frac{\dot{\bar{\epsilon}}_0}{\bar{\sigma}_0}\sigma'_{ij}$$

DEFORM automatically maintains the ratio between average strain rate and limiting strain rate. Generally, the value of limiting strain rate should be 0.1% to 1.0% of the average strain rate.
If the limiting strain rate is too small, the solution may have difficulty converging. If it is too large, the accuracy of the solution will be degraded. If the problem is not converging, the limiting strain rate can be increased for 2 or 3 steps, then returned to the original value.

Volume penalty constant (PENVOL)
The volume penalty constant specifies a large positive value that is used to enforce volume constancy of plastic objects. The default value of 10^6 is adequate for most simulations. If the value is too small, unacceptably large volume losses may occur. If the value is too large, the solution may have difficulty converging.

Target Volume (TRGVOL)
There are several causes of volume loss in finite element analysis.

- The penalty formulation used by DEFORM will naturally loose a fractional percentage of volume at each step. This is normal and generally not a significant cause of concern.
- If a large time step is used and sub stepping is disabled, when contact occurs nodes will penetrate slave surfaces, then be repositioned at the end of the step. This repositioning can cause slight volume loss. Over the course of a simulation, this can become significant.
- As elements of slave objects stretch around corners of master objects, the elements will cut the corner of the object. The volume which crosses the corner will be lost on remeshing. This phenomenon can be limited by the use of small elements around corners.

The system provides several controls to minimize this volume loss during the simulation. (see Figure 2.4.14) Typical volume compensation can be activated to maintain or restore part volume during remeshing. The target volume should be set to the initial part volume. This value can be obtained from the *volume* icon on the *Meshing/Remeshing* window.
For porous materials, the volume is expected to change throughout the simulation. If volume compensation is activated, the current part volume will be maintained during remeshing.
For certain geometries with large free surfaces, volume compensation can cause distortion. If this distortion is unacceptable, the best alternative is to use a fine mesh, and set polygon length sub stepping to a small value. Frequent forced remeshings may be useful if element stretching around corners is a problem.

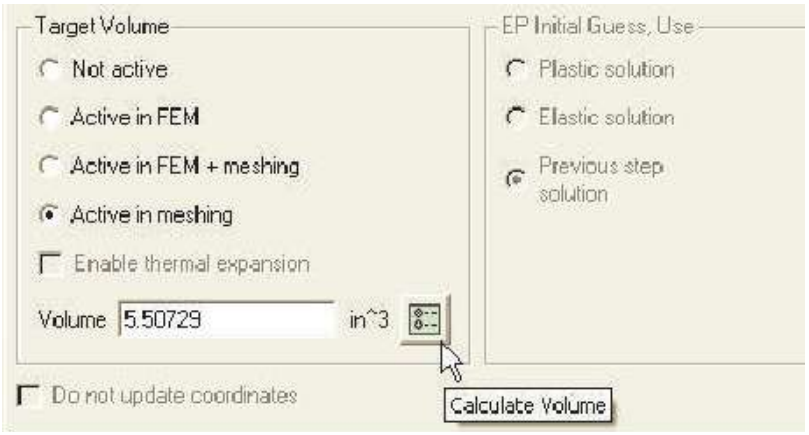


Figure 2.4.14: Object deformation properties.

Elasto-plastic initial guess (ELPSOL)
The convergence of an elasto-plastic solution is dependent on the initial guess of the stress-strain state. Three initial guess solutions are available:

- Plastic solution: Uses the purely plastic deformation data to generate the initial guess.
- Elastic solution: Uses the purely elastic deformation data to generate the initial guess.
- Previous step solution: Uses the elasto-plastic solution from the previous step to generate the initial guess.

The previous step solution seems to give the best convergence in most cases. If convergence is poor for a particular problem, the elastic or plastic solution can be used.

Creep solutions (CREEP) [MIC]
Activates creep calculations for a particular object. For more information on creep calculations, refer to chapter 6.

Reference point (REFPOS)
Point on object used in distance calculation for the Stopping Distance stopping control. Refer to Stopping Distance in the Simulation Controls-Stopping / Step-Stopping Controls subsection.

Thermal properties

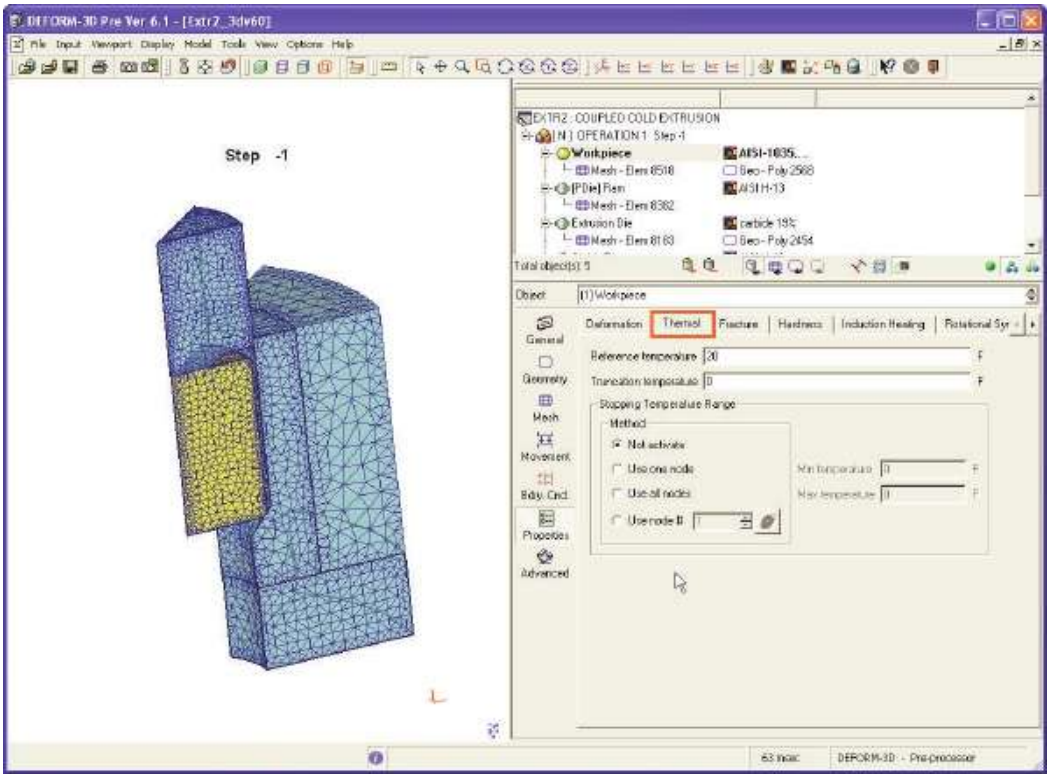


Figure 2.4.15: Object thermal properties window.

Reference temperature (REFTMP)
For elastic objects, the reference temperature is the temperature on which thermal expansion calculations are based. That is, thermal strains are given by (see Figure 2.4.15)

$$\epsilon^{th} = \alpha(T - T_0)$$

Where α is the Coefficient of thermal expansion,
T_0 is the reference temperature and
T is the material temperature.
For elasto-plastic objects, instantaneous coefficient of thermal expansion is used. Coefficient of thermal expansion is set in the *Material Properties, Elastic* menu.

Truncation temperature (TMPLMT)
The Truncation Temperature is the maximum nodal temperature allowed at any point in the object. If the calculated temperature exceeds this value, it will be reduced to this value.

Stopping temperature (OTPRNG)
The stopping temperature sets an upper and lower temperature limit which, if exceeded, will stop the simulation. The user has the option of enforcing this limit if any single node exceeds the temperature or only if all nodes exceed the temperature or based on temperature at a specific node.

Fracture properties

This feature allows the user to activate the number of fracture elements which will determine whether a fracture simulation should perform element deletion or perform element softening. For an overview of fracture, please refer to Appendix F .This window is seen in Figure 2.4.16.



Figure 2.4.16: The fracture object properties window.

Hardness properties [MIC]
Material hardness predictions can be based on:(see Figure 2.4.17)

- Volume fraction of various phases
- Jominy curve data
- Cooling time

Hardness data for a material can be entered in the *Material Properties* menu. A description of the hardness prediction method is given there.
Referenced Start temperature, referenced end temperature: Upper and lower temperature values for Jominy or cooling time hardness prediction curves.

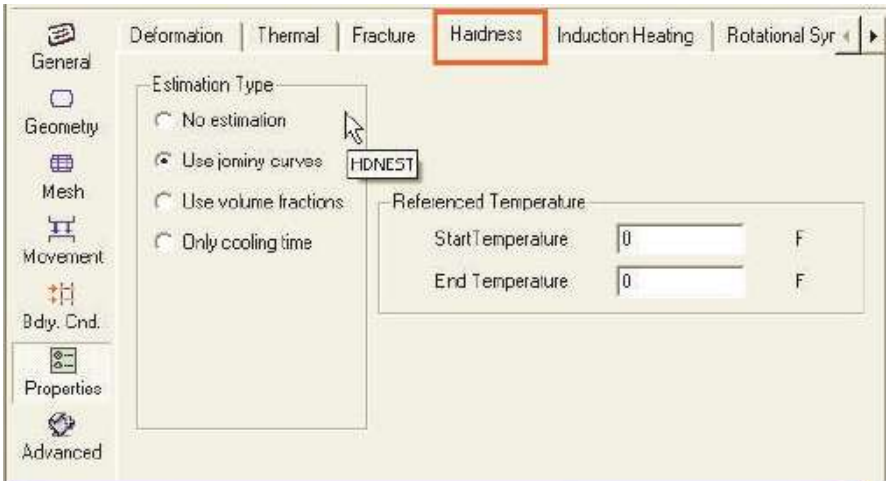


Figure 2.4.17: Object hardness properties

Induction Heating

When Induction heating computations are required, the same needs to be first turned on in the simulation controls (see Figure 2.4.18) before defining the associated object values like current frequency (see Figure 2.4.19). This simulation mode is supported starting from 3DV6.1

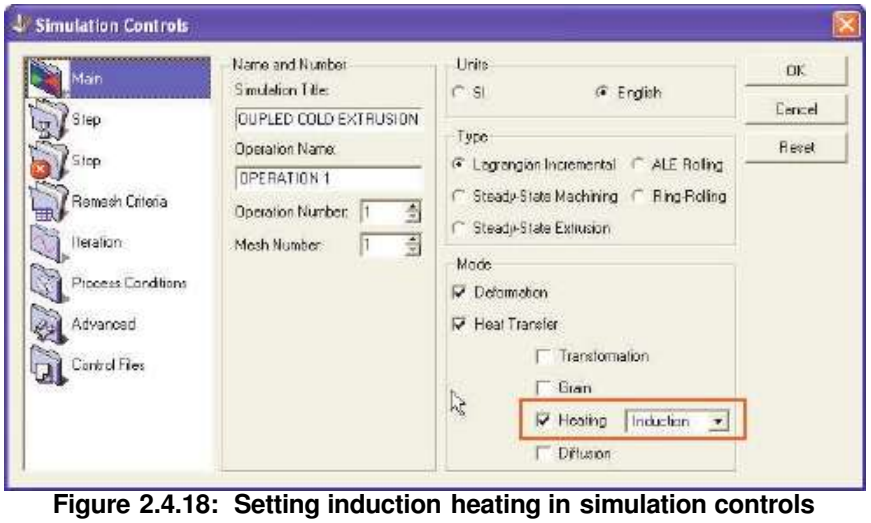


Figure 2.4.18: Setting induction heating in simulation controls

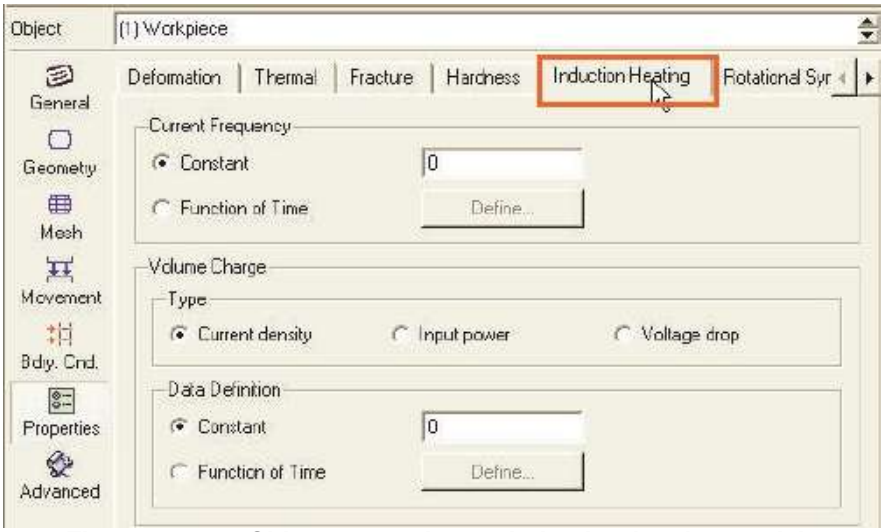


Figure 2.4.19: Object properties for induction heating models

Rotational Symmetry

This boundary condition allows the user to match the velocity of the nodes of any surface of a body to the nodes of any other surface on the same body. The purpose of this is to model the more general case where rotational motion occurs during the forming of a part such as in the case of the forging of a helical gear. The manner in which to setup a rotational symmetry problem is to go to the Object, Properties window for the object and select the Rot-Sym tab. Set the following values: (see Figure 2.4.20)

1. **Angle** The angle of the part simulated in units of degrees. For example, 180 means that only half the part is simulated and 90 means one quarter of the part is simulated.
2. **Center** A point on the centerline about which the deformation occurs. The format is in global (x,y,z) coordinates.
3. **Axis** The vector through which the centerline is parallel to. The format is in (x,y,z) coordinates. For example, if the user wants to specify a vector parallel to the z axis, the value of (0,0,1) should be typed in.

The second item the user needs to specify is the surfaces which have the rotational symmetry relationship to one another. The manner in which this is done is to place contact boundary conditions on face which obeys the right-hand rule. The boundary condition can be applied under the Objects, boundary conditions window using the advanced boundary conditions. The user needs to select the face which obeys the right-hand rule and apply self-contact conditions. This will allow the simulation engine to know which faces the rotational symmetry condition applies to.

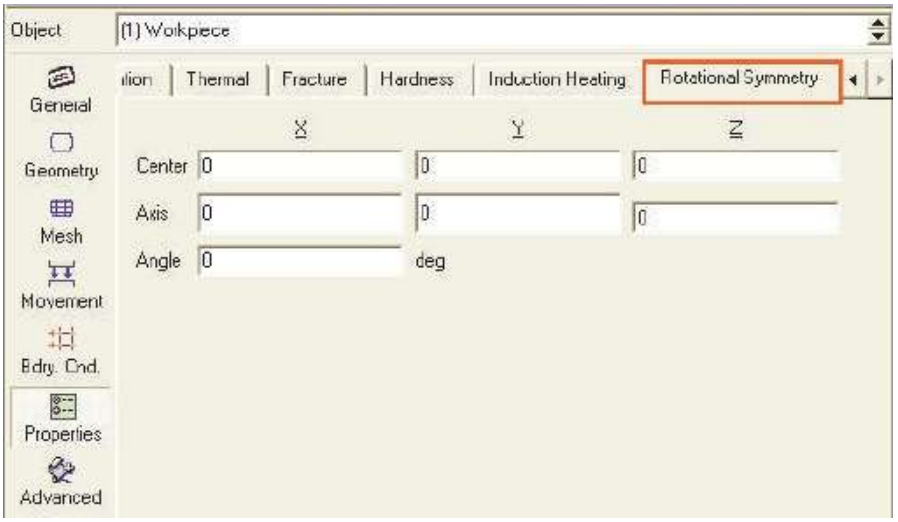


Figure 2.4.20: Object rotational symmetry properties.

2.4.10. Object boundary conditions

Boundary conditions specify how the boundary of an object interacts with other objects and with the environment. (see Figure 2.4.21) The most commonly used boundary conditions are heat exchange with the environment for simulations involving heat transfer, prescribed velocity for enforcing symmetry or prescribing movement in problems such as drawing where a part is pulled through a die, shrink fit for modeling shrink rings on tooling, prescribed force, for die stress analysis and contact between objects in the model.

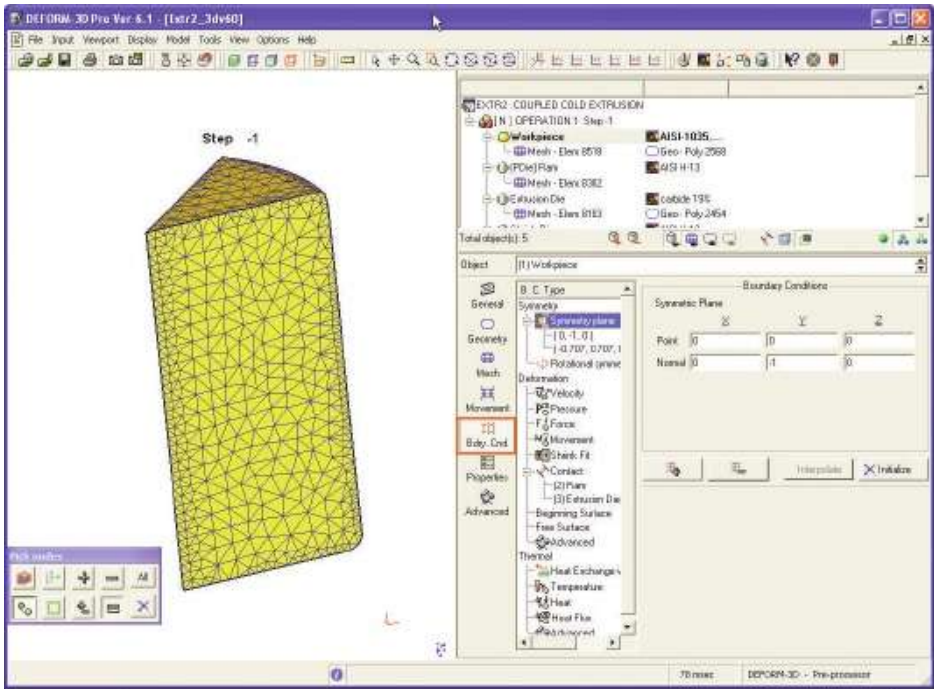


Figure 2.4.21: Object boundary condition window.

Defining object boundary conditions
Boundary conditions are specified and enforced at nodes in the finite element mesh. The basic procedure for setting any boundary condition except *Contact* is the same:

1. Select the appropriate condition type.
2. Select the direction (where applicable).
3. Select the nodes to which boundary conditions will be applied using one of the selection tools in the lower left button bar.
4. Apply the boundary conditions.

The selected nodes will be highlighted. To apply the boundary conditions click the Generate BCC's button. Colored markers will highlight the nodes to which boundary conditions have been applied. To delete specific boundary conditions, select the start and end nodes, and click the Delete BCC's button. To delete all boundary conditions of the specified type and direction, click the Initialize BCC's button.

Note: You can either select faces of the surface by using the surface patches feature or use the node button to select individual nodes .

Deformation boundary conditions

Velocity
Velocity of each node can be specified independently in the x and y directions (or x, y, and z directions in 3d). Velocity boundary conditions are normally set to zero for symmetry conditions (see section on symmetry in this manual), but may also be set to a specified non-zero value for processes such as drawing in which a work piece is pulled through a die.

Force
Force boundary conditions specify the force applied to the node by an external object. The force is specified in default units. For die stress analysis, the force that the die exerted on the work piece can be reversed and interpolated onto the dies by using the interpolation function. Refer to the tutorial labs on die stress analysis for a detailed procedure for using force interpolation to perform die stress analysis.

Pressure
The pressure boundary conditions specifies a uniform, or linearly varying, force per unit area on the element faces connecting the specified nodes.

Displacement and shrink fit
A specified displacement can be specified in any direction for each node. This is frequently used for specifying shrink fit conditions between a die insert and a shrink ring. More information on this is available in the section on die stress analysis in this manual.

Movement
The movement of nodes on an object can be specified. If the movement boundary condition is specified, object movement controls must also be specified.

Contact
The Contact boundary condition displays inter-object boundary contact conditions on a given object. The user should gain some experience with DEFORM before using this option. The contact conditions are stored in three components to represent the fact that there are three degrees of freedom for any given node.

Thermal

Heat exchange with the environment
This boundary condition specifies that heat exchange between element faces bounded by these nodes and their environment should occur. The contact boundary condition determines whether exchange will occur to the ambient atmosphere or to a contacting object.

Heat flux
Specifies an energy flux per unit area over the face of the element bounded by the nodes. Units are energy/time/area.

Nodal heat
Specifies a heat source at the given nodes. Units are energy/time.

Temperature
Specifies a fixed temperature at the given nodes.

Heat Exchange windows
This function allows the user to define heat exchange conditions for local areas on a body by use of three dimensional window. To use heat exchange windows, perform the following actions:

1. Go to the Boundary Conditions window.
2. Select the Thermal tab.
3. Select the Heat exchange windows button.
4. Note the tools in the lower left corner of the display window changes and the new heat exchange window that comes up.
5. At this point, heat exchange windows can be defined using the tools in the lower left corner of the display window. Each window has its own local environmental temperature, convection coefficient and emissivity. See Figure for an example heat exchange window.
6. You can define up to 20 independent windows by the method. If two regions share the same space, the lower number window wins.

Diffusion [DIF]

Diffusion with the environment
Specifies diffusion of the dominant atom through the boundary elements bordered by the indicated nodes. Environment dominant atom content and surface reaction rate are specified under the *Simulation Controls, Processing Conditions* menu. Environment content and reaction rate for various regions of the part may be modified by using diffusion windows.

Fixed atom content
Specifies fixed dominant atom content at the given nodes.

Atom flux
Specifies a fixed dominant atom flux rate over the elements bordered by the indicated nodes.

2.4.11. Contact boundary conditions

Contact boundary conditions are applied to nodes of a slave object, and specify contact between those nodes and the surface of a master object (see master-slave relationships under the Inter-object data section). If a node is specified to be in contact with a particular object, it will placed on the surface of that object. If this requires changing the position of that node, it will be changed as necessary. Contact boundary conditions are generated under the *Inter-object , Contact Boundary Conditions*.
It is for this reason that the user should be VERY careful with how contact is specified. If it is improperly used, the mesh may be damaged and very often remeshing cannot aid this situation since the AMG cannot interpret the users intentions.
Contact boundary conditions can be displayed for a given object using the *Objects, Boundary Conditions, Advanced Deformation BCC's* icon.

2.4.12. Object movement controls

Movement controls can be applied to: (see Figure 2.4.22)

1. Rigid objects;
2. It can also be defined boundary nodes of meshed objects with a movement boundary condition. The surface defined by these nodes can be thought of as a "rigid surface". *Note that these nodes should not be or become a contact surface.*

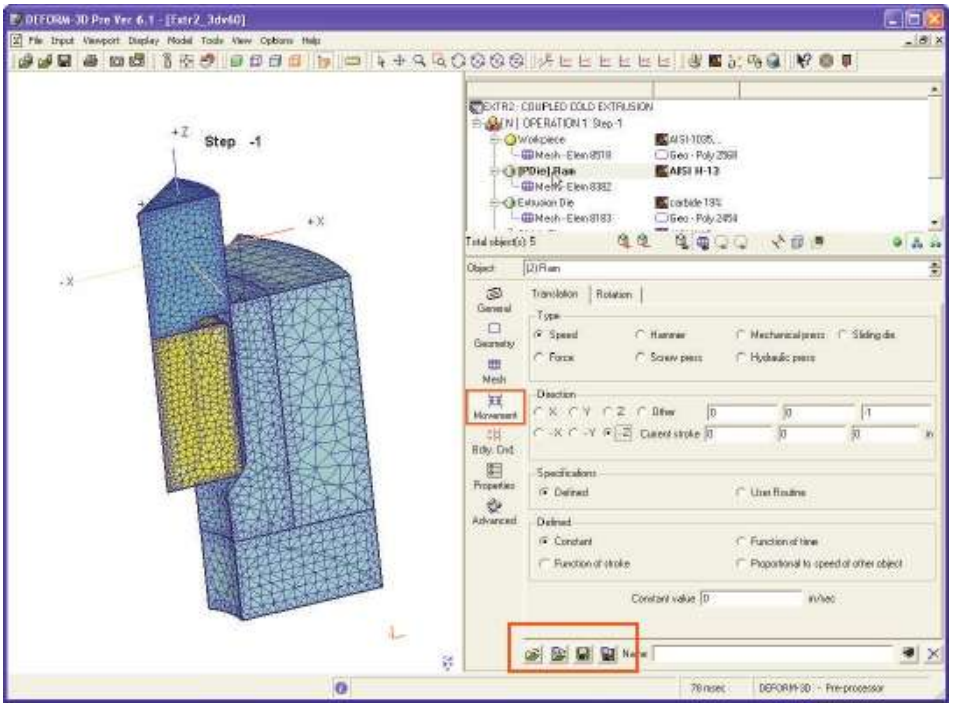


Figure 2.4.22: The import movement controls window.

The lower portion of the opening window of the movement controls window (See Figure 2.4.22) allows the user to import or save movement specifications from other keyword or database files. The user can also preview the motion of the die using the preview button.

Movement Preview

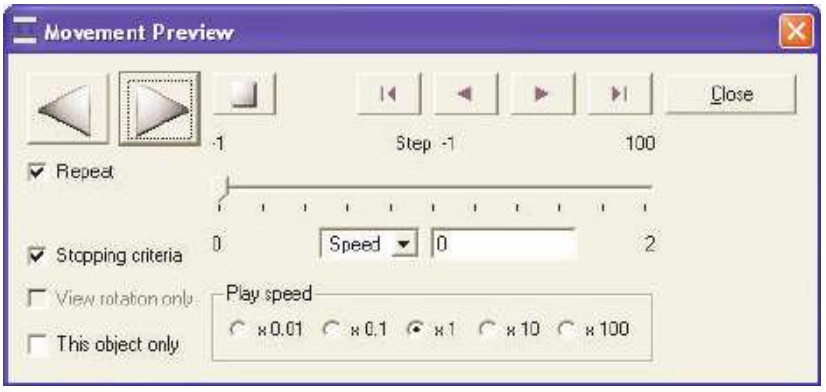


Figure 2.4.23: The movement preview dialog.

Clicking the preview button allows the user to see the movement that has been specified for a given object (See Figure 2.4.23). The buttons in the movement preview dialog allow the user to see the movement of the current object in the display screen. This will only take into account only translation and rotational movement but not force or torque control.

Translational Movement

During the simulation, the constrained nodes will move synchronously in the speed and direction defined by the movement controls. (See Figure 2.4.24).

Speed control

This is the default movement control. This specifies the speed and direction of a tool. Speed can be defined as a constant, or as a function of time or stroke. When an object is rigid, the entire object will move at the assigned speed. When the object is elastic, plastic, or porous, each node with a movement boundary condition assigned will maintain the assigned speed. Note that movement boundary conditions should never be assigned for all surface nodes. In general, no more than 1/2 to 2/3 of the boundary nodes on an object should have movement boundary conditions.

Symmetry planes are defined with V=0 boundary conditions perpendicular to the applied surface. Due to limitations of border extraction during remeshing, parallel symmetry planes should be defined using at least one rigid symmetry surface, instead of V=0 boundary conditions on both sides of the object..

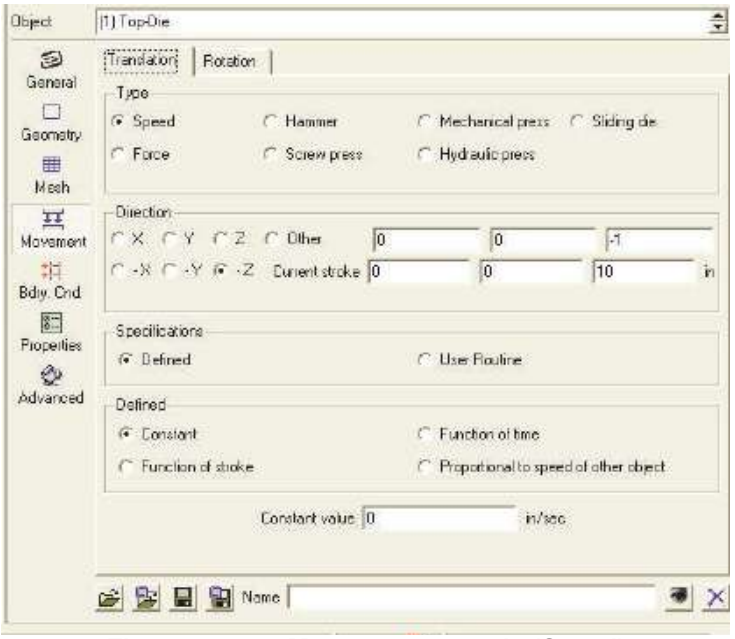


Figure 2.4.24: Movement controls (Speed)

Force control

For force control, the speed of the object is defined such that the specified load is maintained. When the object is rigid, the load is the resultant load applied by all non-rigid objects that contact it. When the object is elastic, plastic, or porous, the load is the sum of the nodal loads of all nodes with movement boundary condition codes defined. This boundary condition adds a degree of freedom to

the system of equations to be solved during the simulation. Arbitrary application of this condition can create difficultly in obtaining a converged solution. Please refer to the appendix section for guidelines on applying this movement in the most effective manner.

Hammer energy

Hammer forging operation is controlled by energy. During a working stroke, the deformation proceeds until the total kinetic energy is dissipated by plastic deformation of the material and by elastic deformation of ram and anvil when the die and ram faces contact each other. (see Figure 2.4.25)

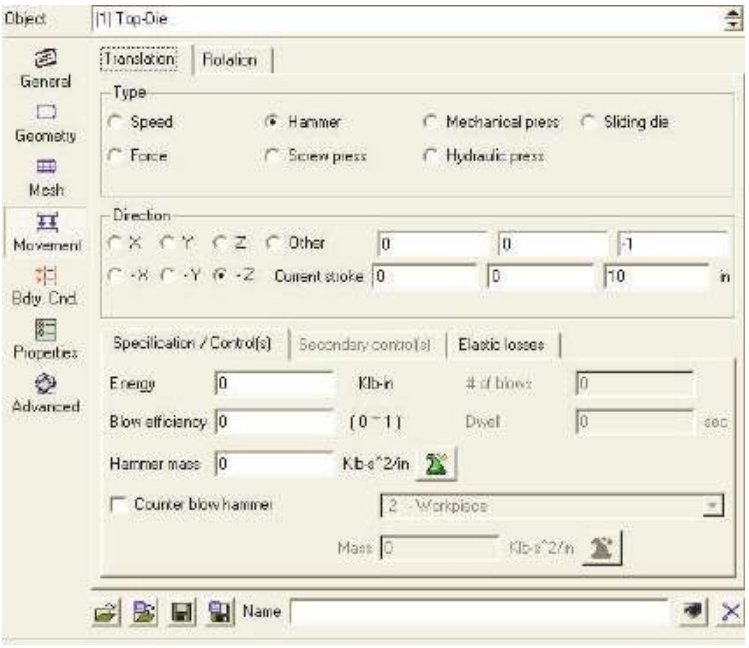


Figure 2.4.25: Movement controls (hammer press settings shown)

During hammer forging operation, only a portion of the kinetic energy of ram is used for the plastic deformation of work piece. The rest of the energy is lost through anvil and machine frame. The blow efficiency, η_B is defined by:

$$\eta_B = W_U / E_T$$

Where W_U is the energy consumed for the plastic energy of work piece and E_T the total kinetic energy of ram. These values can be set in the movement controls window (See Figure 2.4.25).

There are basically two types of hammer. The first is an anvil type hammer and the other counter blow hammer. The formulations and assumptions used for the two types of hammer forging operations are given below:

1. Anvil Type Hammer

In an anvil type hammer, the workpiece, together with the lower die set, is placed on an anvil which is stationary. In a simple gravity drop hammer, the ram is accelerated by gravity and accumulates energy. Therefore, the blow energy, E_T, is calculated by:

$$E_T = m_T g H$$

Where m_T is the mass of the ram, g the acceleration due to gravity and H is the ram dropping height. In a power drop hammer, the ram is accelerated by steam, cold or hot air pressure in addition to gravity. The total blow energy is given by:

$$E_T = (m_T g + p_m A) H$$

where A is the piston area and p_m is the indicated steam, oil, or air pressure on the piston. The velocity of ram, V_T, is calculated by:

$$V_T = \sqrt{\frac{2E_T}{m_T}}$$

The plastic deformation energy during a small time increment Δt is calculated by:

$$\triangle E_D = L_T \triangle s$$

Where L_T is the deformation load at the ram, ΔE_D is the energy consumed by the workpiece over the ram travel increment and Δs is the ram travel during Δt . After the increment, the blow energy is adjusted by:

$$E_T(t + \triangle t) = E_T(t) - \frac{\triangle E_D}{h_B}$$

This adjustment accounts for elastic energy loss. The simulation is repeated until the blow energy E_T becomes zero. The characteristic values of Anvil type hammers are given in the following table:

Hammer type		E, kN-m (ft-lbf)	nB	Vs, m/sec (ft/sec)	nB, min-1	nH
Free Drop Hammer	Belt	40(29,440)	0.3-0.6	4-5(13-16.4)	40	0.2-0.3
	Board	16(11,780)	0.3-0.6	4-5(13-16.4)	35	0.2-0.3
	Chain	100(73,600)	0.3-0.6	4-5(13-16.4)	55	0.5
	Piston	63(46,370)	0.3-0.6	4-5(13-16.4)	60	0.5
Power Drop Hammer	Pneumatic	50(36,800)	0.8-0.9	5-8(16.4-26.3)	80-250	0.45-0.55
	Open-die,single frame	40(29,440)	0.8-0.9	5-8(16.4-26.3)	450	0.45-0.55
	Open-die,double frame	250(184,000)	0.8-0.9	5-8(16.4-26.3)	55-240	0.5
	Die Forging	100(73,600)	0.3-0.6	5-8(16.4-26.3)	55-240	0.5

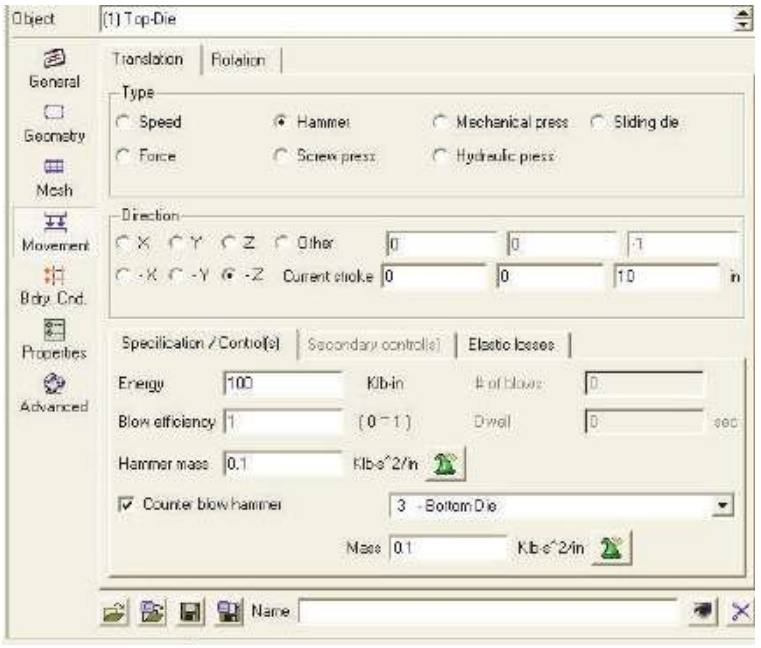


Figure2.4.26: Movement controls (Counter blow Hammer)

2. Counterblow Hammer

A counterblow hammer can be specified for movement by selecting the Counter blow hammer checkbox as seen in Figure2.4.26. After this, the other moving hammer object can be specified as well as the mass of the other moving hammer. The mass of the objects is not required to be equal but the total energy is always split between the two hammer dies.

Screw press

The unique characteristic of a screw press is the method of driving it. A motor drives a flywheel which is either directly connected or can be connected to a screw spindle. The screw spindle transmits the rotation through the threads, which have pitch angles usually between 13 and 17 degrees, to a linear movement of the main ram. On contact with the work piece, the complete kinetic energy of the flywheel and the ram is transformed into useful work (work on the work piece) and losses (elastic deformation work in the work piece and the frame of the structure and friction). The elastic deformation work results in a reaction force in all the press parts lying in the force transmission path.

The Screw press energy method will mimic the movement of a screw type press on the selected die. In a screw press a flywheel is taken to a given speed and a clutch is engaged. Once the clutch is engaged, the screw press begins to draw energy to drive the screw from the flywheel. Once the flywheel energy is

expended the stroke is over and the movement will stop. Screw controlled movement can only be specified for rigid objects or deforming objects with a movement boundary condition applied. Motion of which is controlled by screw press parameters can only be applied in the +X, +Y, +Z, -X, -Y or -Z directions. The data required to run a screw press driven tool is:

1. **Blow Energy:** The Blow Energy is a measure of the total energy that the flywheel will contain when the desired speed has been reached and prior to engaging the clutch. The units for blow energy are klb-inch in English units and N-mm for SI units.

2. **Blow Efficiency:** The Blow Efficiency represents the fraction of the total energy that will be converted to deformation energy. The rest of the energy is absorbed through the clutch mechanism, friction, and the machine frame. There are no units for this quantity.

3. **Moment of Inertia:** The Moment of Inertia is the moment of inertia of the flywheel. The English units of inertia are lbf sec2 inch, and the SI units are kg m2. The mass moment of inertia for a circular disc with the Z-axis perpendicular to the center is $I = 2\ E_T/\omega^2$ where E_T is the total energy of the flywheel, and ω is the angular velocity in radians per second.

4. **Ram Displacement:** The Ram Displacement specifies the distance per revolution of the flywheel that the screw will advance. This helps in determining the linear velocity of the ram. The English units for Ram Displacement are inch/revolution, while the SI units are mm/revolution. If only the pitch angle and diameter of the spindle is known, the Ram Displacement can be calculated using $\pi d\sin(\theta_t)$ where d is the diameter of spindle and θ_t is the pitch angle of the spindle.

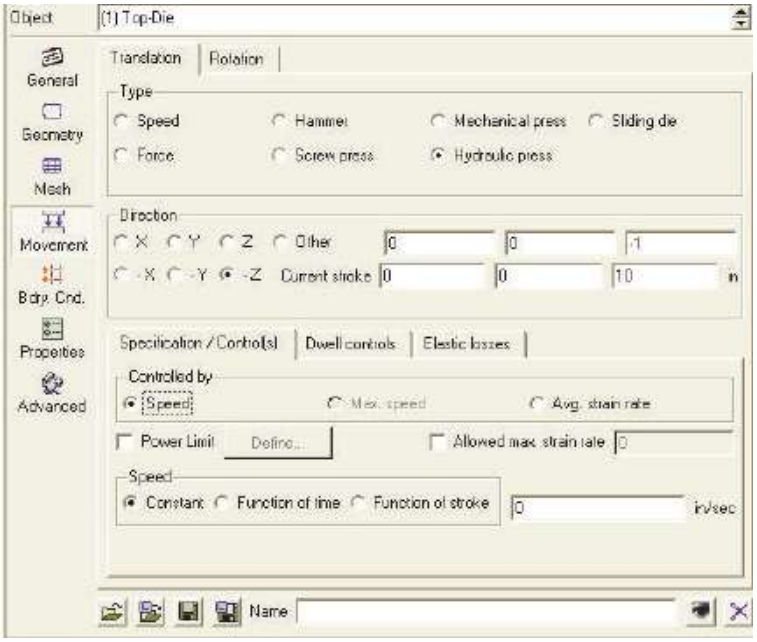


Figure 2.4.27: Movement controls (Hydraulic controls – Speed control)

Hydraulic press

To use the hydraulic press model the user can control one of two manners.

1. **Speed:** The speed of the press can be input as a constant or as a function of time or stroke (See Figure 2.4.27). With this, the power limit of the hydraulic press is also entered in the familiar table format as a function of load. See Example Lab 21: Hydraulic Press Simulation for an example of a simulation using the hydraulic press model.

   Note: To activate maximum speed control, the power limit must be defined.

2. **Average strain rate:** The average strain rate control can also used to define the press speed (See Figure 2.4.28). Note that the initial billet height needs to be provided and should be a reasonably accurate measurement.

Allowed maximum strain rate: This setting can be defined in addition to the controls mentioned above. This will prevent the speed from exceeding a condition where the maximum strain rate in the part would exceed the defined maximum strain rate.

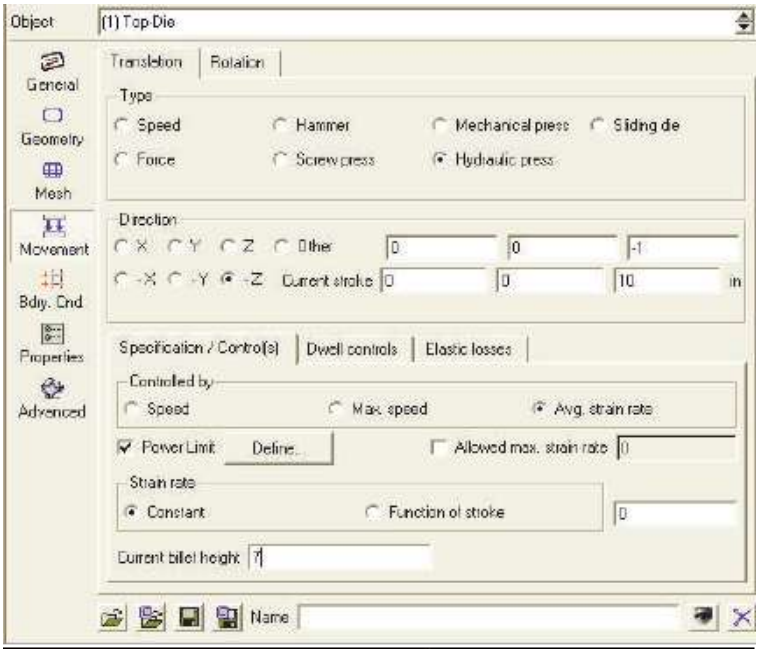


Figure 2.4.28: Movement controls (Hydraulic controls – Average strain rate control)

Mechanical press

The Mechanical Press type replicates the cyclic motion of a mechanical press. The parameters required to simulate the motion are the total displacement of the press (D_{tot}) relative to the current displacement (D_{cur}), and the number of strokes per unit of time (S'). (see Figure2.4.29) Using these parameters, DEFORM can compute the die speed at any point of the travel of the die. The movement direction can only be specified in the X, Y, Z, -X, -Y or -Z directions.
The equation to derive the die speed is:

$$V = 2pS' D_{cur}\sqrt{\frac{D_{tot}}{D_{cur}} - 1}$$

The parameters required to specify the movements of a mechanical press are:

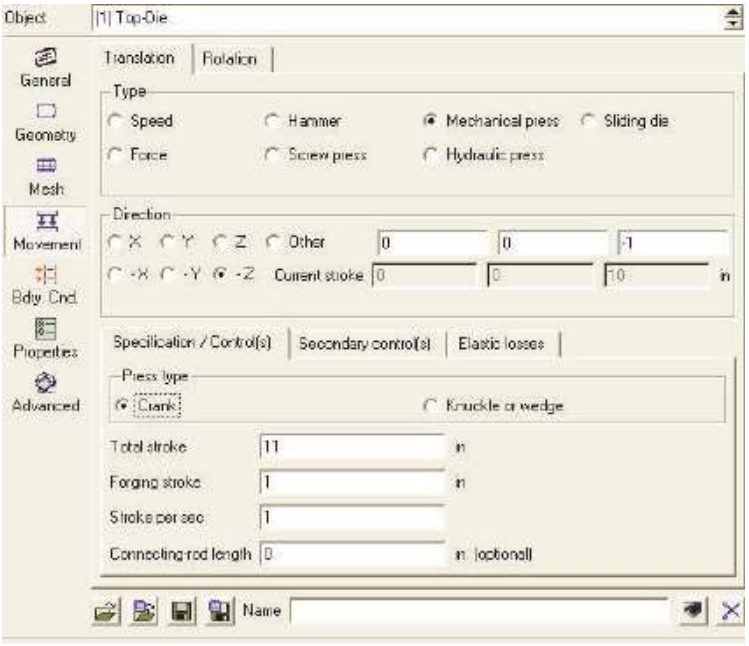


Figure2.4.29: The mechanical press settings window

1. **Total stroke:** The total stroke for the mechanical press represents the total travel of the die from its top position to its lowest position. In English units this is inch, in SI units this is mm.

2. **Strokes per second:** The Strokes per seconds represents the frequency of the press blows. This is a measure of blows per second or cycles per second.

3. **Forging Stroke:** This value is total remaining distance of the die in the given stroke. This value will depend on the current position of the moving die relative to the stationary die.

4. **Direction:** Direction is used to designate a direction in which the object's stroke will be applied.

5. **Connecting-Rod Length:** As seen in Figure 2.4.30, the connecting rod length can have an influence on the speed of the ram. If the length of the connecting rod is known, it can be input as a field. If it is not known, it can left as zero and its contribution to the ram speed will not be considered.

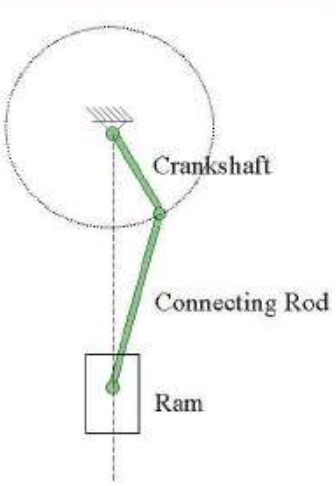


Figure 2.4.30: Sketch of a simple direct crank drive.

Knuckle or Wedge Press

In the case of less common presses, there is a capability to model their movement by defining them strictly as a velocity profile. The velocity profile is defined as an angle (in degrees) versus speed. As an angle, this has to do with the angle of rotation of a driving motor.

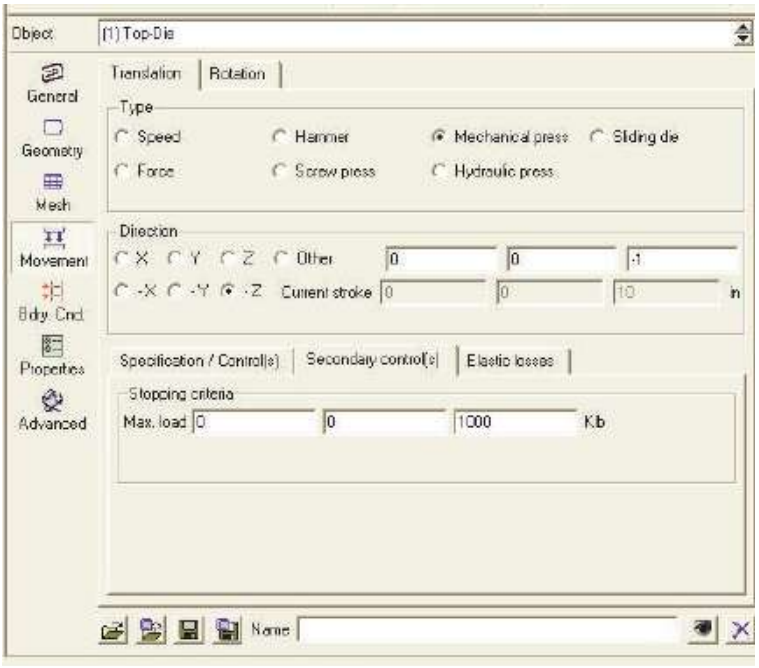


Figure 2.4.31: Secondary controls for mechanical press

Secondary Control(s)

The type of control will depend on the type of movement specified. In the case of a mechanical press (as seen in Figure 2.4.31) the only control is based on load. In the case of a hammer, there are no secondary controls since the only manner in which to stop is to run out of energy. In the case of a hydraulic press, it can stop based on the load or a minimum velocity.

Sliding Die Movement

Defining sliding movement can be done in the movement controls window as seen in Figure 2.4.32. To use spring-loaded dies, the object should be rigid and should not have any other movement specified. The following items should be specified for all spring-loaded die cases:

1. Turn spring-loaded die controls to 'On'.
2. Specify the compression direction – Spring-loaded dies only give a reaction force in the compression direction.
3. Stiffness – This is specifies the stiffness in klb/in (English) or N/mm (SI) units. The stiffness can be either a constant or a function of compression amount.
4. Preload – This specifies the amount of force to overcome before compression occurs in klb (English) or N (SI).
5. Maximum displacement – This is the distance where the spring eventually bottoms out and which cause the spring to not move anymore in the compression direction.
6. Other end of spring – This determines whether the spring is fixed to another rigid object or whether the other end is fixed to a position in space. In the case where the object is fixed to a position in space, the compression direction, the current displacement and maximum displacement determine how much the spring is compressed and whether it is bottomed out. In the case where the spring is attached to another object, the distance between the two objects will determine the amount of compression.

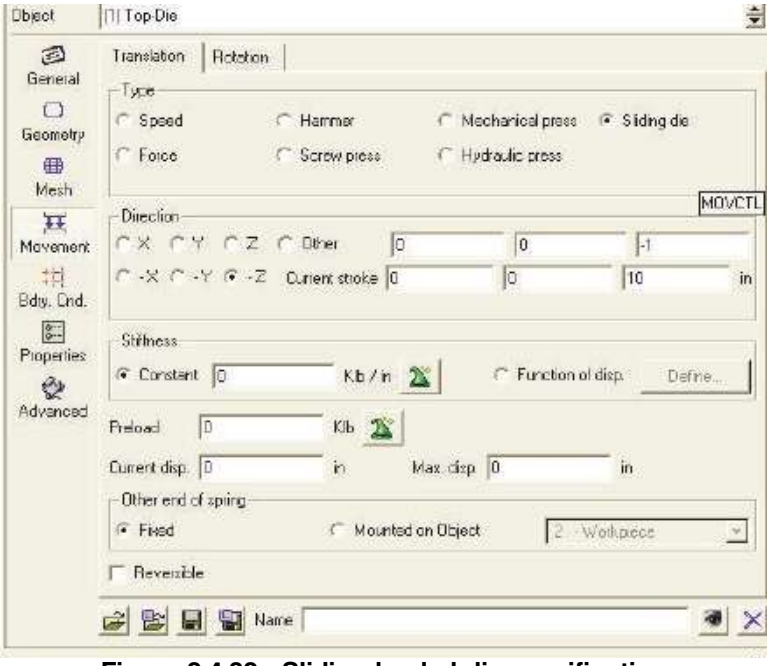


Figure 2.4.32: Sliding-loaded die specification

Rotational movement

Rotational movement is defined by an angular velocity about a fixed center of rotation (See Figure 2.4.33). This movement type causes only rotation. Unless otherwise specified, translation is constrained. The rotational speed is controlled through the Controlling Method option and the point at which the object is rotated about is set through the Center of Rotational Movement.

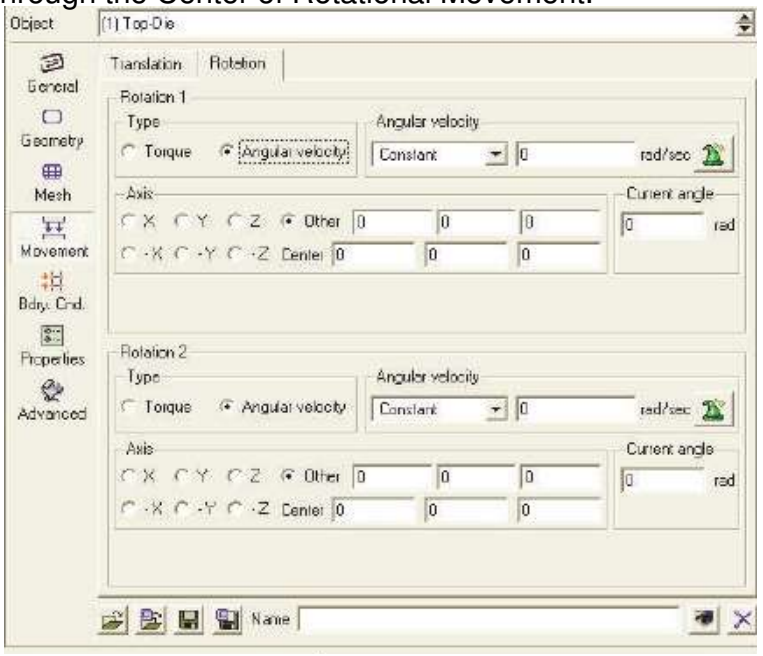


Figure 2.4.33: Movement controls (rotational movement).

Rotational Motion can be applied to simulate rolling or any type of movement where an object will rotate about a fixed axis. Rotational Motion can only be applied to Rigid objects. Rigid objects can have both Rotational and Translational movement simultaneously.

1. **Controlling Method** The objects rotation can be controlled by an Angular Velocity or a Torque. Select the required control and enter the value for the rotation just below.
Torque The Torque movement type will apply a rotational motion about the defined axis at a specified torque. The torque can be specified as a constant or as a function of time or angle.
Angular Velocity Angular Velocity will apply rotational motion about the defined axis at a specified angular velocity in radians per second. The angular velocity can be specified as a constant or as a function of time or angle.
NOTE: Idle rolls can be defined by specifying torque control with a very low torque value

2. **Axis of Rotation** is specified by a vector originating at the point *center* and through the point *direction*. Rotation direction is defined by the right hand rule. That is, positive rotation is clockwise as viewed from the center point looking towards the direction point.

Movement control user subroutines
Complex die movement can be defined using user defined FORTRAN subroutines. Please refer to later in the manual for a description of how to implement user defined subroutines.

2.4.13. Object node variables

The nodes window displays all available information about nodes. This window can be reached through the object->advanced properties window (See Figure 2.4.34) by clicking the Node Data button (See Figure 2.4.35). All information can be modified, and many of the variables can be plotted. Features of the display window include:

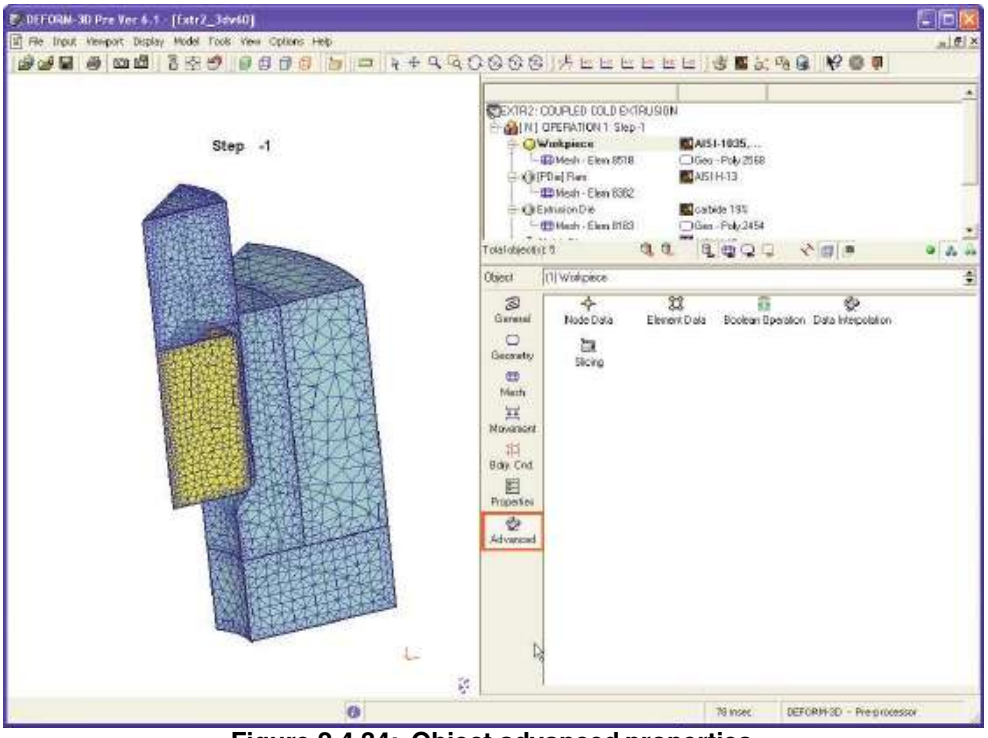


Figure 2.4.34: Object advanced properties.

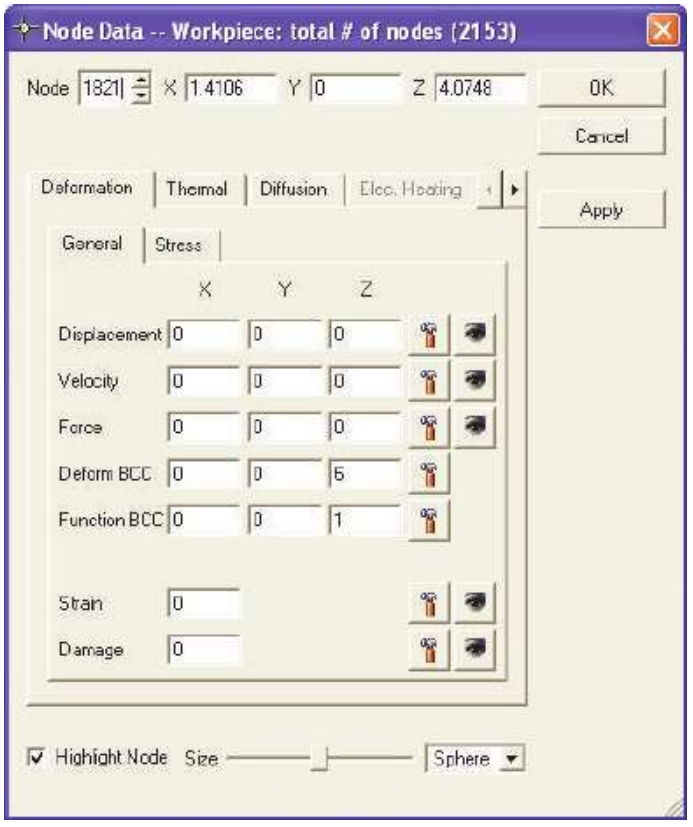


Figure 2.4.35: Node data window (deformation data).

Node number

The number of the node for which information is being displayed. A node may be selected either by keying in the value or by using the *select* icon and mouse-picking a node.

Show node numbers

Toggles graphic display of all node numbers

Initialize values

Initialize the specified variable value on all nodes in the object.

Plot variable

Produces contour or vector plot of selected node variables

Line contour / Shaded contour

The plot type can be selected at the top toolbar.

Node Coordinate

The coordinates of the node are displayed. The values can be modified to slightly adjust the position of nodes where boundary conditions were not properly enforced. This should be done with *EXTREME* caution. For features such as boundary extraction and mesh generation, the database must be saved, and the newly saved step read into the preprocessor before the adjusted coordinates can be used.

Deformation

Displacement (DRZ)

Displacement stores the displacement of each node since the last remesh. For elastic objects, a displacement may be specified for interference fits. The elastic recover of the object will cause the appropriate stress values to be developed.

Velocity (URZ)

URZ is the X, Y, and Z velocity components of each node.

Force (FRZ)

FRZ specifies the value of the constant nodal force at individual nodes.

Pressure (PRZ)

PRZ maintains a specified normal pressure or shear traction across the face of the elements lying between the selected boundary nodes.

Boundary condition code (BCCDEF)

BCCDEF specifies the deformation boundary condition in X, Y, and Z.

The code values are:

0

Node force specified

1

X, Y, or Z component of node velocity constrained, corresponding to the X, Y, or Z component of BCCDEF

2

Constrained node tractions specified by PRZ

3

Node movement control defined

-n

Node is in contact with object number N. Note: There is no significance to the X, Y, or Z component of contact. Contact values are stored in the first free component, starting with Z, then Y, then X.

Boundary condition functions (BCCDFN)

BCCDFN specifies if the value of a deformation boundary constraint (nodal velocity, force, or traction) associated with a particular node is to be specified as a constant or as a set of time/nodal value data.

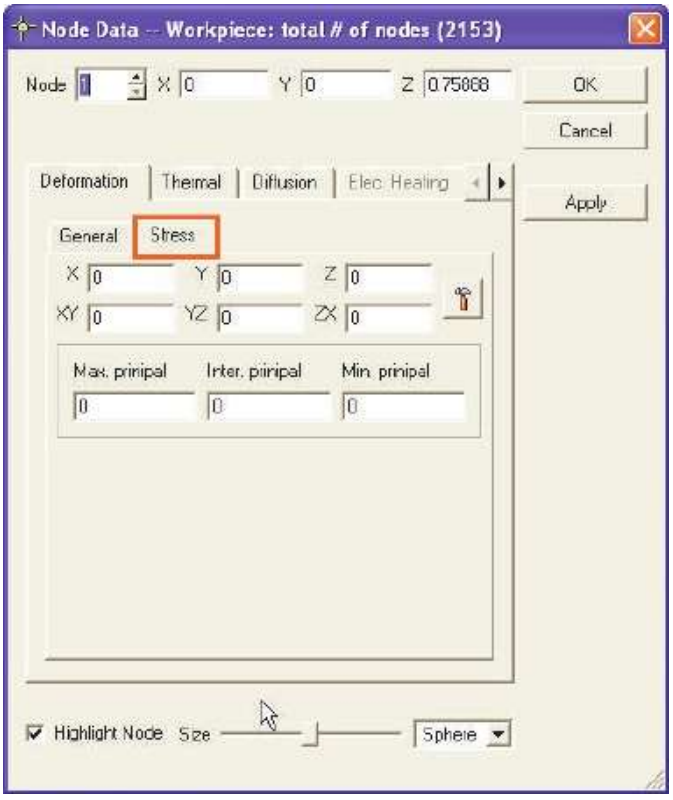


Figure 2.4.36: Node data window (Stress).

Stress (STRESS)
When nodal definition of stress (for elasto-plastic objects), strain and damage is set in Simulation controls: Advanced: Out put Controls, nodal values of the same can be examined from this nodal data dialog. (see Figure 2.4.35and Figure 2.4.36).

Thermal
All the nodal thermal data of the model can be examined, defined or initialized from this dialog. (see Figure 2.4.37).

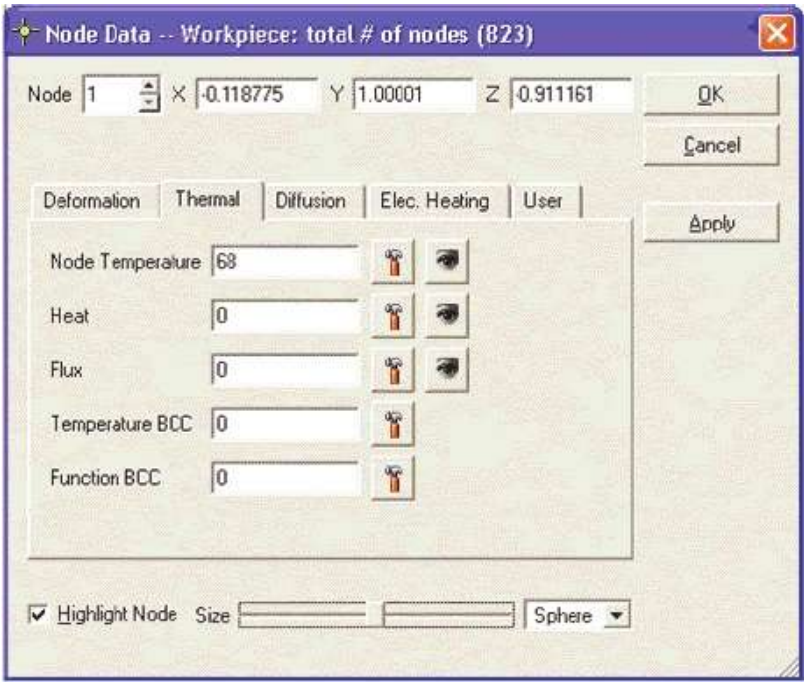


Figure 2.4.37: Node data window (thermal data).

Node Temperature (NDTMP)
NDTMP specifies the nodal temperature to be applied to individual nodes.

Heat (NDHEAT)
NDHEAT specifies the nodal heat to be applied to individual nodes.

Heat flux (NDFLUX)
NDFLUX specifies the distributed nodal heat flux to be applied to individual nodes. The heat flux constraint will be applied to the element faces lying between the selected boundary nodes.

Boundary condition code (BCCTMP)
BCCTMP specifies the heat transfer boundary constraint code for individual nodes.

1

Constant nodal temperature is specified

2

Heat transfer with environment boundary condition

3

Specified nodal heat flux

Boundary condition function (BCCTFN)
BCCFNC is used to specify time/nodal value pairs for nodal boundary constraints. Types of nodal boundary constraints that can be specified as time/nodal value pairs include velocity, force, traction, temperature, heat, and distributed heat flux. BCCFNC can only be used when a node's boundary constraint function type, BCCDFN or BCCTFN, has been specified as a time/nodal value type.

Diffusion [DIF]

For all the model with diffusion turned in the simulation controls, the nodal diffusion data for the object can be examined, defined or initialized in this dialog. (see Figure 2.4.38)

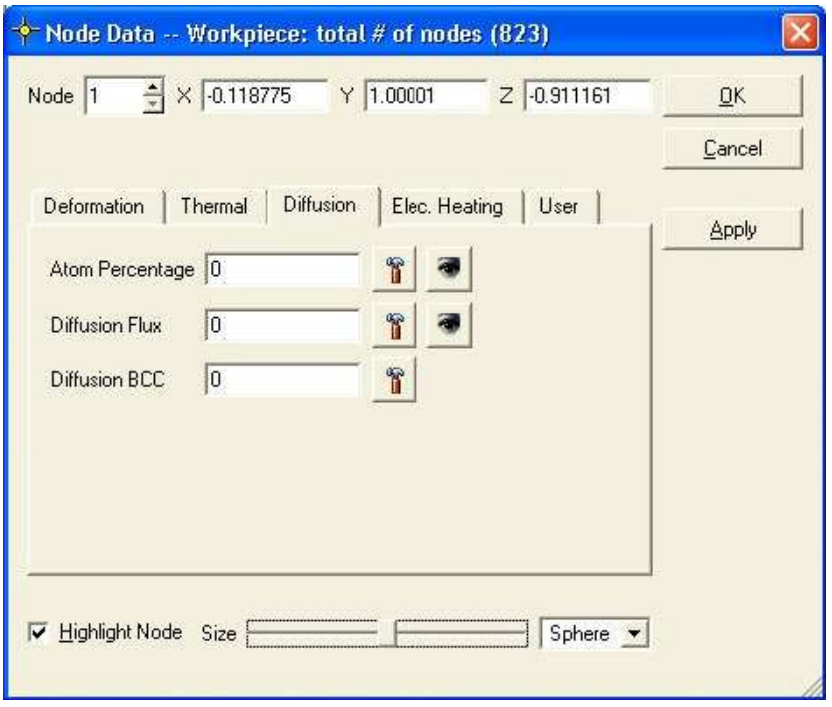


Figure 2.4.38: Node data window (diffusion data).

Dominant atom content (DATOM)
DATOM specifies atom content at a node.

Atom flux (CRBFLX)
CRBFLX specifies "carbon flux" or atom flux for the surface of a work piece.

Boundary condition code (BCCCRB)
BCCCRB specifies the atom transfer boundary constraint code for individual nodes.

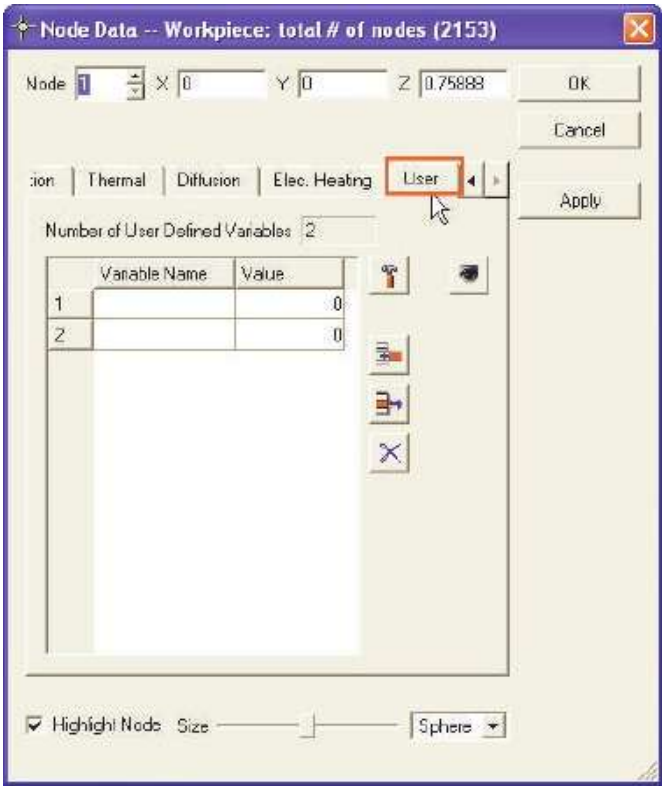


Figure 2.4.39: Node data window (user-defined data).

User variables

User node variables (USRNOD)

User node variable values can be defined using FORTRAN subroutines. Refer to a later section for more information on the subroutines. Each node value may accept both a name and a value (see Figure 2.4.39). Also, an infinite number of variables may be defined. A minimum of 2 user node variables will be defined by default, however, the user may increase this to as large number as wished. The only caveat is a large number of variables defined can lead to a large database file. The user specified nodal variables can be viewed, initialized, modified, or point tracked.

2.4.14. Object element variables

From the element data dialog user can examine a complete set of elemental data that is applicable to a given model (See Figure 2.4.40). This data includes, stress, strain, strain rate, material axis, hardness, transformation data, heating data, grain data and user defined elemental variables.

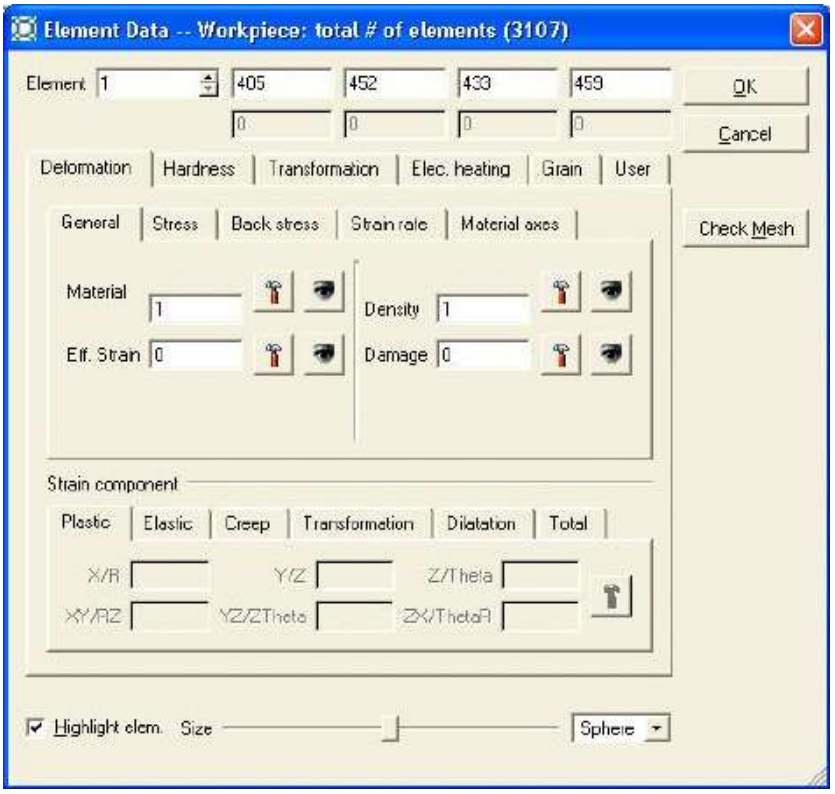


Figure 2.4.40: Element data window (deformation data).

Connectivity (ELCON)
The element connectivity specifies the numbers of the 4 or 8 nodes which define the corners of the element.

Deformation

Material group (MTLGRP)
MTLGRP specifies the material number associated with each element.

Relative density (DENSTY)
DENSTY specifies the relative density of the material at each element. DENSTY is used when a porous material with relative densities less than 1.0 is being simulated. If no value is specified for density, it is assumed to be 1.0. The flow stress of porous objects should be specified for the fully dense material.

Effective strain (STRAIN)
STRAIN specifies the value of total effective strain at the centroid of each element. Elemental strains are interpolated between meshes during remeshing procedures.

Damage (DAMAGE)
DAMAGE specifies the damage factor at each element. The damage factor can be used to predict fracture in cold forming operations. The damage factor increases as a material is deformed. Fracture occurs when the damage factor has reached its critical value. The critical value of the damage factor must be determined through physical experimentation. Damage factor, D_f, is defined by

$$D_f = \int \frac{\sigma^*}{\sigma} d\epsilon$$

where σ^* is the tensile maximum principal stress, σ is the effective stress, and $d\epsilon$ is the effective strain increment.

Stress components (STRESS)
Defines the stress tensors of each element of an object. The values as displayed are σ_x, σ_y, σ_z, σ_{xy}, σ_{yz}, σ_{xz}.

Yield surface translation tensor (YLDS) [MIC]
YLDS specifies the yield surface translation tensor for kinematic hardening.

Hardness [MIC]

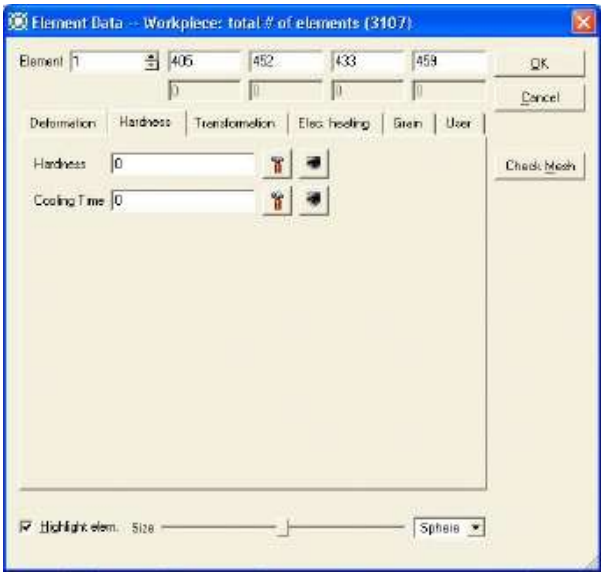


Figure 2.4.41: Element data window (hardness data).

Hardness (HDNOBJ)
HDNOBJ1 specifies hardness and HDNOBJ2 cooling time from high temperature to low temperature for each element. (see Figure 2.4.41)

Transformation [MIC]
All the phase volume information and transformation data can be examined for every element, or initialized as well for the model. (see Figure 2.4.42)

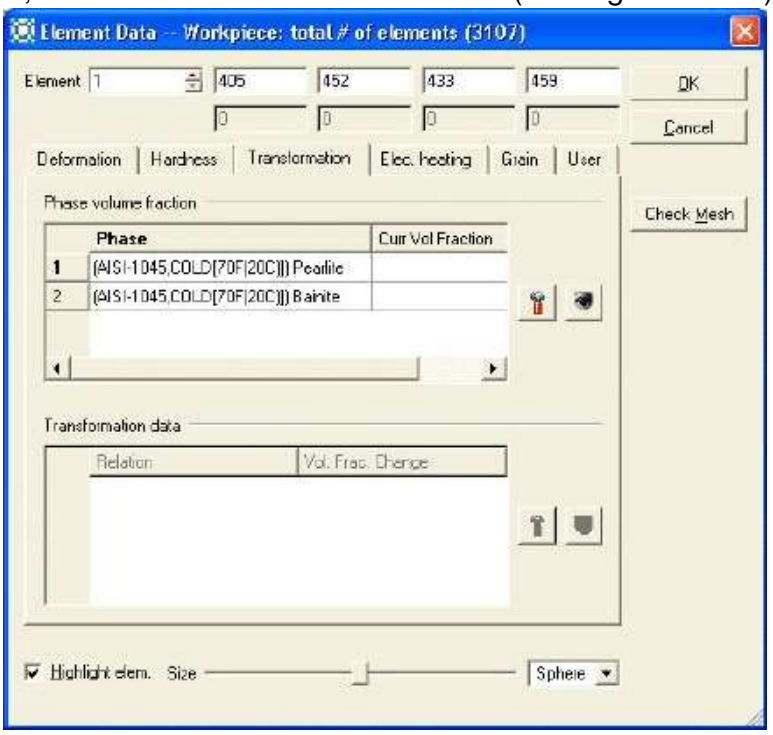


Figure 2.4.42: Element data window (transformation data).

Current volume fraction (VOLFC)
VOLFC specifies the initial volume fraction of a phase (material) in an element at the beginning of a simulation. In addition, throughout the simulation, VOLFC stores the volume fraction of all phases in each element per step. The volume fraction is determined from the keyword TTTD, which specifies the model or data used in calculating the volume fraction of each phase. It is important that the user specifies the necessary input for the keyword TTTD or else the volume fraction (VOLFC) will not be calculated for the object. The user must input the type of diffusion model and at least two Time-Temperature curves, the beginning of the transformation and the end of the transformation.

Starting volume fraction (VOLFS)
VOLFS specifies the volume fraction limit of a phase in an element of an object. VOLFS is only stored in the database at the beginning of a new phase transformation, ex. Austenite -> Pearlite or Austenite -> Martensite. The intent of

VOLFS is to assure that the volume fraction amount transformed from Austenite -> Pearlite does not exceed the volume fraction of Austenite prior to transformation. It should be noted that VOLFC is different than VOLFS in that VOLFC stores the volume fraction of the phases every step. Typically, the user will only be concerned with inputting volume fractions for VOLFS at the beginning of the simulation.

Incubation time (TICF)
TICF (consumption of incubation time) specifies the fraction of time that has occurred before the transformation has started. It is calculated for diffusion type transformation as follows:
TICF = where is the time increment and is the incubation time at temperature .The variable total is temperature dependent. If TICF reaches, the transformation starts.

Volume fraction change (DVOLF)
DVOLF specifies the change in volume fraction of all the different phases that resulted from the transformation during each time step. DVOLF of each phase is initially set to zero in a simulation. DVOLF is determined by for each step where f is the volume fraction of a particular phase. DVOLF is used in calculating the latent heat due to transformation and the change in volume of the object. DVOLF can be invaluable in determining the progress of a transformation and aid the user in deciding whether to increase or decrease the time step for a particular transformation.

Grain size (GRAIN)
Current this is not implemented

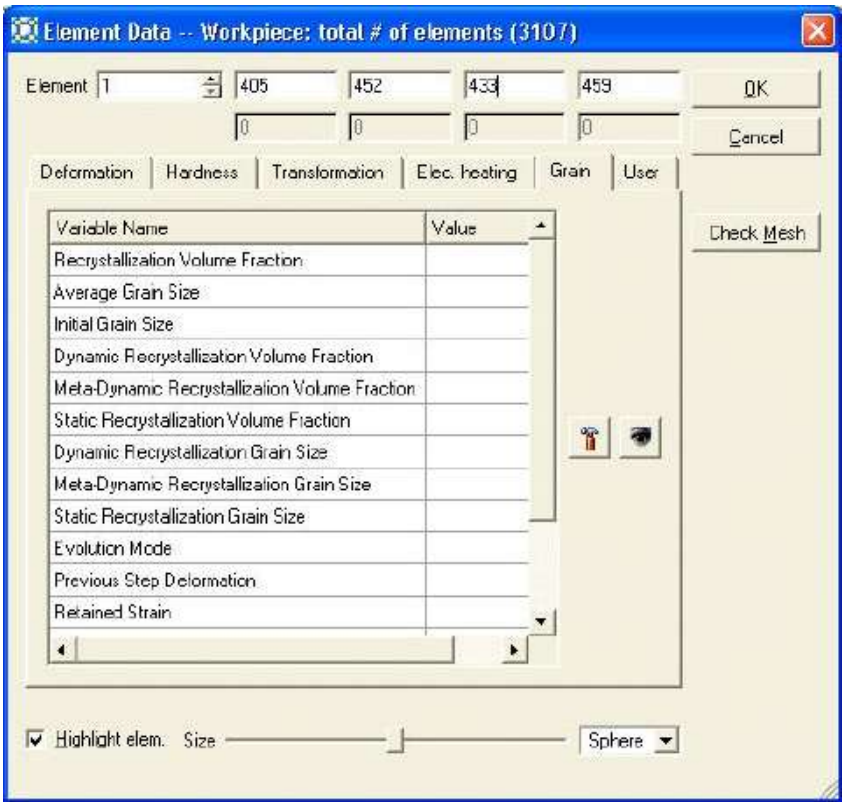


Figure 2.4.43: Element data window (grain data).

User element variables (USRELM)

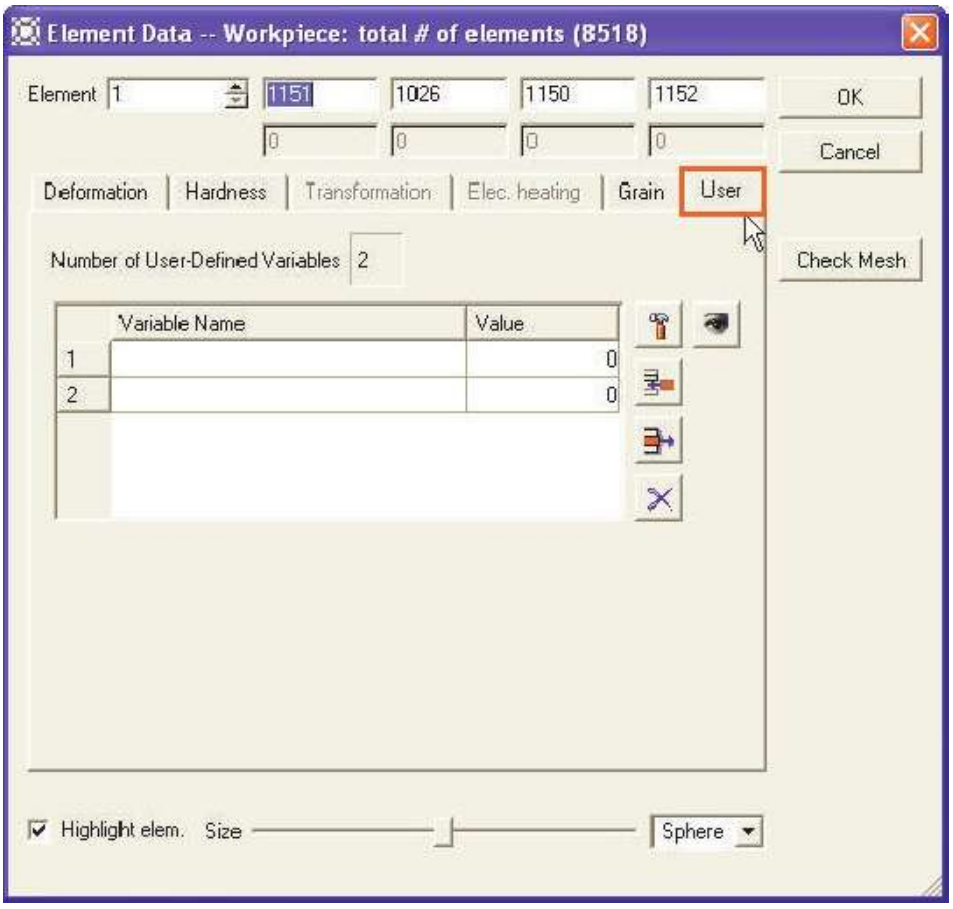


Figure 2.4.44: Element data window (user element data).

User element variable values can be defined using FORTRAN subroutines. Refer to a later section for more information on the subroutines. Each element value may accept both a name and a value (see Figure 2.4.44). Also, an infinite number of variables may be defined. A minimum of 2 user element variables will be defined by default, however, the user may increase this to as large number as wished. User needs to be cautious that a large number of variables defined can lead to a large database file. The user specified nodal variables can be viewed, initialized, modified, or point tracked.

2.4.15. Boolean Operation

This capability allows the user to subtract volume from the mesh of an object from the geometry of another object. At this time, only subtraction is available and the item to be subtracted from must be a meshed object and the item performing the subtraction must be a geometry. (see Figure 2.4.45)

Definitions:

Meshed object: An object with a solid mesh giving the object a volume definition.

Geometry object: An object with a surface definition only.

Note that some objects can have both definitions such as a rigid object with thermal calculations turned on. The geometry is used for deformation calculations and the mesh is used for thermal calculations. In the case of a plastic, elastic, porous or elasto-plastic object, the geometry is only used for meshing purposes and is not used for simulation calculations.

The Boolean operation dialog is available through the advanced dialog and the object to be subtracted from should be selected and the Advanced->Boolean Operation button should be selected. After this, the following dialog should be seen. There are two ways in which to perform this Boolean calculation:

a. Boolean with respect to another object – the user should pick the object to Boolean from. After this please click Apply.

b. Boolean with respect to a plane that can be defined - the user should pick the Plane and then define a point and a normal to Boolean from the object. After this please click Apply.

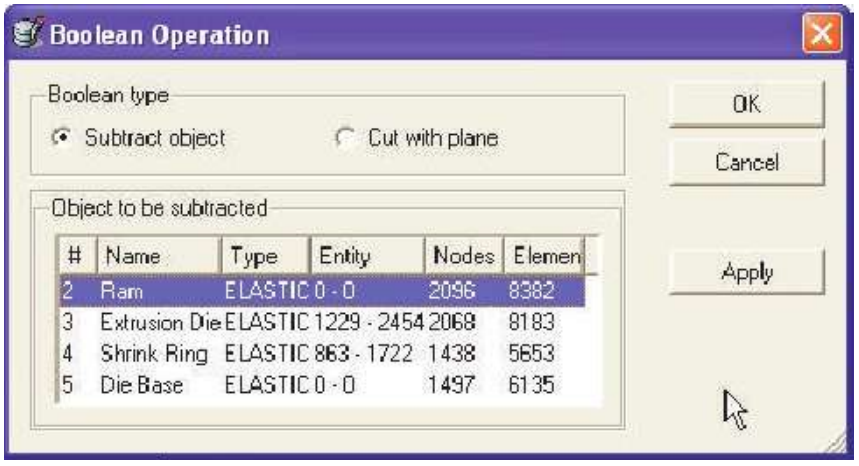


Figure 2.4.45: Boolean Operation Dialog

Note: In some rare cases where the Boolean definition corresponds exactly to the node locations, some nodes can cross the Boolean plane you have defined. In this case, slightly tweaking the position of the plane will solve this (as little as 1e-6 inches or mm).

What to do after Boolean operation

The cut planes will show a poorly structured mesh. This is because many elements were cut. To improve the mesh quality perform the following actions:

1. Press Ok when the Boolean operation is satisfactory.
2. Go to Geometry
3. Extract the mesh (The purpose is to prevent any geometry definition from being used for the meshing)
4. Go to mesh and mesh the object (Do not perform a manual remesh).
5. Go to Advanced->Data Interpolation to restore the state variables.

After this, you should have successfully updated your part with the desired volume removed.

2.4.16. Data Interpolation

While doing the manual remeshing in the preprocessor, user can transfer data from another object from a different database using this dialog. (see Figure 2.4.46). Once the database is selected user can select the object, and step number from where the data needs to be interpolated.

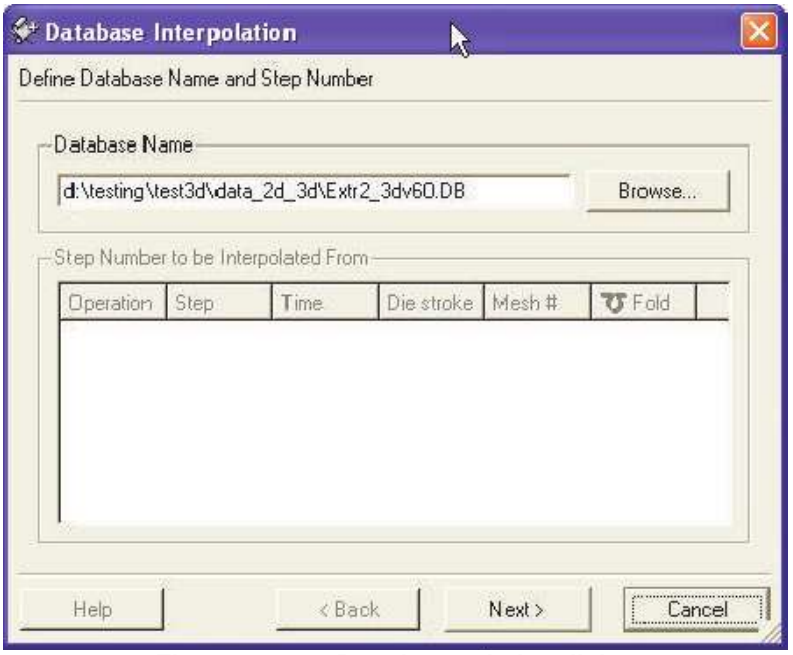


Figure 2.4.46: Data interpolation dialog

2.4.17. Slicing

This utility enables the users to slice an object and save the 2D cross section, either as geometry or as a keyword file including the state variable data from the slicing plane, (see Figure 2.4.47)

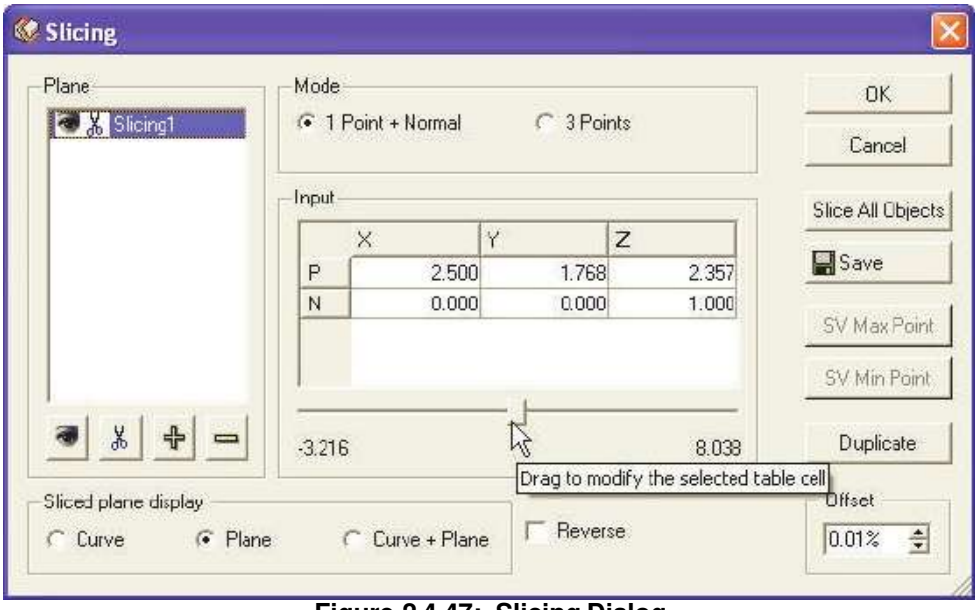


Figure 2.4.47: Slicing Dialog

2.5. Inter Object Definition

Inter Object data definition page can be accessed from the following icon (top row of icons)



Contact relation (CNTACT)
The contact relation parameter is used to set the Master/Slave relationship between work piece, dies, and deformable bodies. The Slave object should be the softer of the two materials and should also have the finer mesh. In the case of two objects consisting of the same material, either can be the slave although the object expected to elastically deform the most should be defined as the slave. Setting a ``No Contact'' relation causes the objects to be invisible to each other and allows them to pass through each other uninhibited.
CNTACT should be specified between every pair of deformable objects that may contact each other during the simulation, and between deformable objects and tools which they may contact.

Note: When a node from one deformable object contacts the surface of another deformable object, a relationship between the two objects must be established to keep the objects from penetrating each other. This relationship is referred to as a master-slave or slave-master relationship. When two objects are contacting each other, the contact nodes move on the master surface as long as the two objects are in contact. The slave nodes are considered to be in contact with the master object as long as the nodal forces indicate a compressive state. When a slave node develops a tensile force larger than the value specified in SEPRES, the node is allowed to separated from the master object.

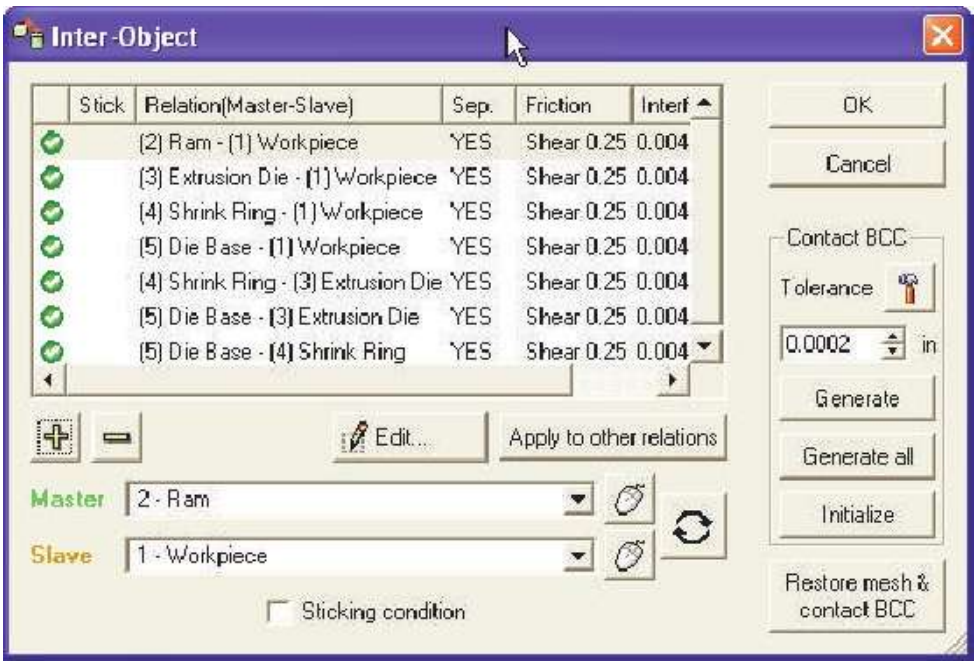


Figure 2.5.1: Inter-object window.

The purpose of inter object relations is to define how the different objects in a simulation interact with each other. The relations table shows the current inter object relations that have been defined. All objects which may come in contact each other through the course of the simulation must have a contact relation defined (see Figure 2.5.1). This includes an object having a relationship to itself if self-contact occurs. It is very important to define these relationships correctly for a simulation to model a forming process accurately. The critical variables to be defined between contacting objects are:

- Friction factor
- Interface heat transfer coefficient
- Contact relation
- Separation criterion

The inter object controls also contain object positioning controls and inter object boundary condition generation.

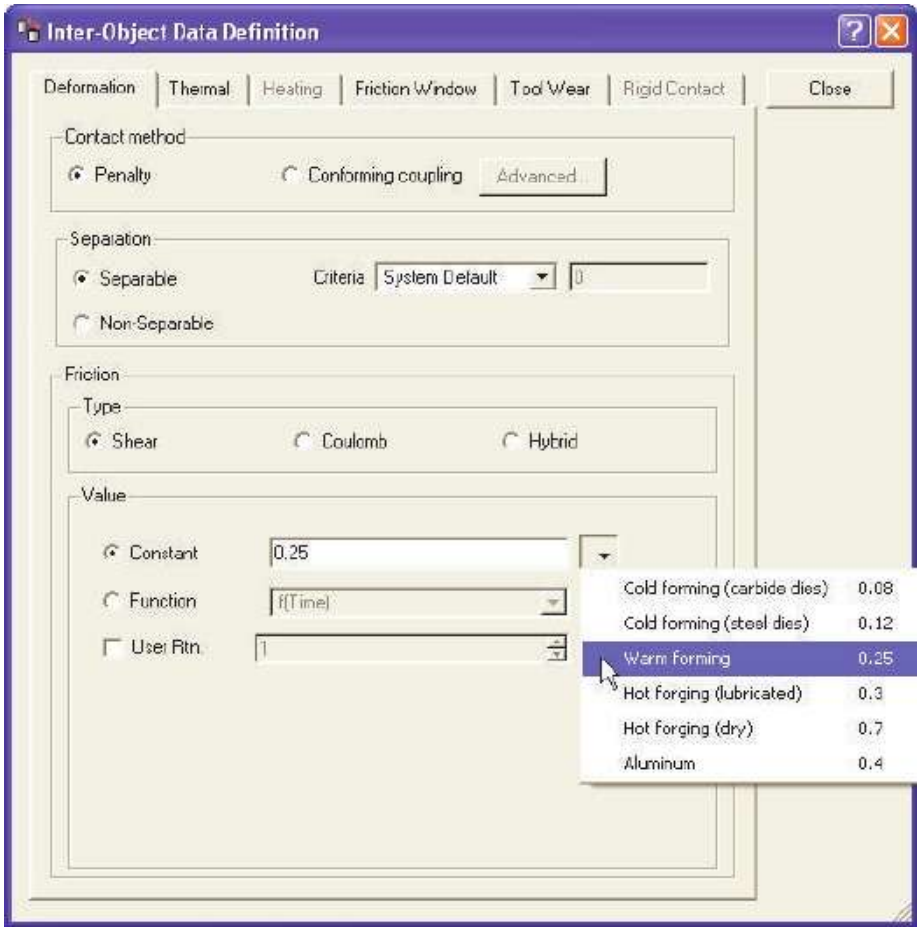


Figure 2.5.2: Inter-object window (deformation settings).

2.5.1. Inter object Interface

The inter object interface defines what objects can contact each other, and how contacting objects will behave while in contact. Contact relations, inter object boundary conditions, friction and heat transfer relations are set here for each object pair. (See Figure 2.5.2).

Friction (FRCFAC)

The friction coefficient specifies the friction at the interface between two objects. The friction coefficient may be specified as a constant, a function of time, or a function of interface pressure. The friction types allowed are shear and coulomb friction.

Shear (sticking)

Constant shear friction is used mostly for bulk-forming simulations. The frictional force in the constant shear model is defined by

$$f_s = m\,k$$

Where
f_s is the frictional stress,
k is the shear yield stress and
m is the friction factor.
This states that the friction is a function of the yield stress of the deforming body.

Coulomb
Coulomb friction is used when contact occurs between two elastically deforming objects (could include an elastic-plastic object, if it is deforming elastically), or an elastic object and a rigid object; generally to model sheet forming processes. The frictional force in the Coulomb law model is defined by

$$f_s = \mu p$$

Where
f_s is the frictional stress,
p is the interface pressure between two bodies and
μ is the friction factor.
There must be interfacial pressure between two bodies for frictional force to be present. If two bodies contact each other, however there is no force pressing the bodies together, there will be no resulting friction.
For contact between two plastic or porous objects, the frictional stress is calculated using the flow stress of the slave object.

Common Question: What is a good value for a friction coefficient?
Answer: The lubricant used on the tooling plays a large role in the amount of friction that exists between the tooling and work piece. The friction in turn affects the metal flow at contact surfaces.
Typical values (using constant shear only, as shown in Figure 2.5.1)
(0.08-0.1) for cold forming processes
(0.2) for warm forming processes
(0.2 to 0.3) for lubricated hot forming processes
(0.7-0.9) for unlubricated surfaces
Most processes are not extremely sensitive to friction, and the typical values listed above are perfectly adequate. For processes which are very sensitive to lubrication conditions, friction values may be determined by experimentation.

Remarks: Two simple ways of gauging the sensitivity of the process to friction:

1. Would you expect significant variation in the part depending on whether lubricant is applied well or applied poorly in production? If you would not, then the typical friction values listed above should be adequate.

2. If you are still unsure, run two DEFORM simulations with, say, a 20 % variation in friction conditions from the typical values. (For lubricated cold forming, you might run one simulation at 0.08 and one simulation at 0.12). Compare the results, such as load versus stroke or final geometry, particularly the parameters you identified as critical. If there is substantial variation, more careful study of friction is warranted. If there is little variation, then the typical values are adequate.

As indicated in Figure 2.5.3to Figure 2.5.11, in addition to these basic shear and coulomb friction models, DEFORM offers a variety of definitions to model accurate interaction of deforming work piece with other physical components of the system under varying processing conditions. These include definitions of the friction values as a function of time, interface pressure, interface temperature and surface stretch of the deforming work piece or a combination these. Additional definitions also include explicit models that define friction values as a function of pressure, strain rate and sliding velocity as indicated in the Figure 2.5.3to Figure 2.5.11

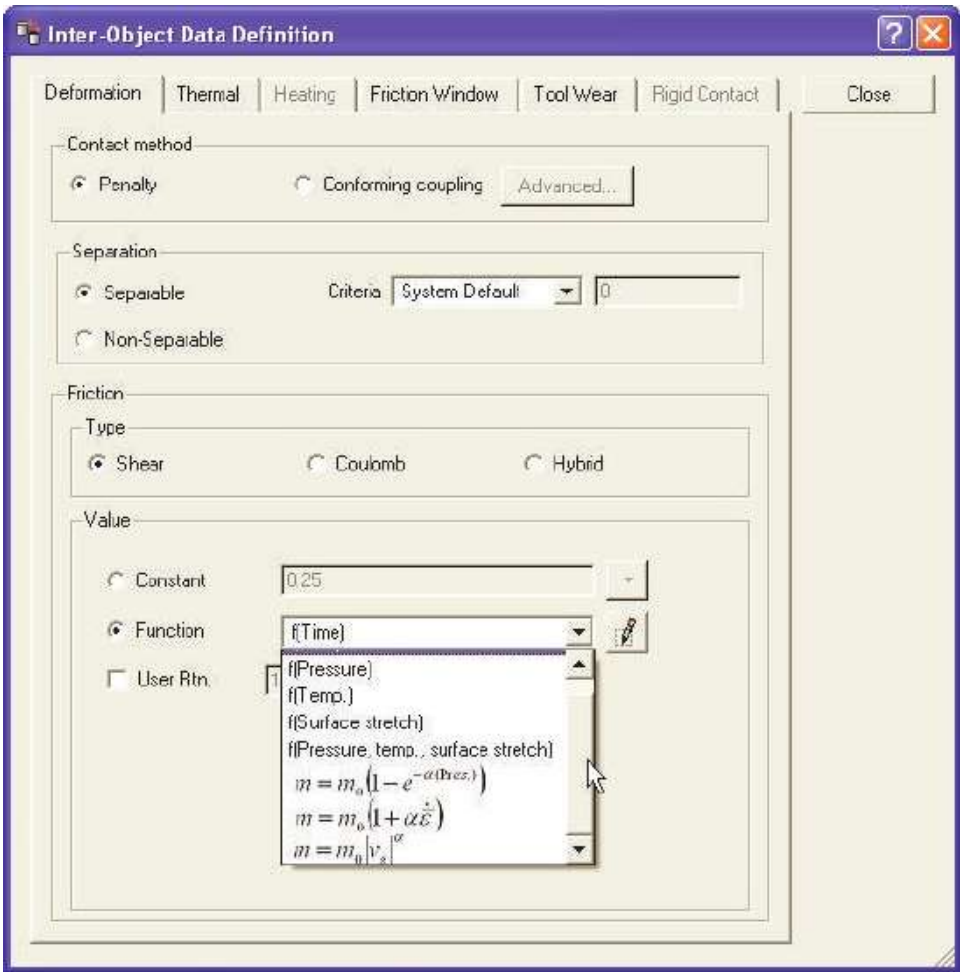


Figure 2.5.3: List of available definitions for shear friction factor

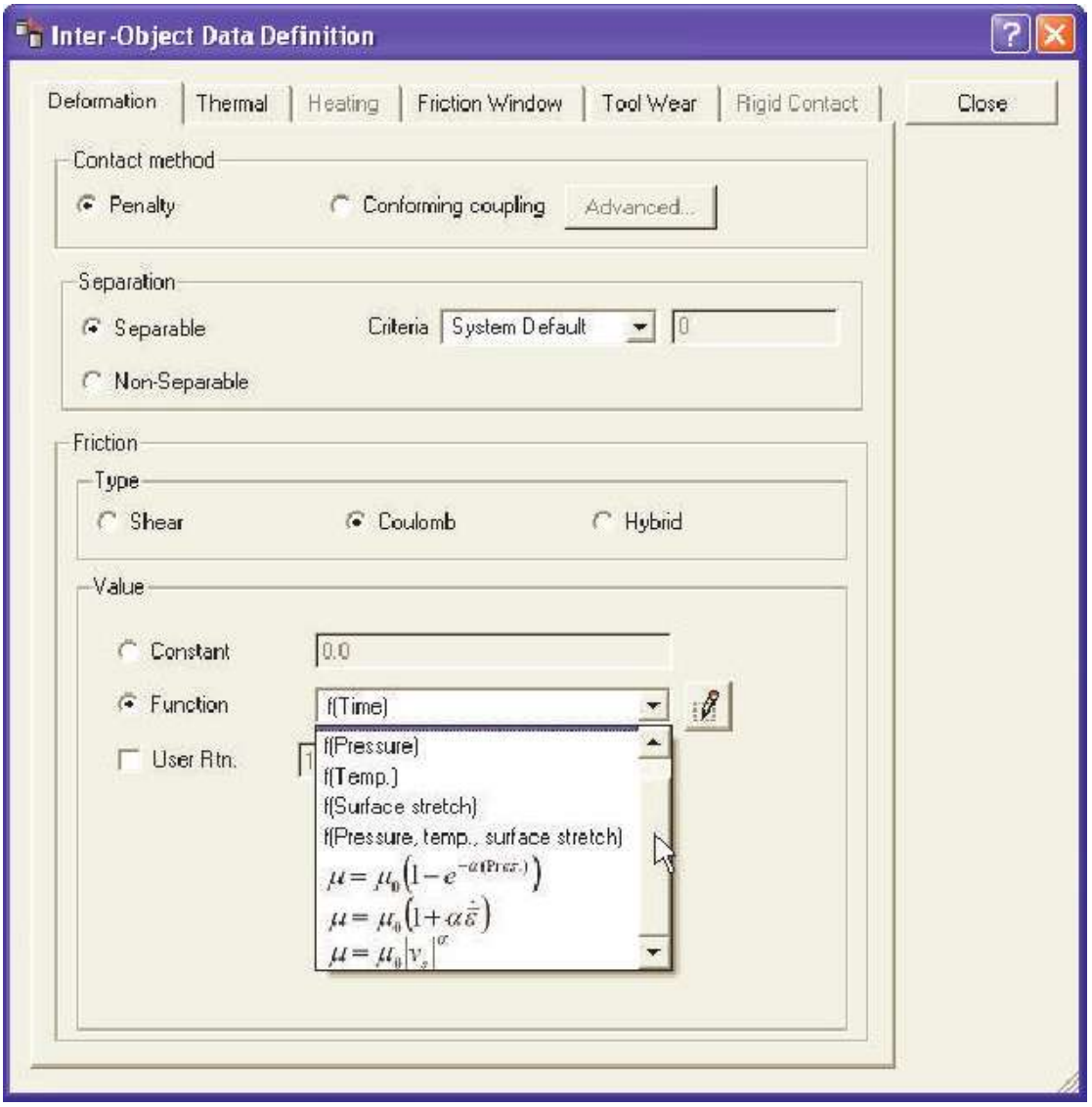


Figure 2.5.4: List of available definitions for coulomb friction coefficient.

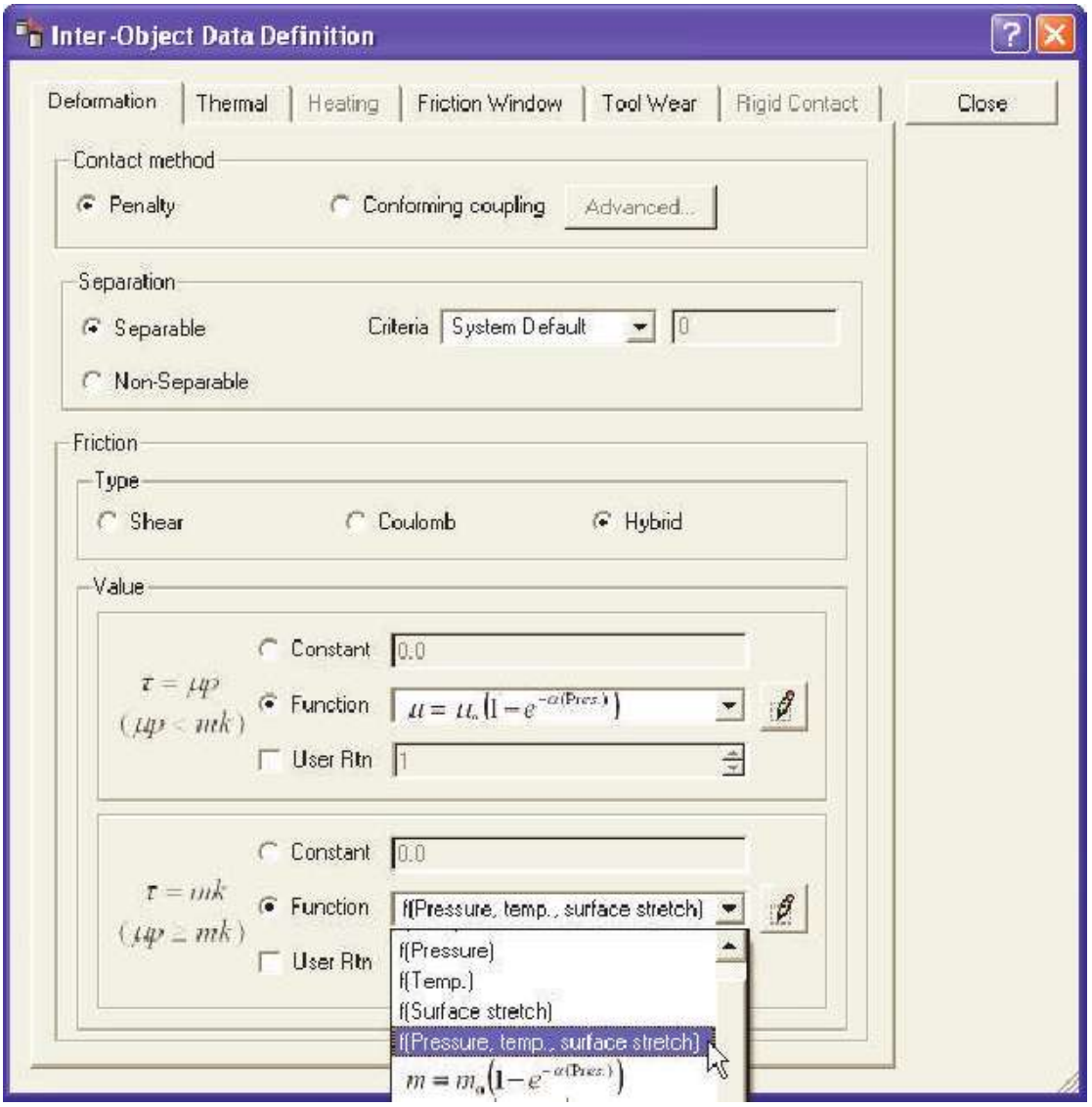


Figure 2.5.5: Hybrid definition for friction coefficient

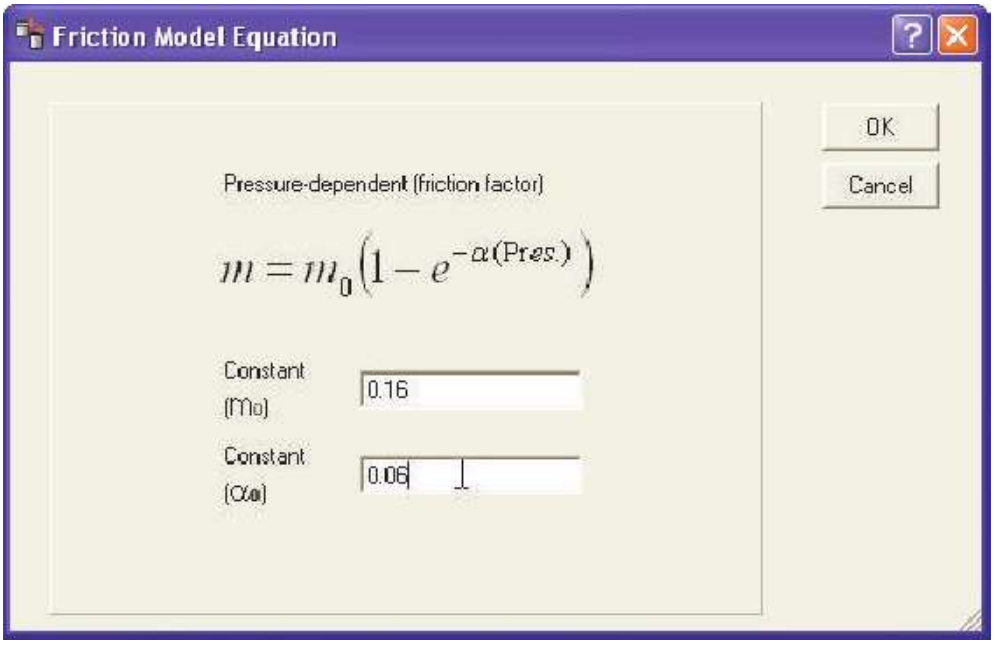


Figure 2.5.6: Pressure dependent shear friction factor

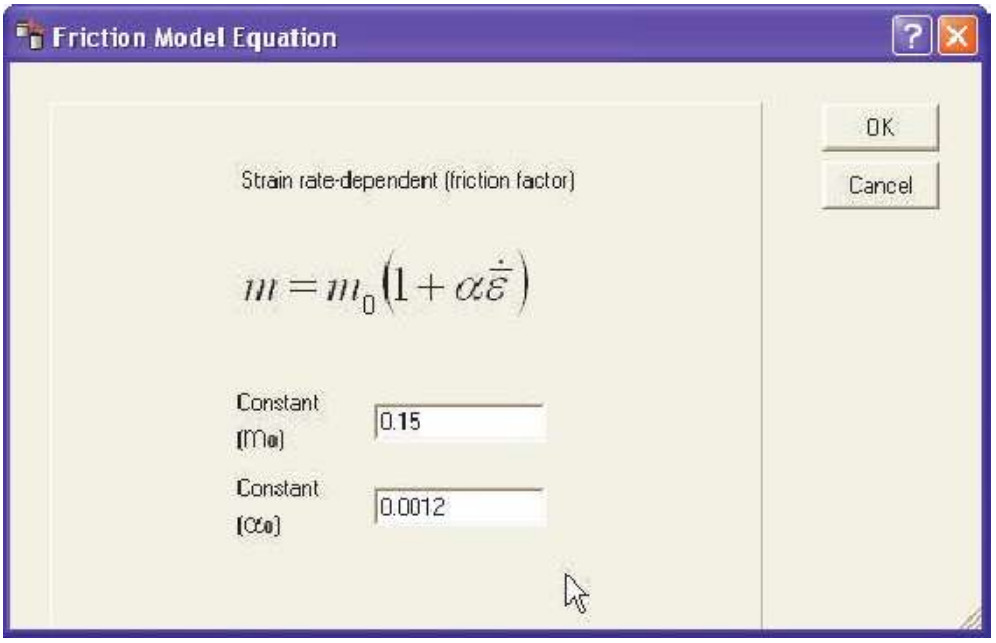


Figure 2.5.7: Strain rate dependent shear friction factor

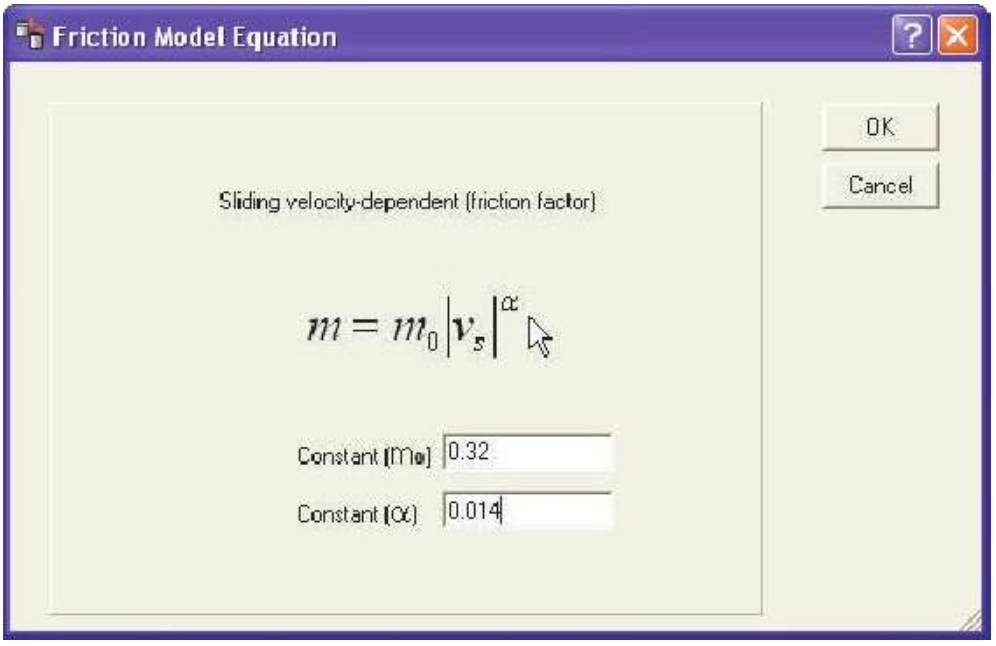


Figure 2.5.8: Sliding velocity dependent shear friction factor

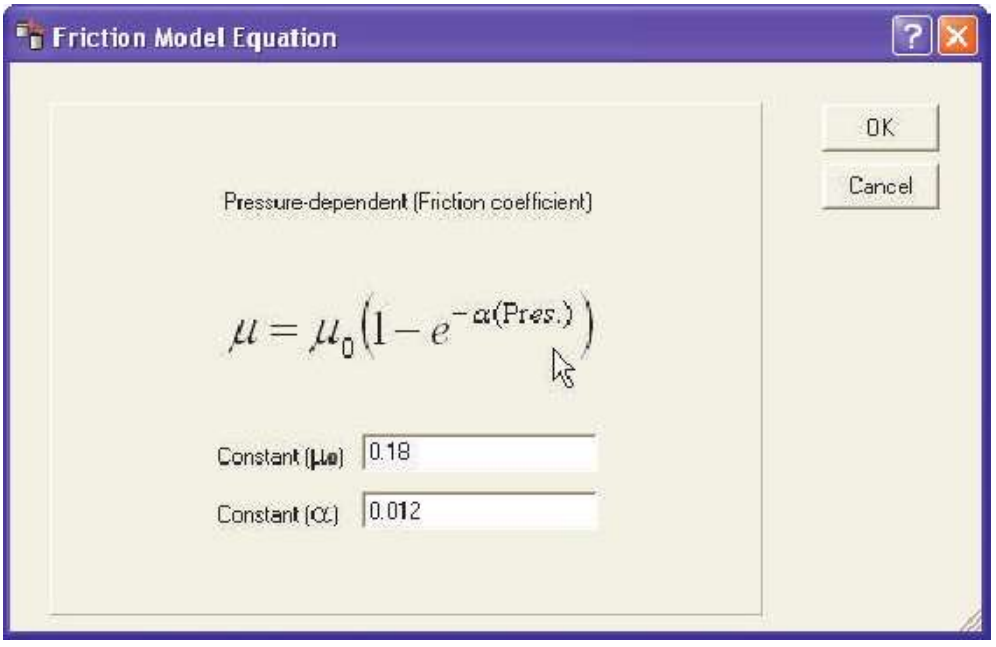


Figure 2.5.9: Pressure dependent coulomb friction coefficient

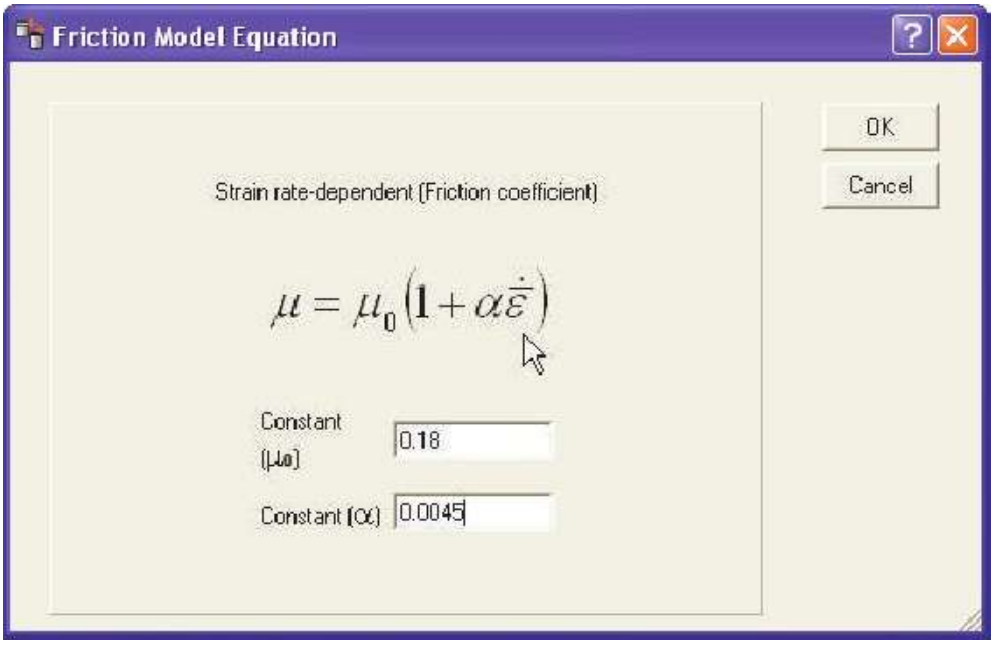


Figure 2.5.10: Strain rate dependent coulomb friction coefficient

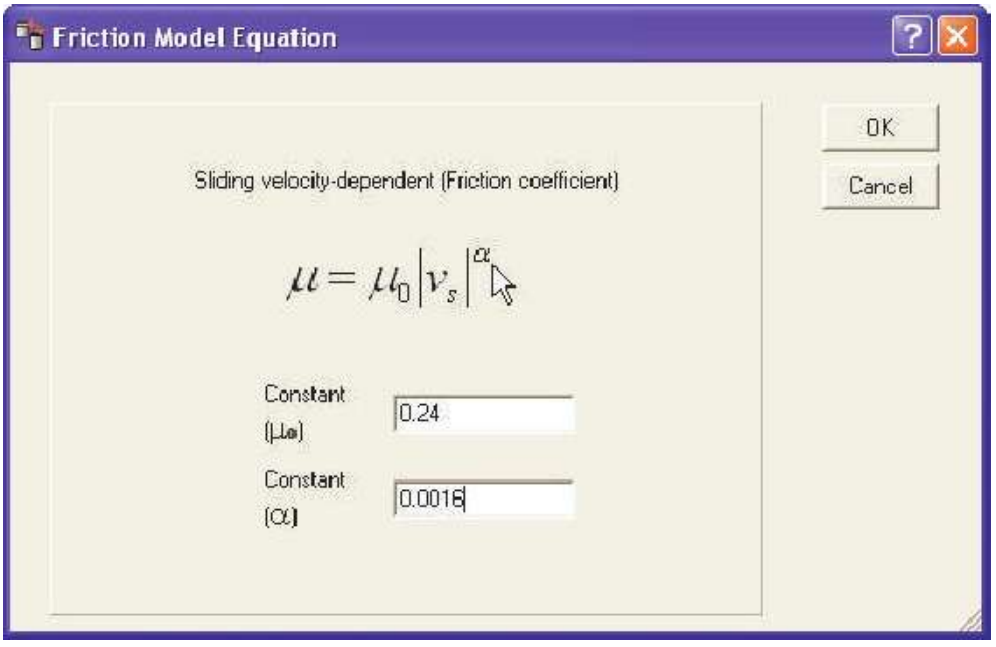


Figure 2.5.11: Sliding velocity dependent coulomb friction coefficient

Separation criteria (SEPRES)
The separation criteria defines how the nodes at the inter object interface will behave when acted upon by a tensile force. Three ways of defining the separation criteria exist. They are:

1. **Default** This setting will cause normal separation when the contacting node experiences a tensile force or pressure greater than 0.1

2. **Flow stress** This setting will cause nodal separation when the tension on the contact node is greater than a given percentage of flow stress of the slave object. This percentage must be input by the user in the Separation Criteria box.

3. **Absolute pressure** This setting will cause nodal separation when the tension experienced by the node is greater than the input pressure. This pressure must be indicated in the box marked separation criteria.

Separation Relation (SEPCRI)
The Separation Relation allows nodal contact to be defined as non-separable under any condition. This condition should generally only be used to attach nodes to a rigid symmetry plane when defining symmetry on a plane other than the XY, YZ, or ZX planes

Self Contact
Self-contact boundary condition is available in DEFORM-3D starting at version 5.1. This capability is under continuous improvement and it is recommended to use the latest release version for this capability.

Interface heat transfer coefficient (IHTCOF)
The interface heat transfer coefficient specifies the coefficient of heat transfer between two objects in contact. This can be specified as a constant or a function of time or interface pressure. (see Figure 2.5.12) The interface heat transfer coefficient is generally a complex function determined by the interface pressure, amount of sliding, and interface temperature. If this data is available, it can be entered as a table.
If no data is available, values of 0.004 (English) or 11 (SI) should give reasonable results.

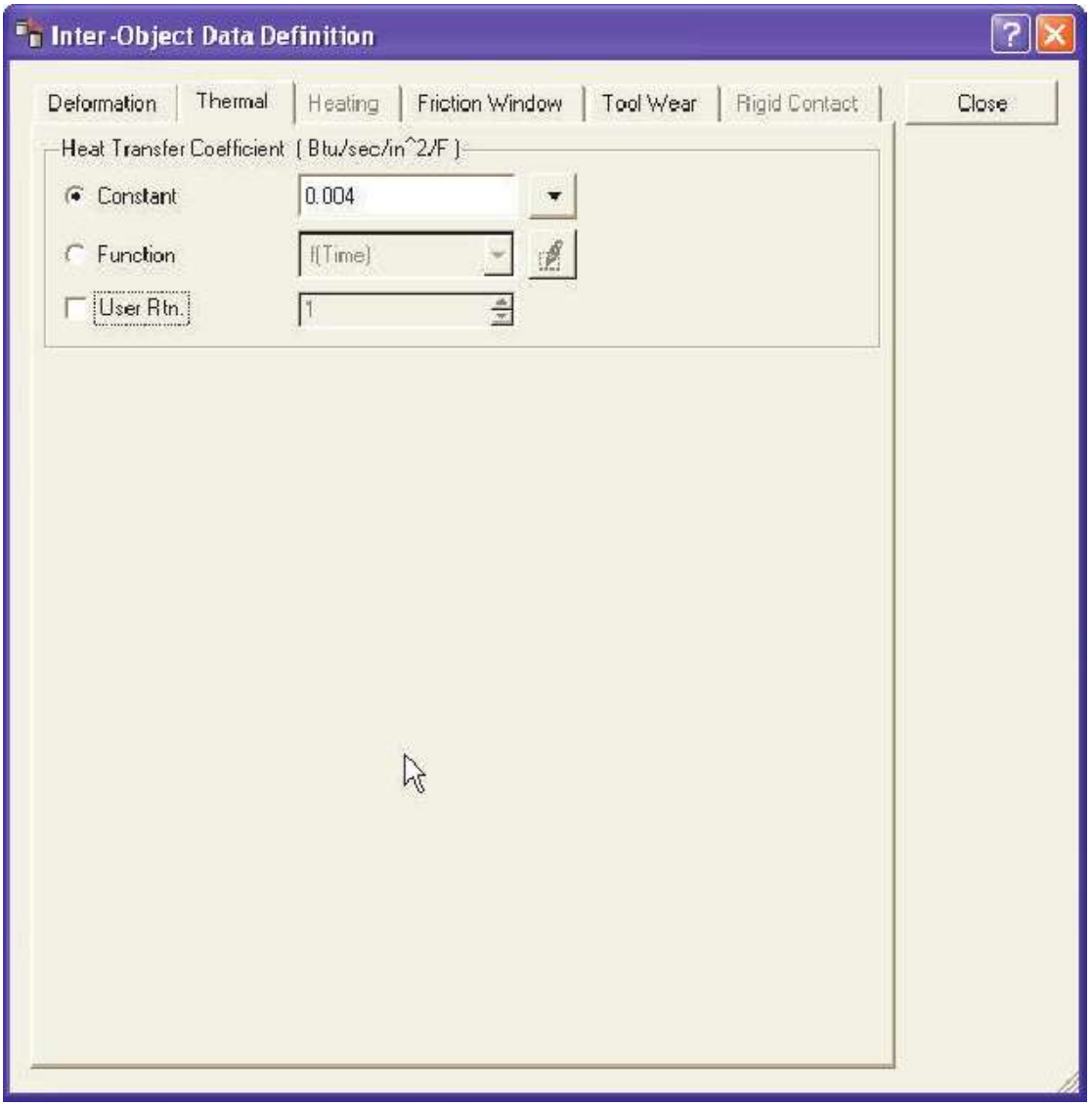


Figure 2.5.12: Inter-object window (thermal settings).

Tool Wear

There are two predefined models for predicting wear: Archard's model and Usui's model. There is also a capability for the user to define an arbitrary wear model using user subroutines. (See Figure 2.5.13).

Wear models can be defined for each pair of objects that come into contact during the process. They are defined under **Inter-Object Data**. Wear rates are computed for the master object, and that object must have a mesh (therefore heat transfer calculations must be activated under **Simulation Controls)**.

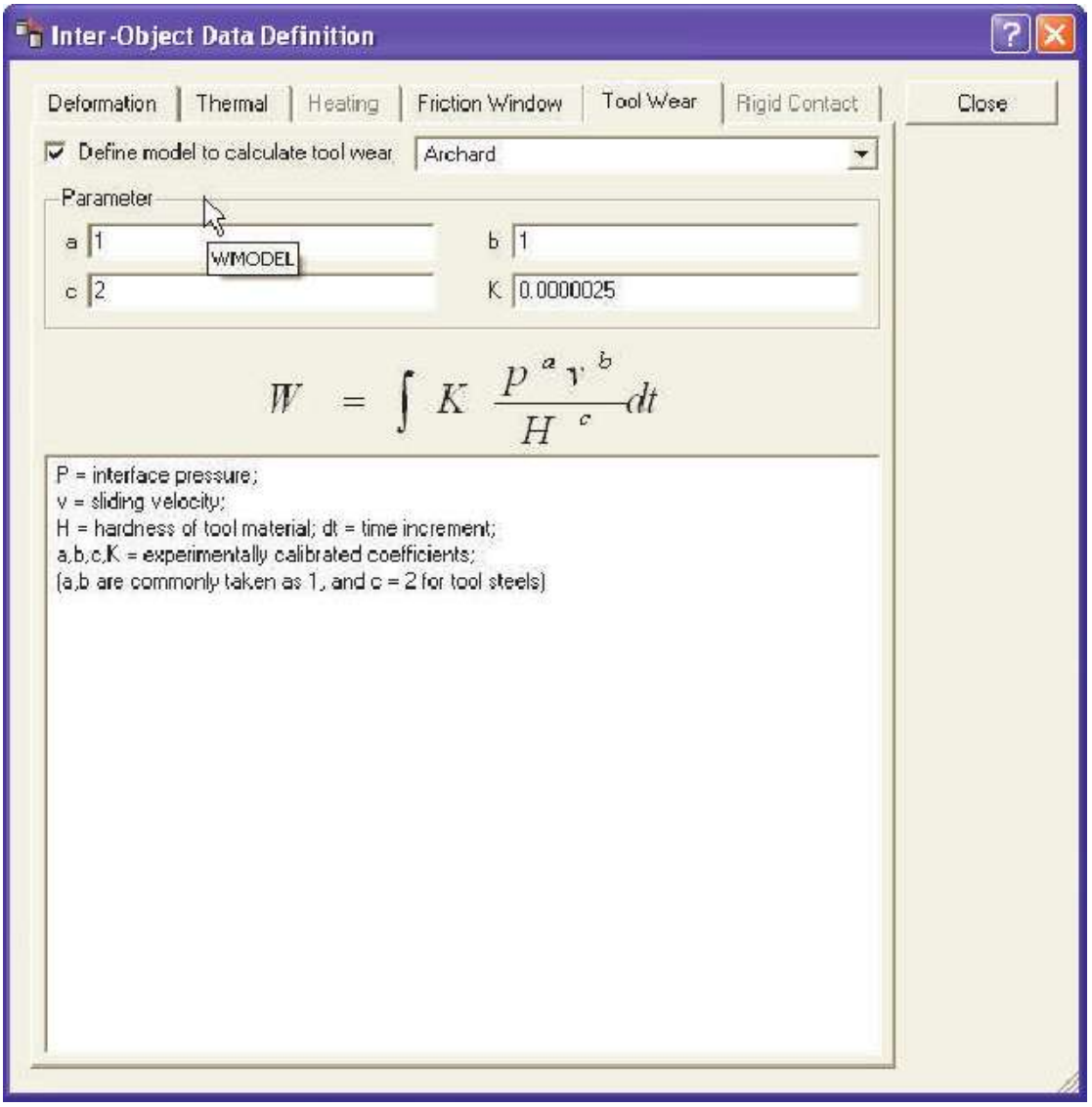


Figure 2.5.13: Inter-object window (die wear settings).

In the Post Processor, the user can evaluate the total (integrated) wear depth up to a specified step for the process as well as the incremental wear depth for the time interval of the last step. Additionally the user can obtain the sliding velocities, contact pressure and interface temperatures at the contacting die surface in the Post Processor.

Models

Archard's model is generally better suited for discrete processes such as cold or hot forging. In these cases, abrasive wear is the dominant wear mode.

Usui's model is generally better for continuous processes such as metal cutting, where diffusion is a major contributor to wear.

Typically the coefficients used for these models should come from a series of calibration experiments. In lieu of calibrated data, standard values can be used to obtain relative wear rates for similar processes. Proper techniques for the modeling of coatings and surface treatments (such as nitriding) is still a topic of very active research. Therefore, comparison of the effects of different surface treatments is difficult without additional data. Contact DEFORM support for assistance in finding the latest research in this area. Nonetheless, the following guidelines can be offered:

For Archard's model, the following coefficients will give reasonable results for common tool steels:

A = 1
B = 1
C = 2
K = 50

Hardness must be specified in the **Advanced->Element Data** section of the Pre Processor. If the K=50 value is used, then hardness values should be entered using the Rockwell C hardness scale. Please note that the value of K used in the model is essentially a scaling factor to calibrate the model using experimentally observed wear values.

For Usui's model, typical values for machining processes are

A = 1.0E-5 (or interface sliding velocity * interface pressure)$^{-1}$
B = 1000 (order of magnitude of absolute interface temperature)

It should be noted that these values are for qualitative comparison of similar processes only, and will not give quantitative estimates of actual tool life.

In order to use tool wear, the following conditions must be met:

1. Thermal calculations must be turned on.
2. The tool should be meshed.
3. The tool should have a hardness value defined under Advanced->Elements Data->Hardness.
4. The tool wear model should be turned on for the inter-object combination in question and non-zero values should be placed for the coefficients.

In addition to Archard's and Usui's models, user routine functionality has been also provided, where in user can evaluate any other model, using the basic model data like, sliding velocity, interface pressure and interface temperature.

From 3DV6.1, the tool wear computations can also be made in the post processor using user defined post processor variables. This means for a given model for which deformation data has been computed, user can evaluate

different too wear models, without having to re run the simulation. All the tool wear variables are stored for both the master and slave meshed objects

Friction Window

Starting with version 5.1, friction windows can now be applied to a simulation as seen in Figure 2.5.14. One of the purposes of this function is to allow the user to apply different friction conditions at specified regions of an object to simulate differences in the lubrication condition. Applying a window is done in the same manner as only other window function. In the case of two overlapping windows, the lower order number window takes precedence. The window defined friction values can also use all the friction model definition available in the system, as explained in the Inter-Object data definition. The friction window can also be defined to follow another object's movement or have it's own velocity defined.

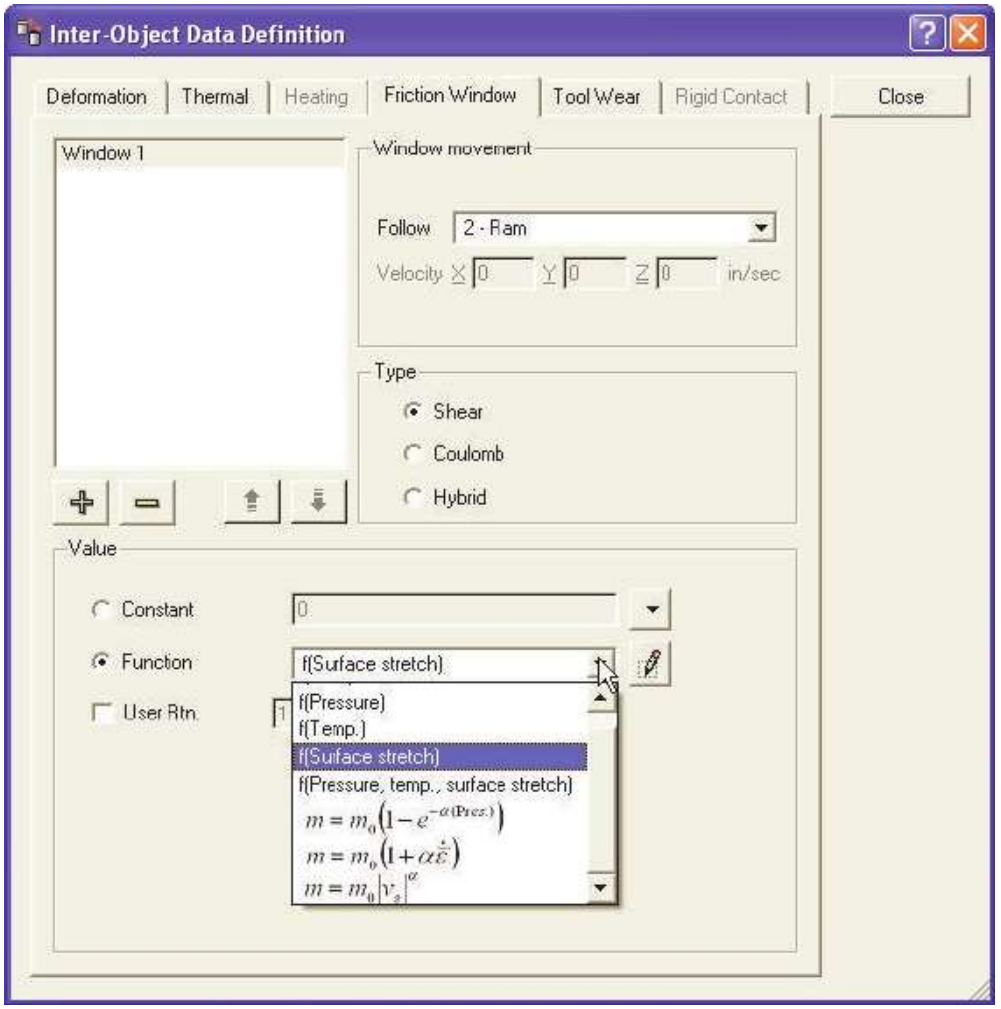
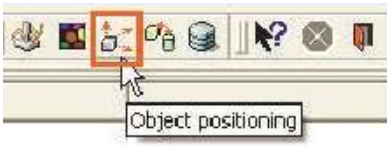


Figure 2.5.14: Friction Windows Dialog.

2.5.2. Positioning

Object positioning dialog can be accessed from the following icon located in the top row of icons.



Once the object is defined, a variety of positioning features are available to place the objects in the correct position before the process is modeled. The object can be dragged using mouse, can be dropped in to the die cavity, moved by a specific offset distance, positioned with an interference or positioned with rotational movement. (see Figure 2.5.15). A set of components can also be selected and positioned together using coupled positioning.

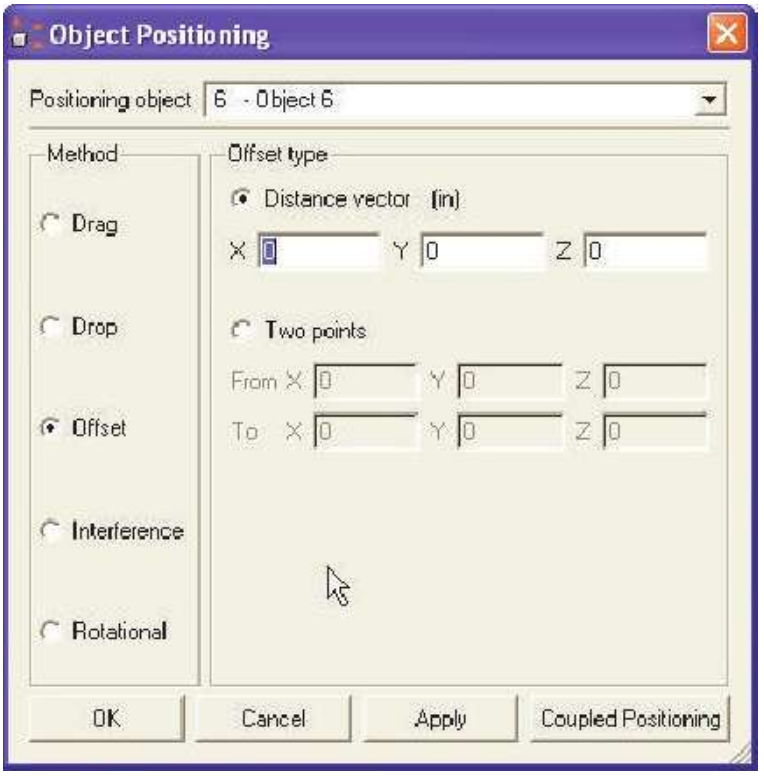


Figure 2.5.15: Positioning window.

Mouse driven positioning

Mouse driven positioning is used to move or rotate the object by allowing the user to select a vector along which to drag or spin any specified object along or about that vector.

Drop positioning
Drop positioning is used to move an object towards another object by interference or by gravity (See Figure 2.5.16).

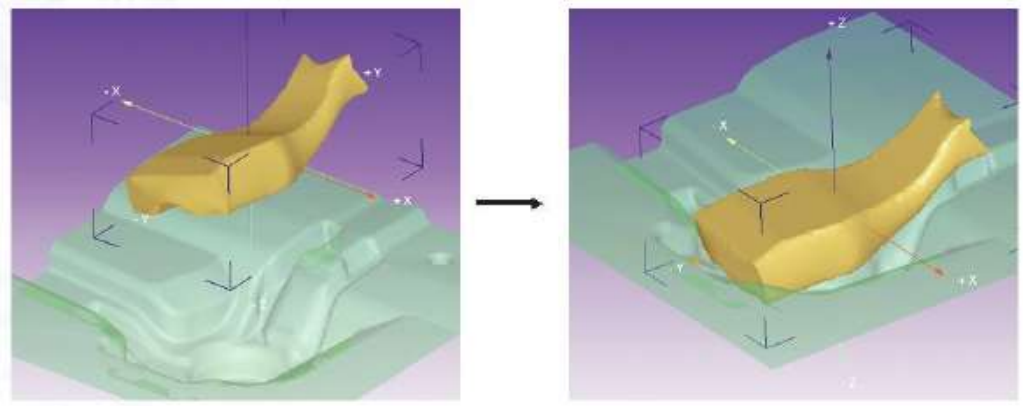


Figure 2.5.16: Drop positioning example.

Gravity activated

In the case of gravity activated, the positioned object should be the meshed object. The direction should be the direction that gravity would drive it to fall on the other object. The object can move in 6 degrees of freedom (3 directions of translation and 3 axes of rotation) without any given constraint. Constraint in rotational direction can be added if only one rotational degree of freedom is required. If only one rotational direction is required, the Allow rotation only about box should be selected and the vector about which to rotate should be specified.

Gravity not activated
In the case where gravity is not activated, this behaves exactly like the interference positioning method.

Offset positioning
Offset positioning is used to move the object to position by a given displacement in the chosen direction. The object to be positioned should be highlighted in the object list table. The displacement in the X, Y, and Z coordinates should be entered in the appropriate fields.

Interference positioning
Interference positioning moves the object to be positioned in the direction indicated until there is a slight overlapping with the reference object. The overlap is indicated by the percentage volume overlap value. During the positioning process, the object being positioned is first moved a large distance away from the reference object, then moved back towards the reference object in the in the indicated direction, until first contact is made.

Common Question: What is a reasonable value of Interference?
Answer: Interference should be adjusted such that when inter-object boundary conditions are generated, a reasonable number of nodes are in contact with tools. Contact nodes should be generated wherever the object should reasonably be touching the tools. This may require increasing or decreasing both the interference value and the contact node generation tolerance.

Rotation positioning
Rotation positioning allows the user to rotate any object about a specified axis. The axis of rotation is specified from a point and a direction vector. The specified rotation angle is positive for a right hand rotation about the axis.

Coupled positioning
This method of positioning can be used to position a set of objects at the same time. (see Figure 2.5.17)

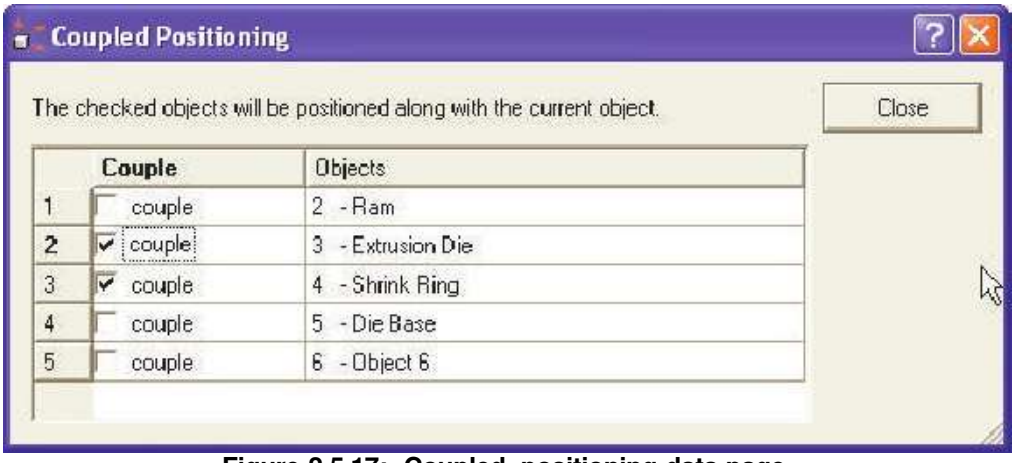


Figure 2.5.17: Coupled positioning data page.

2.5.3. Inter object boundary conditions

Inter object boundary conditions identify nodes of a slave object that are in contact with a master surface. Contact conditions are automatically updated by DEFORM, but it is necessary to define initial contact conditions to assist with the initial solution step. When inter object boundary conditions are generated, DEFORM checks a tolerance band (defined by the tolerance distance) around all master surfaces. Any nodes falling within this tolerance band are considered to be in contact with the surface, and their position is adjusted (if necessary) so they are on the surface of the master object.
The tolerance distance should generally be set to about 10% of the edge length of the smallest element in the slave object.
If a node is inside the object by a distance greater than the tolerance band, contact will not be generated on this node, and it may continue to penetrate the

object during deformation. Colored marks should appear on each contact node. If no contact is generated, it may be necessary to reposition the object using interference positioning, with either a larger or smaller interference band.

Note: If master object geometry is improperly defined with surface normals inward, the behavior of boundary contact generation will be unpredictable

2.6. Database Generation

The simulation data set entered into the preprocessor can be written as a new database, or appended to an existing database file. The information will be written as a negative step, indicating that it was written from the pre-processor and not the simulation engine. In an existing database, any steps higher than the current step will be overwritten at this time.

The simulation database generation page, click on the icon  .The simulation database will be checked as it is written.

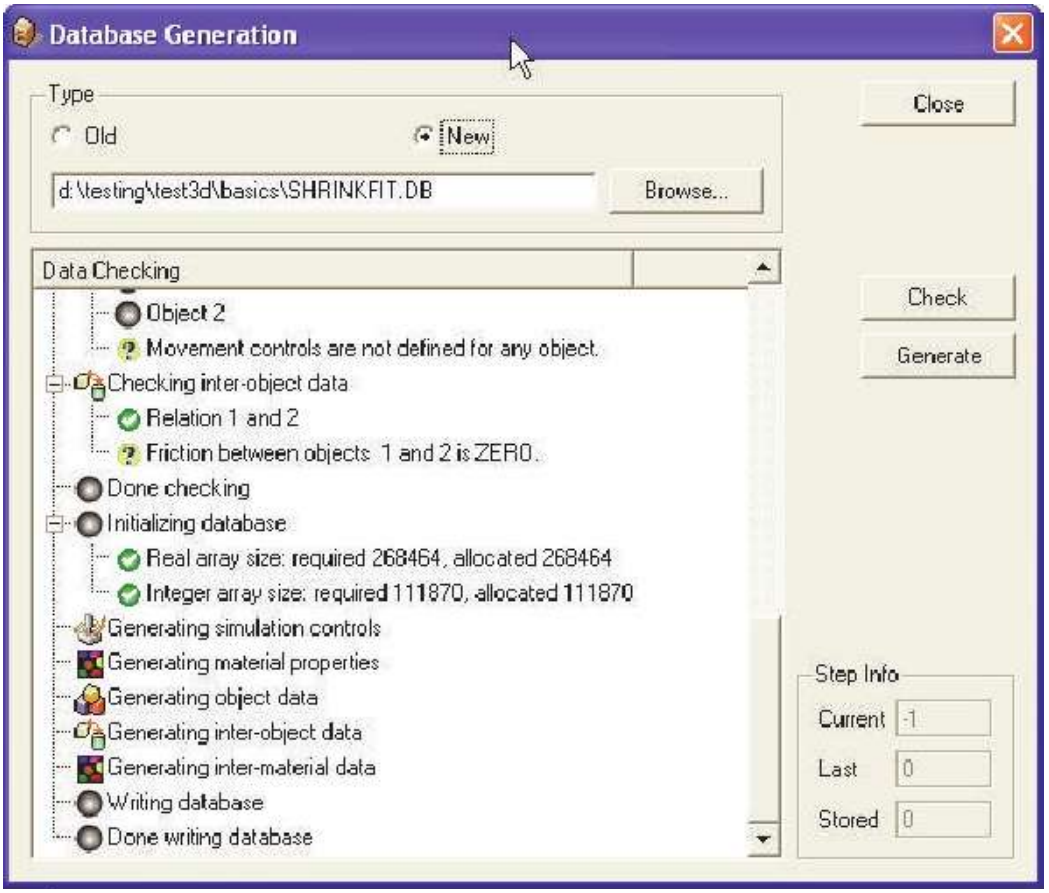


Figure 2.6.1: Database generation dialog.

Data errors
Errors are serious problems with the data set that will prevent the simulation from running. These errors are marked with red flags in data checking and must be resolved before the database can be written.

Data warnings
Warnings are conditions which may cause undesirable solution behavior but will not prevent the simulation from running. Warnings are marked with yellow flags. If warnings exist, each one should be carefully checked and the source identified. Some warnings represent unusual, but valid, data situations. If this is the case, they can be ignored and the simulation can be run.

Chapter 3. Running Simulations

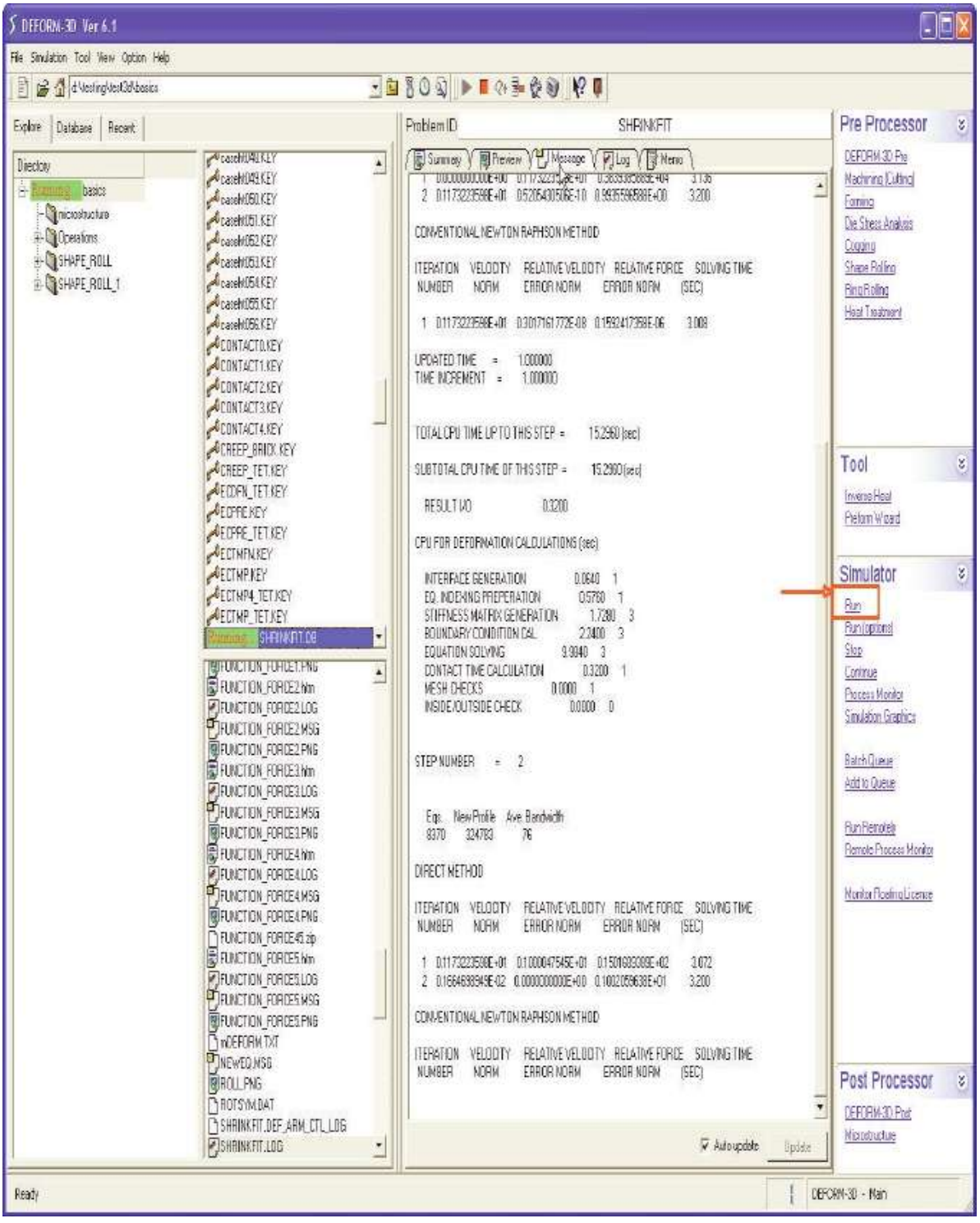


Figure 3.0: Main window of DEFORM-3D. (Running a model)

3.1. Simulation Options

By clicking the Run (Options) button, the following selections are available for running a simulation. The Multi-Processing, simulation graphics and e-mailing is discussed individually below. The last two options are:

Keep the message file – This prevents the message file from being lost after each remesh by renaming it to a unique name for each simulation run.

No automatic remeshing (for nonconvergence) – This prevents the simulation from restarting with a automatic remesh if the simulation stops due to convergence problems.

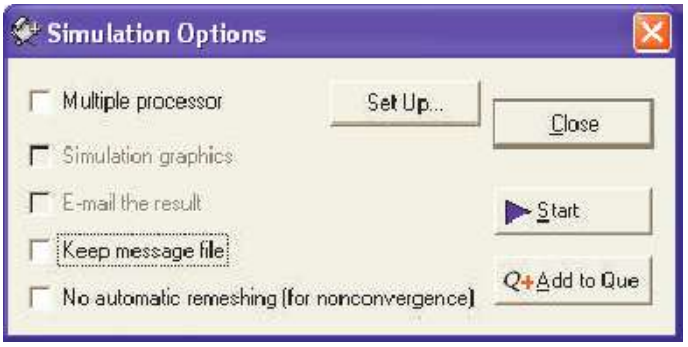


Figure 3.1.1: Simulation Options

3.2. Switching between Solvers (Conjugate-Gradient and Sparse)

The Conjugate Gradient (Deformation solver (SOLMTD)) solver generally runs faster and requires less memory than the Sparse solver (Deformation solver (SOLMTD)). However, as with iterative solvers, sometimes has difficulty in convergence for cases with little contact. It has been found, for instance, that at the beginning of a forming process when there are only very few contact nodes, the convergence difficulty occurs. To overcome the convergence difficulty, an automatic switch to the Sparse solver is implemented in the code.
But for a large size problem with tetrahedral elements, the Sparse solver requires more memory that the computer may be able to provide. The process will stop without a warning message. The program now tentatively sets 140,000 as the limit for the Sparse switch. Of course, this value should depend on the available RAM of the computer and should not be a constant. To specify the user's own value, he/she can set a limit in terms of the total tetrahedral element number in a file called SW2SP.DAT. This data file should be put in the current working directory.
To find out whether your computer can handle a problem with a potential switch to the Sparse solver, one can choose the Sparse solver in the database and run a few DEFORM iterations. If there is a message where the machine is unable to allocate sufficient memory to run the problem, then it is likely that this approach is necessary.

3.3. Multi Processing

This radio box activates whether the Multiple Processing functionality is activated or not. By clicking on the button next to the Multiple Processors Selection, the user will see the Multiple Processor Set Up window. This allows the user to specify the computers used to solve the problem as well as the number of processors used on each computer. It is very important user enters the correct network name of the computer for the simulation to run. Partially parallel FEM option runs only the solver part of the simulation in parallel mode, while the other operations like model stiffness matrix, remeshing and interpolation are run in single cpu on the primary host machine. In PC environment this results in the execution of one DEF_SIM_P4.EXE on each of the processors requested. Fully parallel FEM handles the full model in be run in parallel, including the model stiffness matrix, remeshing and interpolation apart from the solution phase. In PC environment fully parallel run results in the execution of one DEF_SIM_P4P.EXE on each of the processors requested. With respect to the total simulation time, larger model size typically gains from this multiprocessing setup.

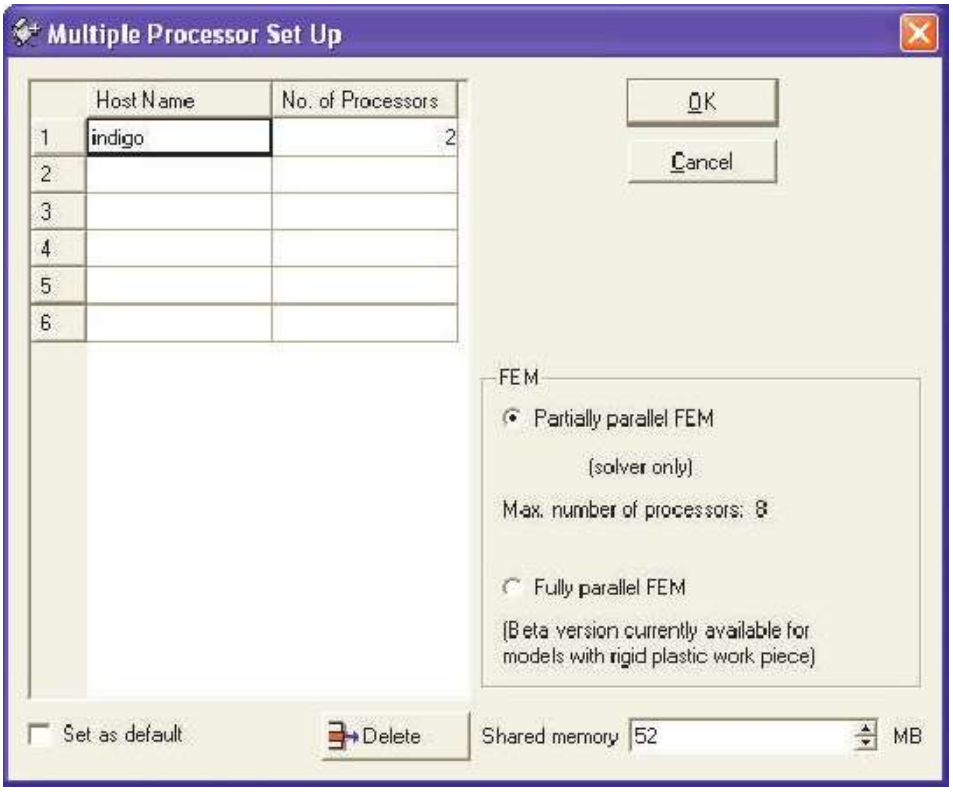


Figure 3.3.1: Multiple Processor Set Up window.

3.4. Email the Result

This function allows DEFORM to send an e-mail notification of the completion of a simulation. If Yes is selected, the e-mail address to send to is required as well as the name of the company SMTP mail server. Enter the environment settings dialog as indicated in Figure 3.4.1 and specify the main settings as shown in Figure 3.4.2 .



Figure 3.4.1: Environment settings

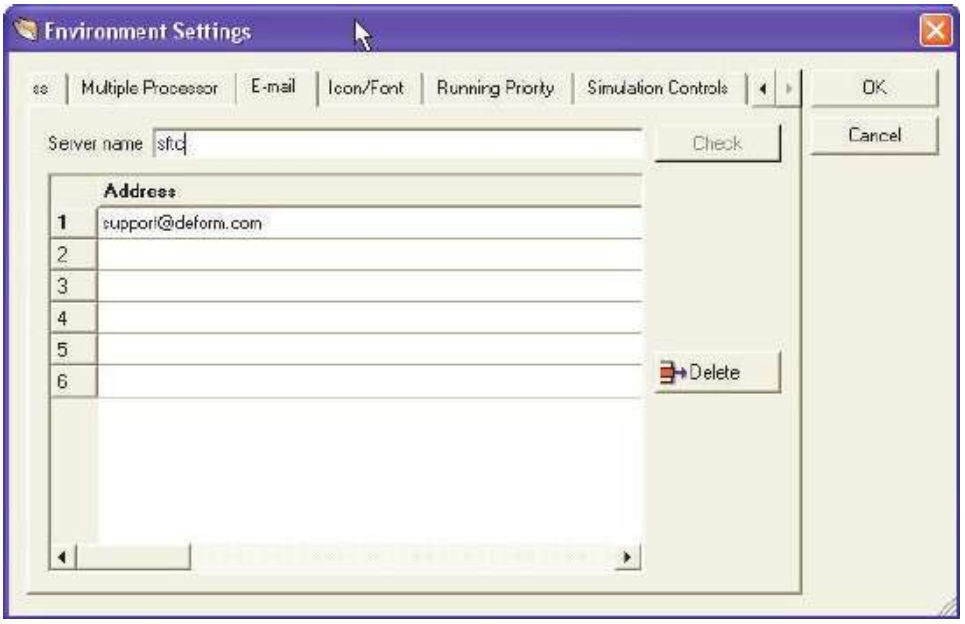


Figure 3.4.2: Settings for email notification on simulation status

3.5. Starting the simulation

The simulation is started by clicking the *Simulation, Start* button. This initiates a series of operations to run the simulation and generate new meshes as necessary.
Run-time information will be written to the ProblemId.MSG and ProblemId.LOG files.

- Execution information, including convergence information for each step and simulation error messages, can be found in the .MSG file.
- Information on simulation and remeshing, execution times, and fatal errors can be found in the .LOG file or in the command window where DEFORM was executed from.
- Memo option allows user to enter and save any notes related to the current simulation.

3.6. Simulation graphics

This function is now available in DEFORM-3D as seen in Figure 3.6.1. While the simulation is running, the second most recent saved step can be viewed. Many different variables can be viewed such as plastic strain, plastic strain rate, temperature and many more.

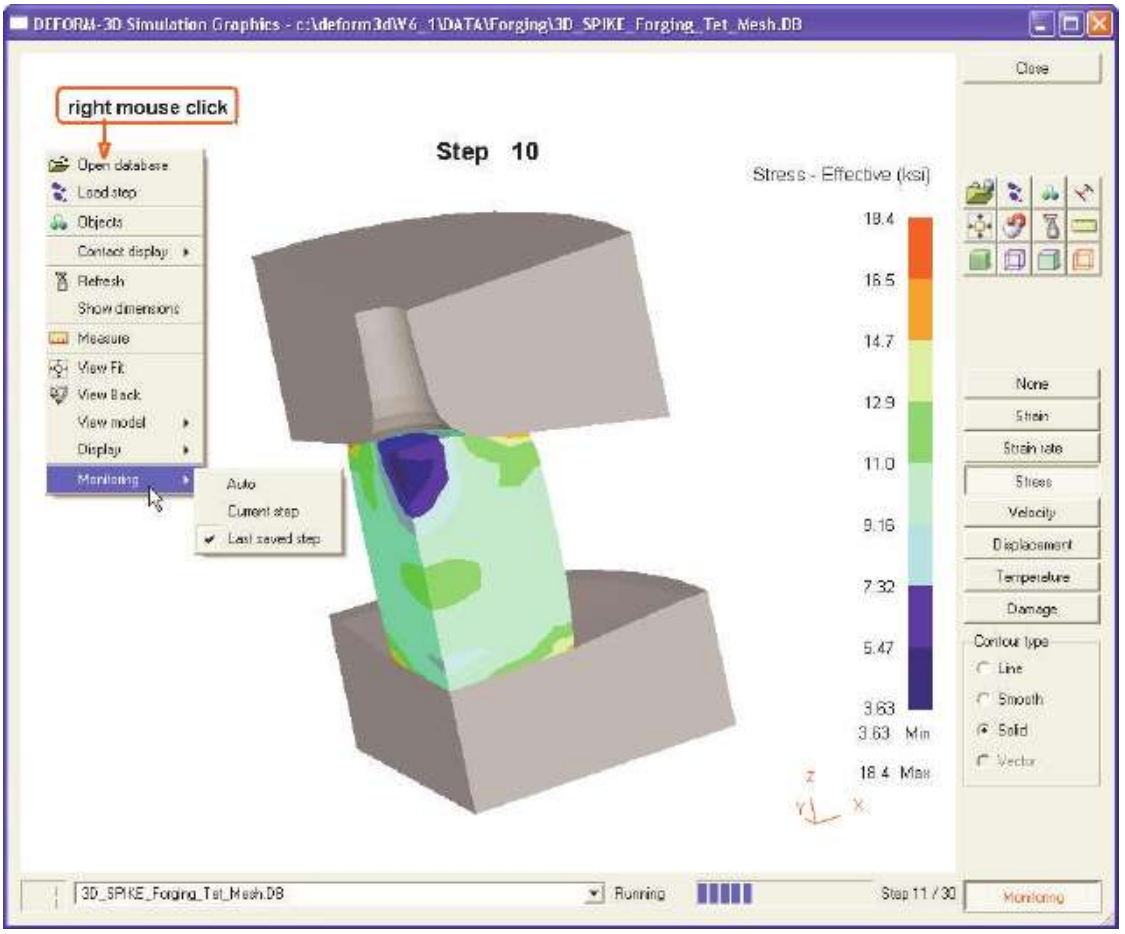


Figure 3.6.1: The simulation graphics window.

There are also three additional ways in which to monitor a simulation.
1. Use the Process Monitor to determine the current step number.
2. Read the message file to determine the current step number and iteration information.
3. Open the simulation in the Post-Processor. During a simulation the database file is renamed to FOR003 and is renamed back to the original database file name upon remeshing and stopping.
4. Depending up on the frequency of the steps saved in the database, it is recommended to opt for saved steps display rather than the current step.

3.7. Add to Queue (Batch Queue)

This will allow the simulation specified in the simulation window to be added to the DEFORM queuing system which will run multiple jobs in a consecutive manner. Once the simulations are added to the batch, they are listed consecutively in the Batch Queue window (see below). The Batch Queue window, accessed from the main DEFORM window, orders the jobs in the queue and lists the problem names and the paths.
From the Batch Queue window, the queue can be submitted and modified. Once a problem is finished, it is removed from the Batch Queue list and the next job in the batch can be submitted by pressing the Submit Queue button.

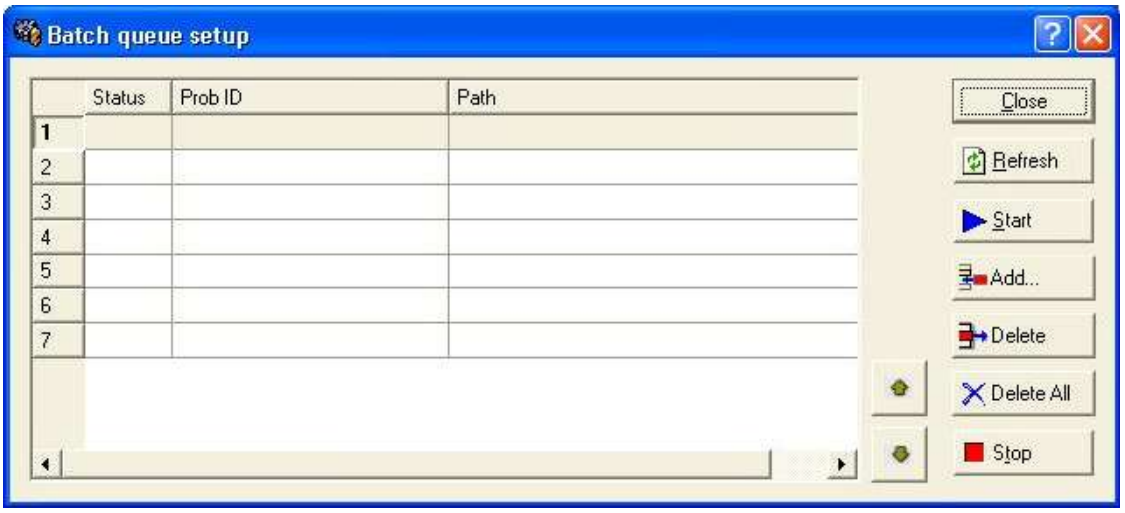


Figure 3.7.1: Batch queue window.

3.8. Process Monitor

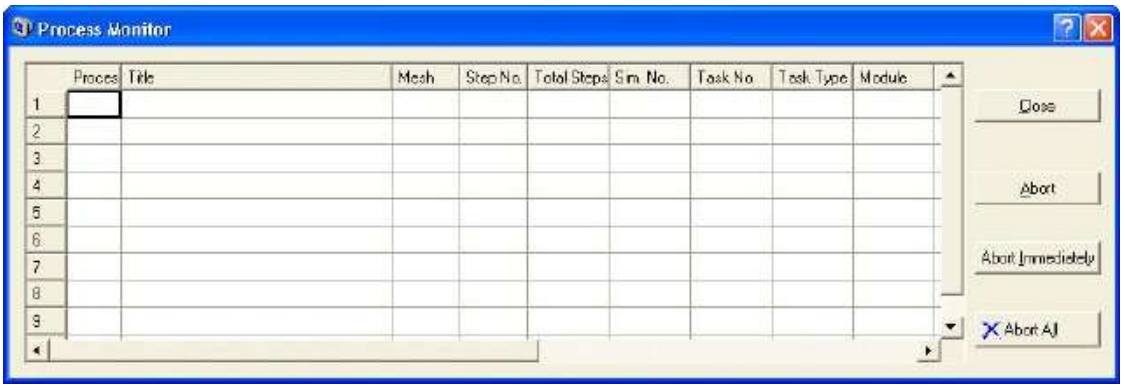


Figure 3.8.1: Process monitor.

The process monitor displays the status of any simulations running on the CPU. The process will be displayed while remeshing is occurring as a different module (either DEF_AMG or other such processes). The Process Monitor reads the DEFORM status file to determine what simulations are running and what simulations are stopped. If this file gets corrupted in any way, the process monitor may have problems. In order to remedy this problem, the user needs to recreate the status file. This can be done by running the INSTALL2D script for 2D and the INSTALL3D script for 3D (Unix only). In the case of Windows, this can be regenerated by running the program GENARM.EXE in the installation directory of DEFORM-3D.

Abort
This will stop a simulation at the next converged step. This may take a few minutes to stop the simulation.

Abort Immediately
This will stop a simulation instantly. Any calculations at the current step will be lost.

Abort All
This will stop all simulations listed at the next converged step. This may take a few minutes to stop the simulations

3.9. Stopping a simulation

A simulation may be stopped using the "Abort" or "Abort immediately" buttons in the process monitor. The simulation will be stopped after the current step is completed.

Killing a simulation process
Since the Abort/Stop command will only stop a simulation after the current step is completed, substantial time may be required for the simulation to stop. To kill a simulation immediately, the process DEF_SIM.EXE must be killed directly from the operating system.

3.10. Troubleshooting problems

The message file is an iteration by iteration account of the simulation. It shows how close each iteration is to converging to the final solution by calculating a relative error norm between successive steps for both nodal velocities and nodal forces. There is also information given on node contact being generated and also if there is a sudden stop. The log file is an account of the module execution

3.10.1. Message file messages

If a simulation stops early, it is recommended to check the end of the message file to see why the simulation stopped. While a simulation is run under the batch mode of operation, there are three places in which the end of operation may be indicated. The message file is where the simulation will indicate a normal stoppage of the FEM module. If there is an abnormal stoppage in simulation, due to an illegal operation in the FEM module or in the mesh generation, this will be indicated in either the .LOG file or in the command window where DEFORM was initially called. This may be also accompanied by a core file which should be deleted as soon as possible since they often consume a large amount of disk space.

3.10.2. Simulation aborted by user

When DEFORM is installed, the permission for the status file (which monitors simulations running on the machine) should be set using the command 'chmod 777 DEFORM.STA'. If this is not done, the FEM engine cannot write to the status file and hence exits. To restart the simulation, first go to the DEFORM3_DIR, run INSTALL3D and select the option to regenerate the status file.

Note: If the user gets the message Command not found, the user should run the INSTALL3D by typing. /INSTALL3D. This will force the operating system to override the paths and force the system to run only the file in the current directory.

3.10.3. Cannot remesh at a negative step

The FEM engine prints this message and exits when a remeshing has just taken place and in the current step a negative Jacobian is encountered so remeshing has to take place again. This simulation normally arises when remeshing has failed to produce a valid mesh. Stopping the simulation prevents the DEFORM system from encountering a never ending loop.
The reasons for this could be that the time step (DSMAX, DTMAX) may be large as might happen in extrusions where the ram speed might be low but the velocity of the extrudate might be very high causing severe mesh distortion near the die radius. Another example would be in a forging when material starts to flash and the elements near the flash region get severely distorted. To avoid this problem, the time step can be reduced, the number of mesh elements can be increased or the mesh density in the area where distortion occurs can be increased. Plotting the node velocities in the Preprocessor can be helpful. It is not recommended for the displacement of the nodes each time step be larger than the edge length of the elements. Polygon edge length sub stepping can be useful in this case.

3.10.4. Remeshing is highly recommended

Remeshing is Highly Recommended because of Unbalanced Mismatch between Master and Slave is a message that occurs when the size of the elements on the slave and master surface are not proper. When specifying master-slave relationships between objects the following rules should be adhered to:

- the slave is the softer material
- the slave should have a finer/denser mesh than the master

This mesh is very important in the region where the slave and master contact each other. If the mesh on the master is much denser than the slave's mesh in a region where they contact the results of the simulation will not be accurate. If the ratio of the number of elements on the contact surface between the master and slave is greater than 3:1 then the message 'Remeshing is highly recommended because of unbalanced mismatch between master and slave' is printed as a warning in the message file. This will reduce the accuracy of the solution and steps should be taken by the user to change the mesh densities on the objects.

3.10.5Negative Jacobian

DEFORM uses the finite element method to solve problems of large deformation and AMG (Automatic Mesh Generator) to automatically provide an optimized remeshing capability. In most cases, the meshing and remeshing runs very smoothly. There are cases though, where a simulation will not continue due to a negative Jacobian (unacceptable mesh) at a step immediately after a remeshing operation. In most cases, these can be easily identified from one of the following areas:

- A closed forging lap will usually lead to this problem. Look at your progressions carefully and see is there is a small area where a lap has

formed and closed. If such a lap is detected and you would like to continue the simulation, you may try to

Using the Surface Patch feature in the Preprocessor under *Objects* → *Geometry* can be useful. Folds will appear as short red lines that do not appear to highlight a single element. Be sure to view the object from multiple angles before deciding that a red mark is a fold, because surface patches from the far side of the object can be misleading.

Modify the geometry to remove the lap, and continue the simulation. This will require remeshing and interpolation to maintain the strain, temperature and damage history

- If, for any reason, a node has penetrated one of the dies, the remeshing will become problematic. This can be evaluated by looking at the FEM MESH of all objects at the step just prior to remeshing. A node that is inside the die surface will move back to the die surface after remeshing and interpolation of boundary conditions, but an element that had an acceptable geometry can be highly distorted during this process.
- If the user sets a problem with very large time steps, and little to no sub stepping criteria, this can lead to a problem resulting from a mesh distorting severely during the first step. Refer to time step definition criteria in the preprocessor section of this manual.
- If substepping is disabled in a 3D simulation, great care must be taken to use sufficiently small time steps. This means that any time a node displacement calculates a node position inside a rigid object; the node is pushed back to the die surface. This is not a bad assumption if the node displacements are small; however, if the node displacements are large this assumption may lose its validity.

These suggestions are intended as general guidelines and may not solve all of this class of problem. If all of these areas have been investigated and subsequent attempts to run your problem fail, we would recommend sending us a keyword file for further investigation.

3.10.6. Solution does not converge

There are several common reasons for a solution not converging.

1. The material has a large rigid body motion. Much of the deforming body has a very low strain rate or is rigid.

2. The material is not strain rate sensitive.

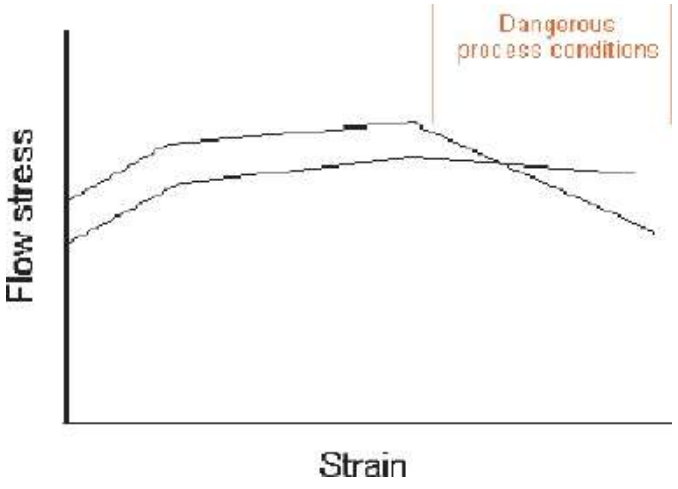
3. Elasto-plastic material is undergoing large deformation or has an inappropriate initial guess.

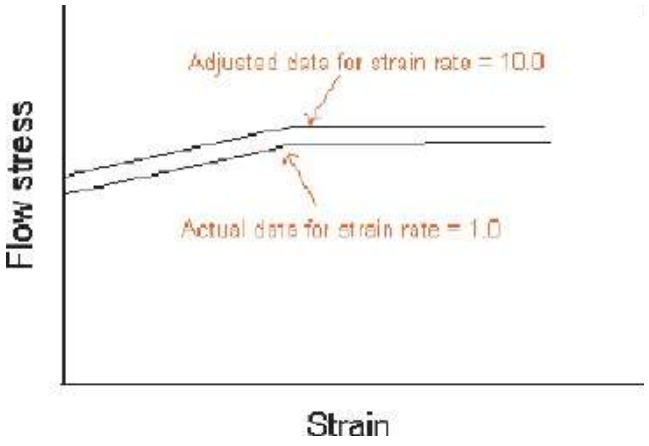
In cases where a problem will not converge, the following checklist should help with the troubleshooting process. This will help with the most common cases.

- Increase your Force Norm or Velocity Norm up to one order of magnitude. The Force Norm may in fact be raised as high as .1 or even eliminated for a few steps. This should not lead to significant error, but could result in reduced accuracy of load calculations. If convergence is improved, allow the simulation to run for 3 or 4 steps, then try reducing the settings to their original values.
- If the simulation is being run with principal die movement under load or energy control, run a couple of steps under speed control to allow the solution to stabilize before continuing under the original mode.
- Increase your limiting strain rate to 1/50 or 1/100 of the average strain rate. This should not cause any significant effect on solution accuracy. If you have an extremely difficult case to converge, this value may be lowered to 1/10 of the average strain rate for a few steps, then reset to a more normal value. Over the years, we have recommended that the limiting strain rate should be 1/100 to 1/1000 of the average strain rate. If this value is set too low, it will result in an artificially lower load calculation.
- Check your material data versus your process conditions to insure that no "strange" material properties are being passed to the FEM engine. Be particularly aware of extrapolation issues. For example, if your process conditions are in an area that is outside of the defined flow stress region, this "reverse strain rate sensitivity" create a problem that is almost impossible to allow DEFORM to converge on an accurate solution. This may be handled by re-evaluating your raw data and adjusting it as required. Since it is highly unlikely that a material has a lower flow stress at a higher strain rate, the common cause for this type of data is the lack of an adiabatic heating correction. In other words, adiabatic heating at the higher strain rates artificially heated and softened the material causing an apparently lower flow stress. If no clear cause can be determined, find data that does not exhibit this reverse strain rate sensitivity.
- Lower your penalty constant of plastic objects to 250,000 to 500,000 using a constant value (PENVOL). This may lead to volume loss if the value is much lower than 100,000 for typical engineering materials.
- Reduce your time step. This advice applies particularly for elastic-plastic materials. A very small time step can frequently allow the DEFORM system to get through a tough region of convergence. After a large number of nodes are in contact with dies and the simulation is in progress, a larger time step can be resumed. This may be accomplished through either controlling the time step or a control modifier that will lead to sub stepping such as DEMAX.
- Change the initial guess calculation method. Refer to *Object Properties* for a discussion of EP Initial Guess.
- In the case of cold materials that exhibit little to no strain rate sensitivity, this is one of the hardest cases to gain convergence. In fact there should

be a very slight strain rate sensitivity even in the cold forging world. A user may help with convergence by creating an artificial strain rate sensitivity. This is not far from reality and may be done by adding a data set of flow stress at a higher strain rate with slightly higher flow stress data. See the below figure for an idea of how to handle this type of issue.

- In a few cases, convergence problems can be caused by a course mesh in an area with high local deformation, such as under the corner of a punch during a piercing operation. In these cases, generate a finer mesh and set the remeshing criteria to have a higher bias towards boundary curvature and strain rate.





These suggestions are intended as general guidelines and may not solve all non-convergence problems. Although DEFORM has excellent convergence behavior for most problems, the occasional problem will occur where the user experiences some difficulty. If all of these attempts fail, we would recommend sending us a keyword file for further investigation.

3.10.7. Stiffness matrix is non-positive definite

DEFORM uses the finite element method to solve problems of plastic deformation, heat transfer and elastic deflection. When using this method, a material stiffness matrix is defined by geometry and material properties. When the user sees a message "non positive definite stiffness" or "zero pivot", this is caused by a stiffness matrix that has a zero value. This problem is usually caused by a material property having a zero value that is related to the "stiffness" or resistance to flow, heat or deformation.
For a given type of simulation, there is a property that is most likely to lead to this problem. It is as follows:

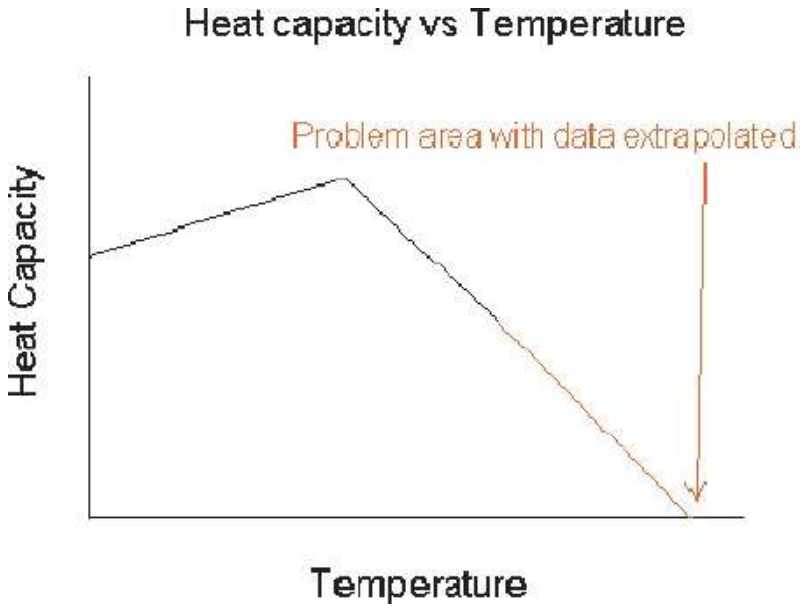
- Heat Transfer: Heat Capacity and/or Thermal Conductivity
- Elastic: Young's Modulus
- Plastic Deformation: Flow Stress
- Heat transfer problems can be identified by the information in the message file. The iteration information will contain the heading Temperature Error Norm in the section just prior to the problem being terminated.
- Plastic deformation problems can be identified clearly by the information in the message file. The iteration information will contain the headings Force Error Norm and Velocity Error Norm in the section just prior to the problem being terminated. These messages will also be seen during the elastic (die stress) type of problem.

3.10.8. Zero pivot

Rigid body motion can lead to the "zero pivot" error. This leads to the velocity norm increasing and a resulting simulation failure. This typically results from the lack of adequate boundary conditions and can be resolved by properly defining these. For a 2-D case, there are two possible geometric modes: Axisymmetric and Plane Strain simulations. In the case of an axisymmetric simulation, only the y-direction needs constraint. In the case of a plane strain simulation, both the x and y directions need to be constrained for the meshed objects. For the case of rigid objects, these do not need any constraint since they act more as boundary conditions.

3.10.9. Extrapolation of data

Many of these problems are not the result of "bad data" or the user neglecting to input the data properly. The problem has to do with a data set that does not cover the process being modeled. This being the case, the available data is extrapolated to the process window. In this case, the material needs to be updated to include data in the actual process window or adjusted (best estimate) to insure that values at or below zero can not occur when these critical properties are included in the stiffness equations. See the below figure for a further description.



These suggestions are intended as general guidelines and may not solve all of this class of problem. If all of these areas have been investigated and subsequent attempts to run your problem fail, we would recommend sending us a keyword file for further investigation.

3.10.10. Bad Element Shape

This is a error created in the FEM engine which warns of dangerous distortion in the mesh of a 3D simulation. The two main causes for this error is a negative jacobian shape that cannot be resolved or a lack of flow stress values. In the case of the negative jacobian, the problematic elements (along with the node coordinates for the problem elements) will be listed in the file BUG.MSG. If every element in the mesh is listed, the flow stress for the object was probably not defined. If there are only a few elements listed, the user may try to load the bad step into the Pre-Processor and check the shape of the elements and the Boundary Condition code. The importance of well-defined boundary conditions cannot be overemphasized. This includes meaningful contact conditions and fixed velocity conditions.

3.10.11. Inconsistent Step Number

This implies that a while the simulation was started/restarted, the most recent step in the database is not a negative step. In the case of DEFORM, a negative value on a step is simply a flag to indicate that a start/restart is possible. The negative value says nothing algebraic as it is merely a flag. The cause of this error can be any of the following:

- An already run simulation was submitted and needs to have a negative step added to the end. Simply creating a negative step in the Pre-Processor and submitting the simulation again can remedy this.

A simulation needed to be remeshed and a preprocessor could not generate a negative step prior to restarting the simulation. If this occurs, please review the simulation and make sure that no problems with boundary conditions occurred prior to the failure. Also, check that there is plenty of disk space available and that full write privileges are available in the directory of interest. If no reason can be determined, please contact SFTC.

Chapter 4: Post-Processor

Deform 3d post processor with a variety of features and graphics allows engineers to check the model results and present them in a way to understand the model results in an efficient manner. This section gives brief details of the systems and the available features. With every release the system is being enhanced to meet the industry requirement and specific user demands. Figure 4.0 shows the overall post processor system of Deform3D.

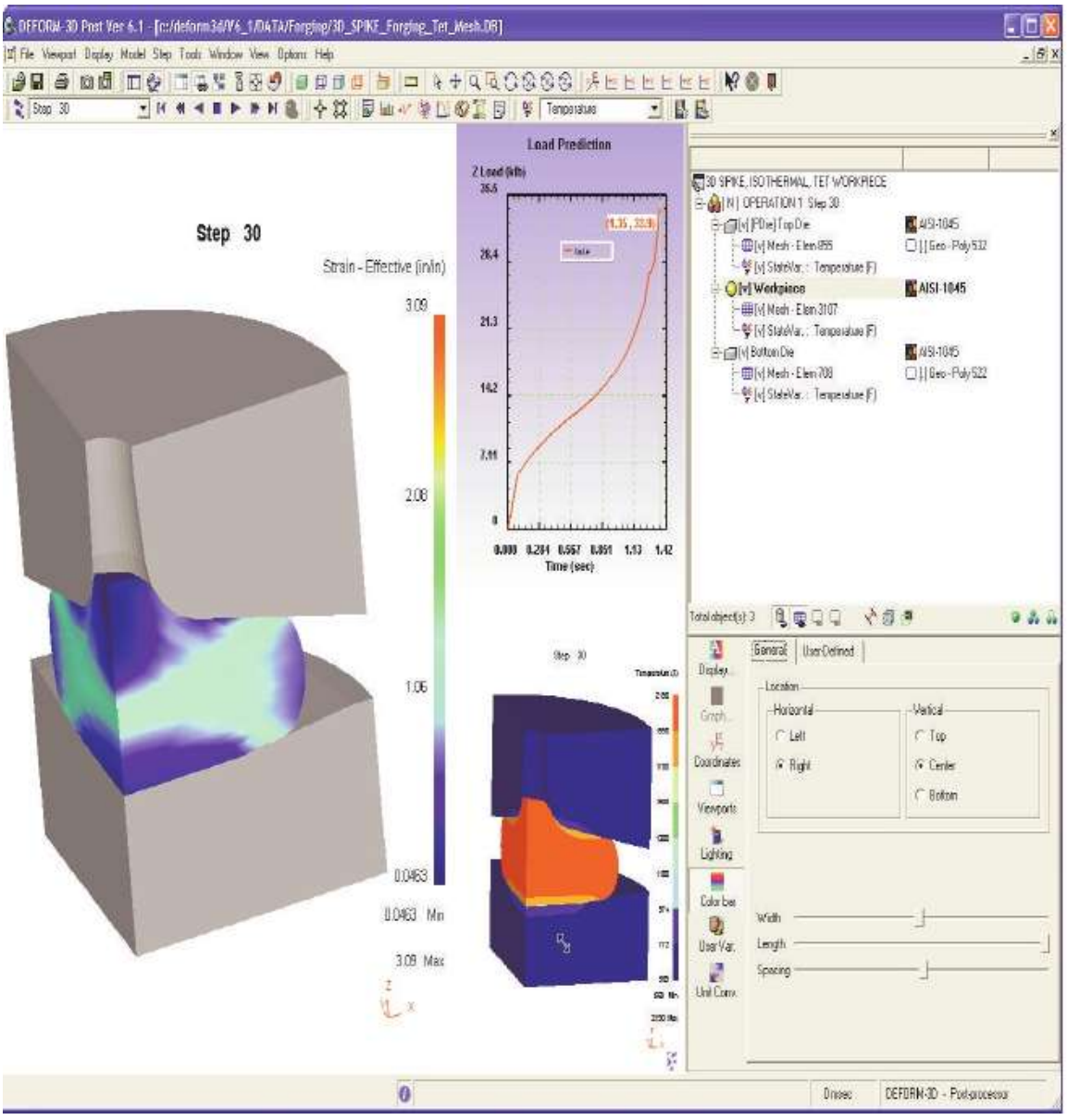


Figure 4.0: DEFORM-3D Post-Processor.

4.1. Post-Processor Overview

The DEFORM post-processor is used to view and extract data from the simulation results in the database file. All results steps which were saved by the simulation engine are available in the post-processor. Information which is available from the post-processor includes:

- Deformed geometry, including tool movements and deformed mesh at each saved step.
- Contour plots: Line or shaded contours display the distribution of any state variables, including stress, strain, temperature, damage, and others.
- Vector plots: displacement and velocity vectors indicate magnitude and direction of displacement or velocity for every node at each step throughout the process.
- Graphs of key variables such as press loads, volumes, and point tracked state variables.
- Point tracking to show how material moves and plots of state variables at these points.
- Flow net showing material flow patterns on a uniform grid. Generally a very good predictor of grain flow patterns in the finished part.
- State variables can be tracked between any two points and plotted in a graph format. The state variables can either follow the boundary or linearly between the points.
- A histogram plot of any state variable can be made to view the distribution of any given state variable throughout a body.

State variable, geometry, and image data can also be extracted in a number of neutral formats for use with other programs.

4.2 Graphical display

4.2.1. Window layout

The postprocessor has three windows: one to display information, one to manipulate the view, and one to control the information that is displayed.

Display window

The display window is the main postprocessor window as seen in Figure 4.2.1. It is the graphical display for all results information, including geometry and meshes, contour and vector plots, and graphs.

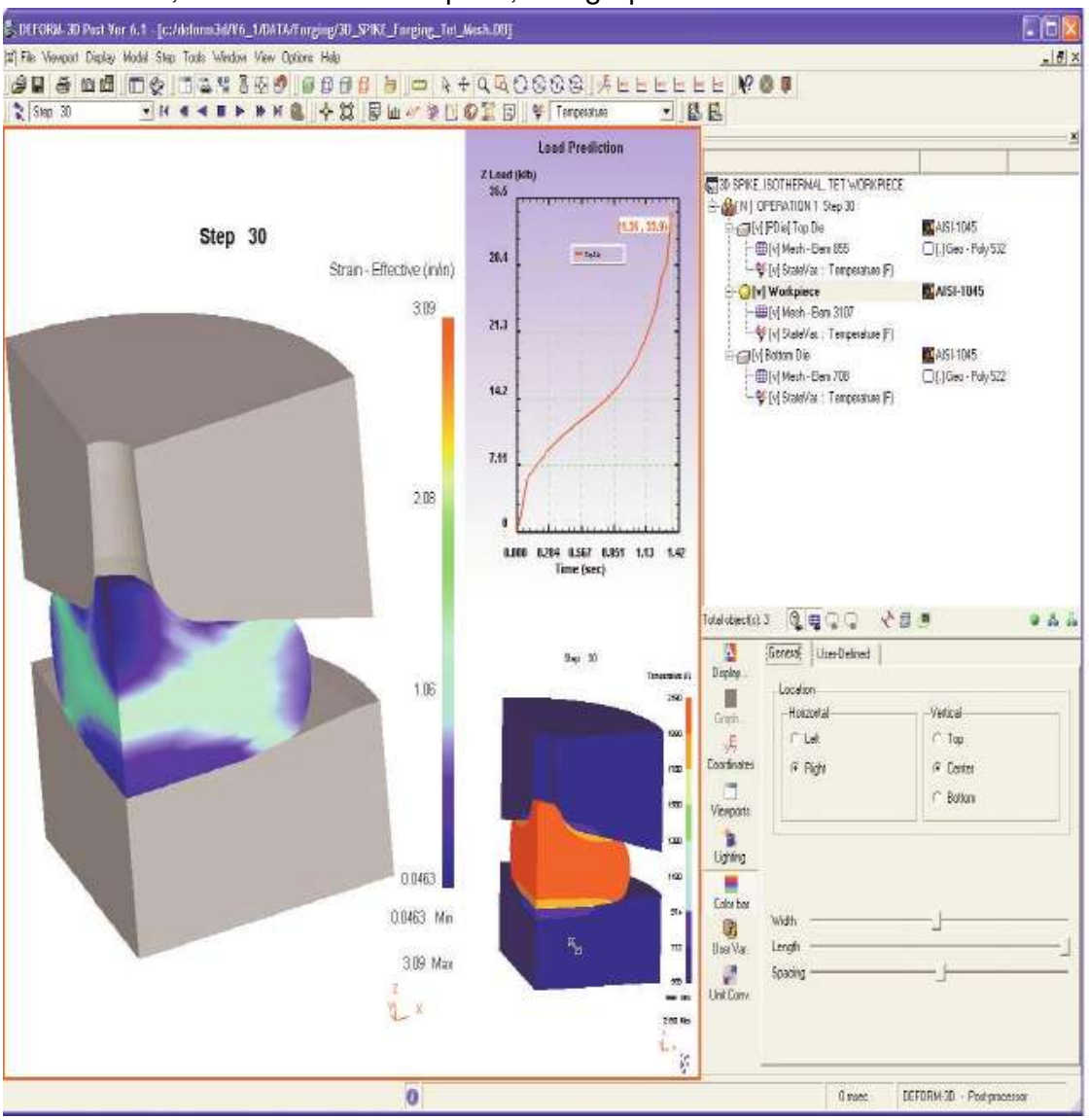


Figure 4.2.1: Display window.

Graphic Utilities

The graphics utilities window provides view manipulation and other utility functions for the object window. Features include zoom, pan, and rotate, measurement utilities, multiple viewport controls, printing, and animation creation utilities. The graphics utilities window also contains step control buttons to continuously play through the database, to single-step forward or backward, or to jump to the beginning or end of the database. (Figure 4.2.2

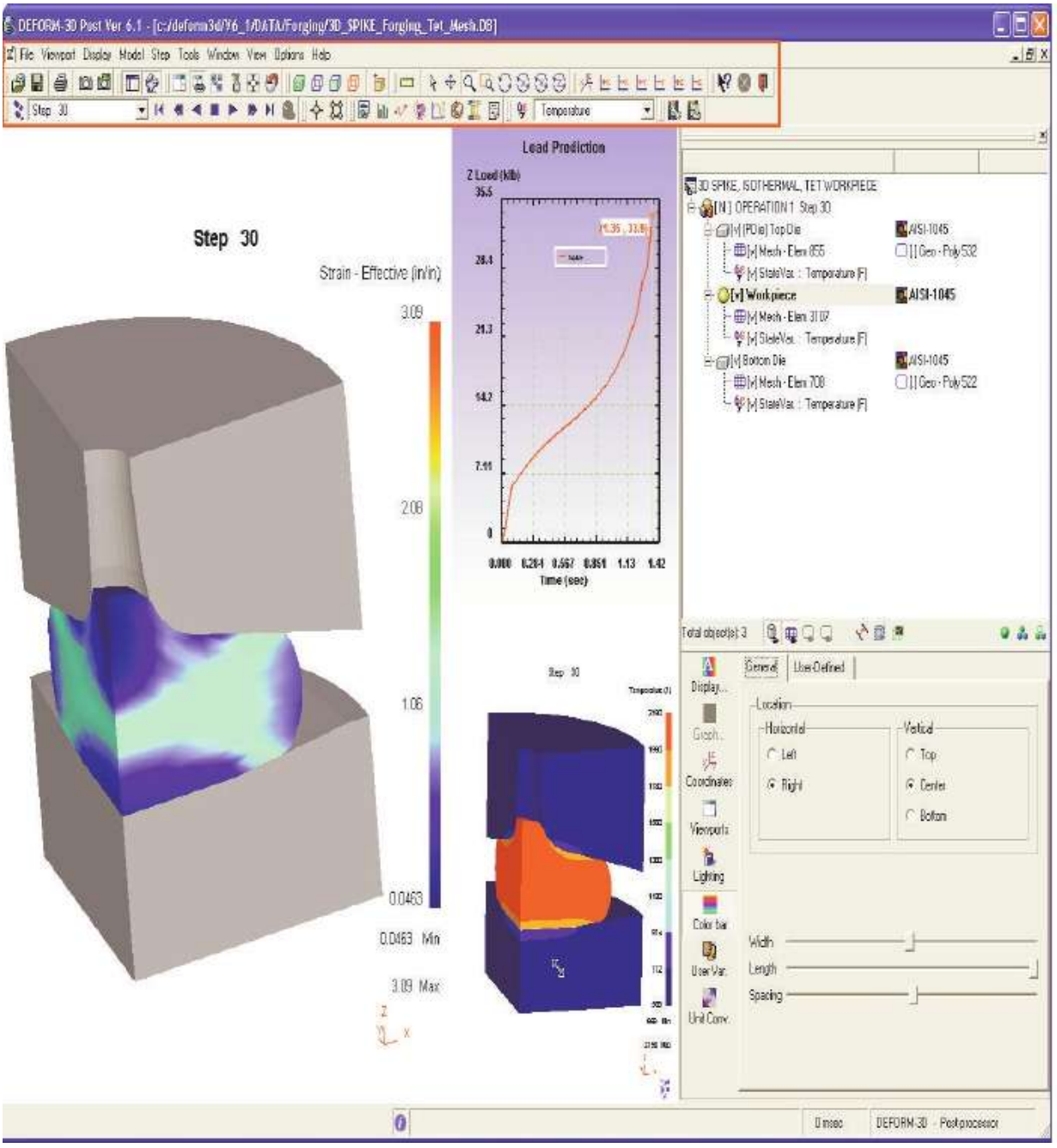


Figure 4.2.2: Icon toolbar.

Post-processor controls

The post-processor controls is used for selecting the specific data set to be displayed in the postprocessor window, and for extracting solution data from the database and writing it to text files. (See Figure 4.2.3 and Figure 4.2.4)

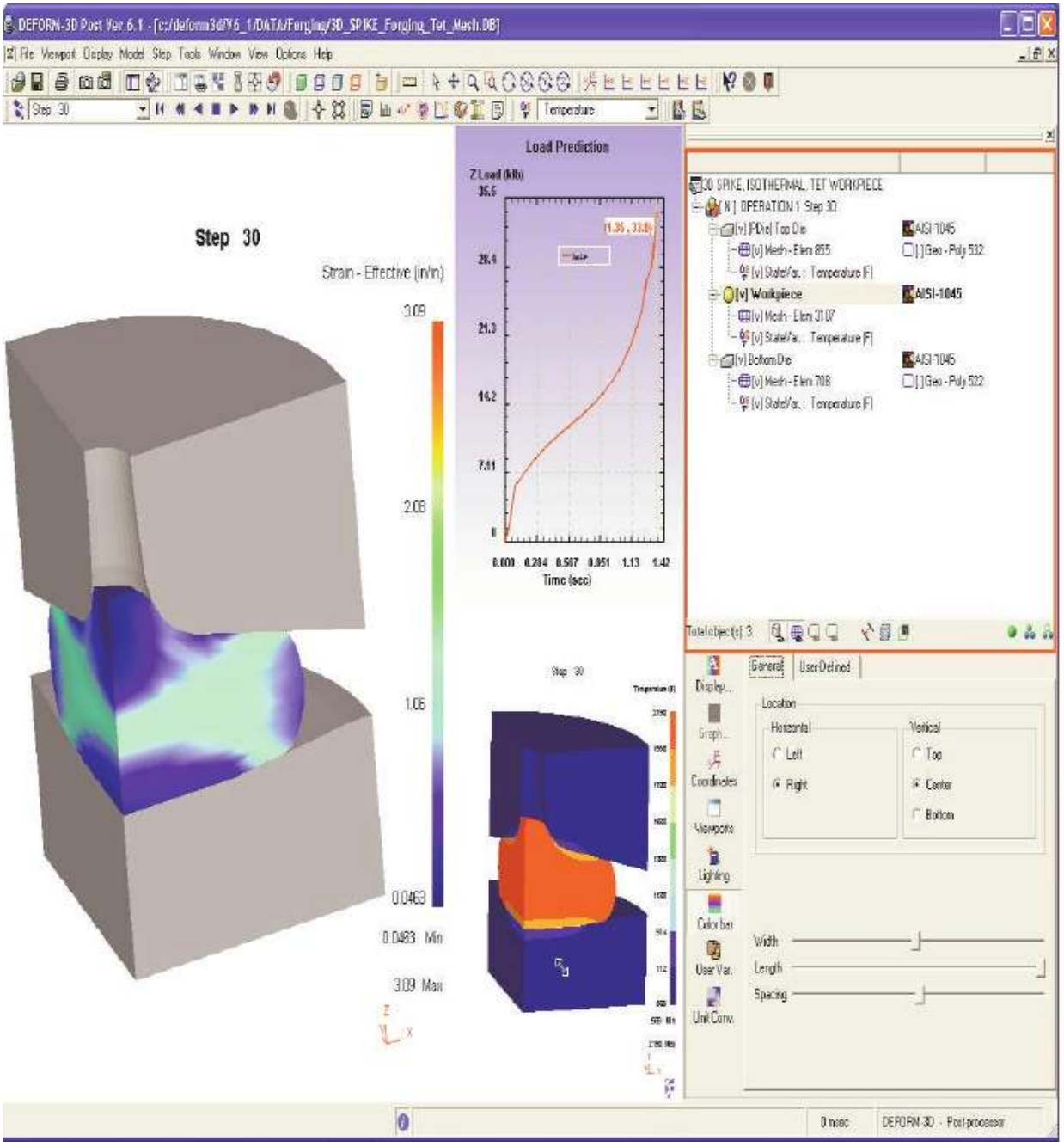


Figure 4.2.3: Display list and controls.

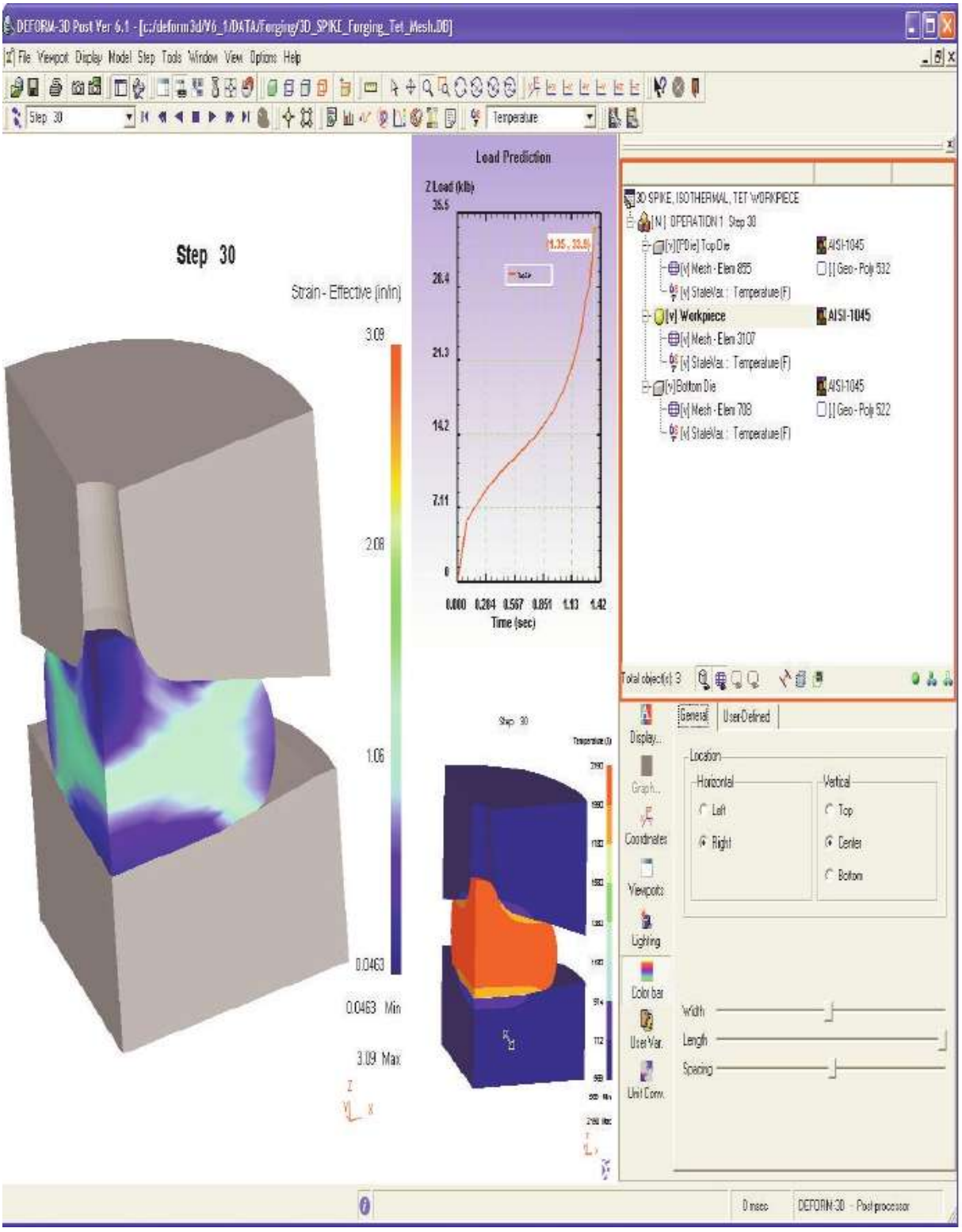


Figure 4.2.4: Display properties.

Right-mouse click menu
Clicking the right mouse button in the display window yields the window seen in Figure 4.2.5.

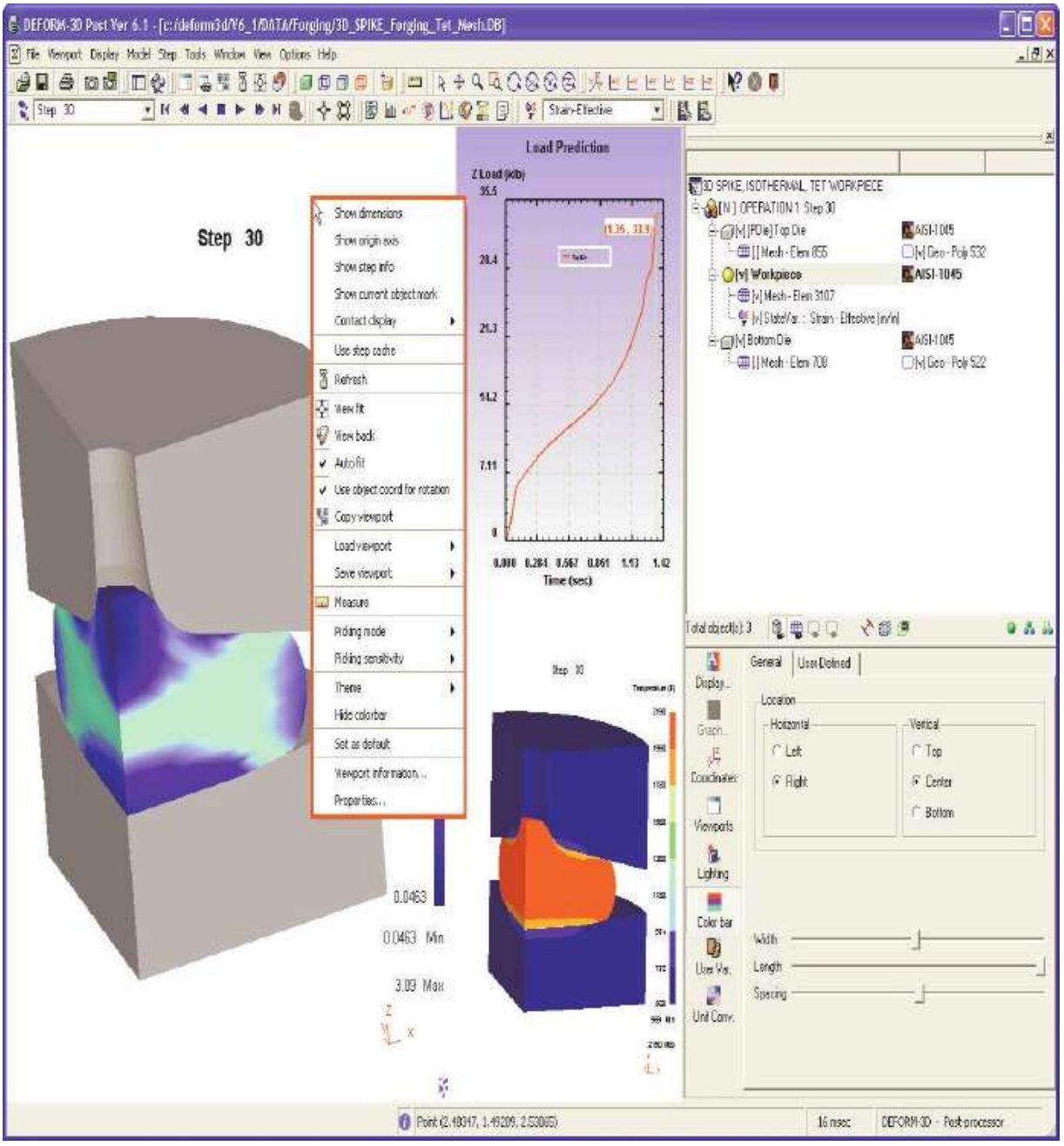


Figure 4.2.5: Right-mouse click window in post-processor.

Object Display Modes

DEFORM has 3 different object display modes as seen in Figure 4.2.6:

- Single Object mode – the selected object is displayed. All other objects are hidden
- Multi Object mode – the selected object is displayed in solid color. All other objects are transparent
- User Object mode – the user can set the display mode (on, transparent, or off) for each object independently.

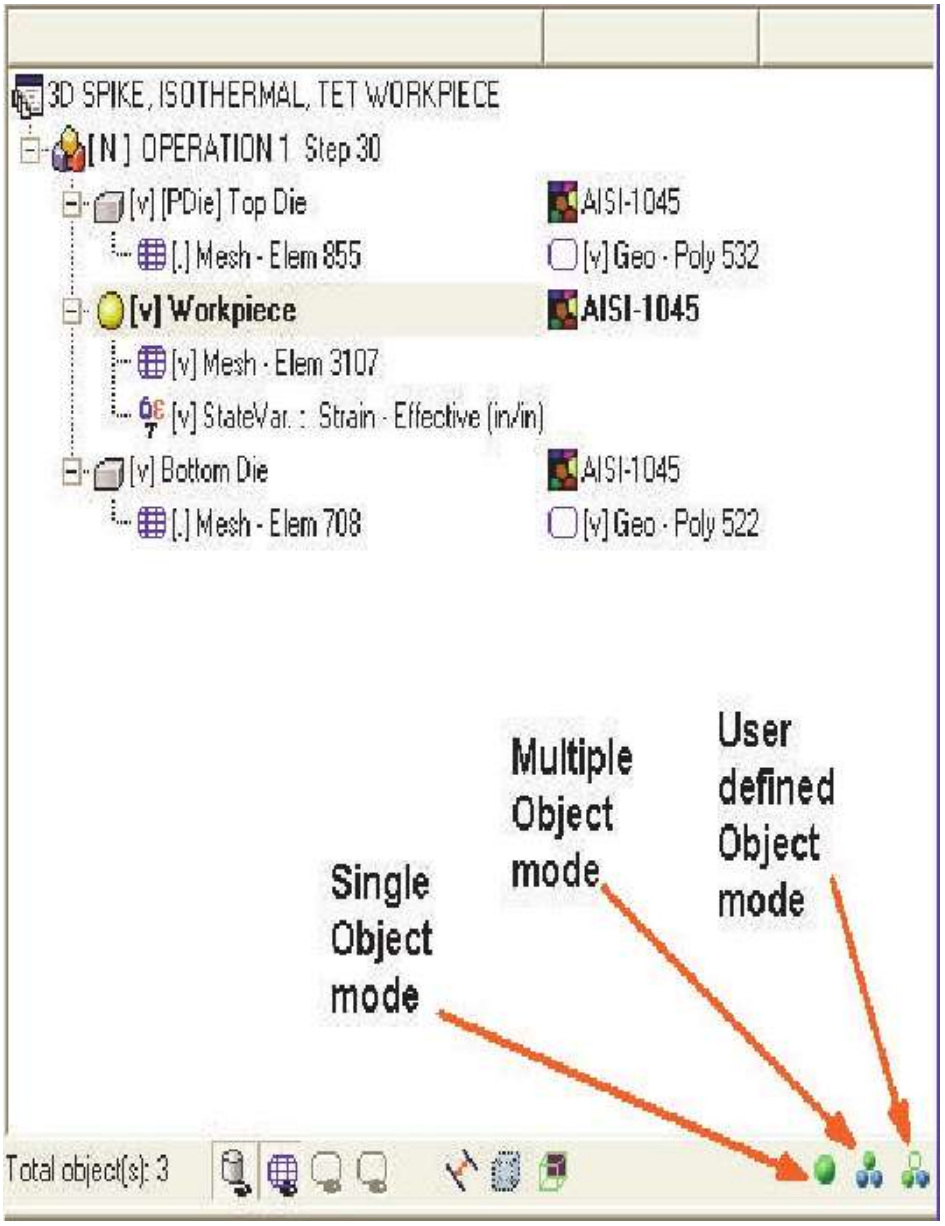


Figure 4.2.6: Detail of Object Tree showing object mode selection icons

Tree levels and functions
The right mouse button menu has functions associated with each level of the object tree. The object tree levels are shown in Figure 4.2.7.

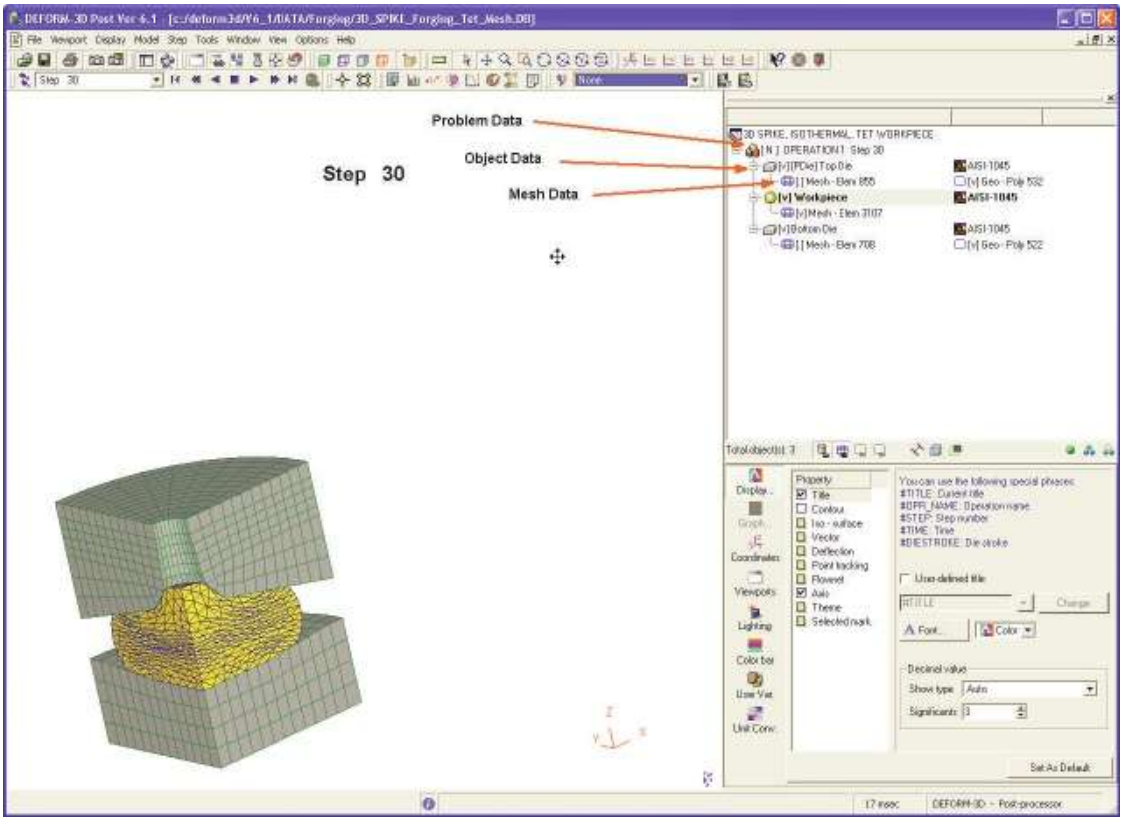


Figure 4.2.7: Levels in the object tree.Right mouse selection at each level accesses a context appropriate menu.

The **Problem Data** menu contains the following commands:
- Turn on all objects
- Turn off all objects
- Turn on all work pieces - Turns on any object which is not rigid. Turns off all rigid objects)
- Turn on all dies - Turns on any object which is rigid. Turns off all other objects) Turn on transparency for all
- Turn off transparency for all
- Turn on backface for all - backface shows the interior or back surface of rigid objects
- Turn off backface for all

The **Object Data** menu contains some or all of the following commands, depending on object type:
- Turn on this only – turns on the selected object and turns off all other objects
- Turn off – turns off the selected object

- Show contact node – highlights any nodes which are in contact with any master object. This is a quick way to display contact. This is a toggle menu selection. Select it once to turn the contact node display on. Select it again to turn the display back off.
- Show BCC – highlights any node with constrained velocity boundary conditions. This is also a toggle selection
- Show geometry normal vector – displays vectors normal to each surface facet.
- Make transparent
- Show backface – backface shows the interior or back surface of rigid objects

The **Material Data** menu allows access to the material properties window (same window as the preprocessor).

A meshed rigid object will contain both mesh data (for temperature calculations) and rigid surface geometry data (for contact with deforming objects). The mesh and geometry data menus allow control of the mesh and geometry surface display.

The **Mesh Data** menu controls display of the object surface mesh description
- **Show Mesh** / **Hide Mesh** shows or hides the display of the object surface mesh
- **Change shade color** – changes the element fill color
- **Change line color –** changes the color of the lines delineating element edges

The **Geometry Data** menu controls the display of the surface geometry description
- **Show Geometry** / **Hide Geometry** shows or hides display of the surface geometry
- **Change shade color** – changes the surface facet fill color
- **Change line color –** changes the color of lines delineating facet edges

Additional Post Processing Functions
When various post processing functions (state variables, load stroke curves, slicing, etc) are displayed, the respective icons will be added to the object tree. Right clicking on these icons allows editing of their respective properties

Object Control Bar

All right mouse menu functionality is duplicated in a control bar at the bottom of the object tree. Figure 4.2.8

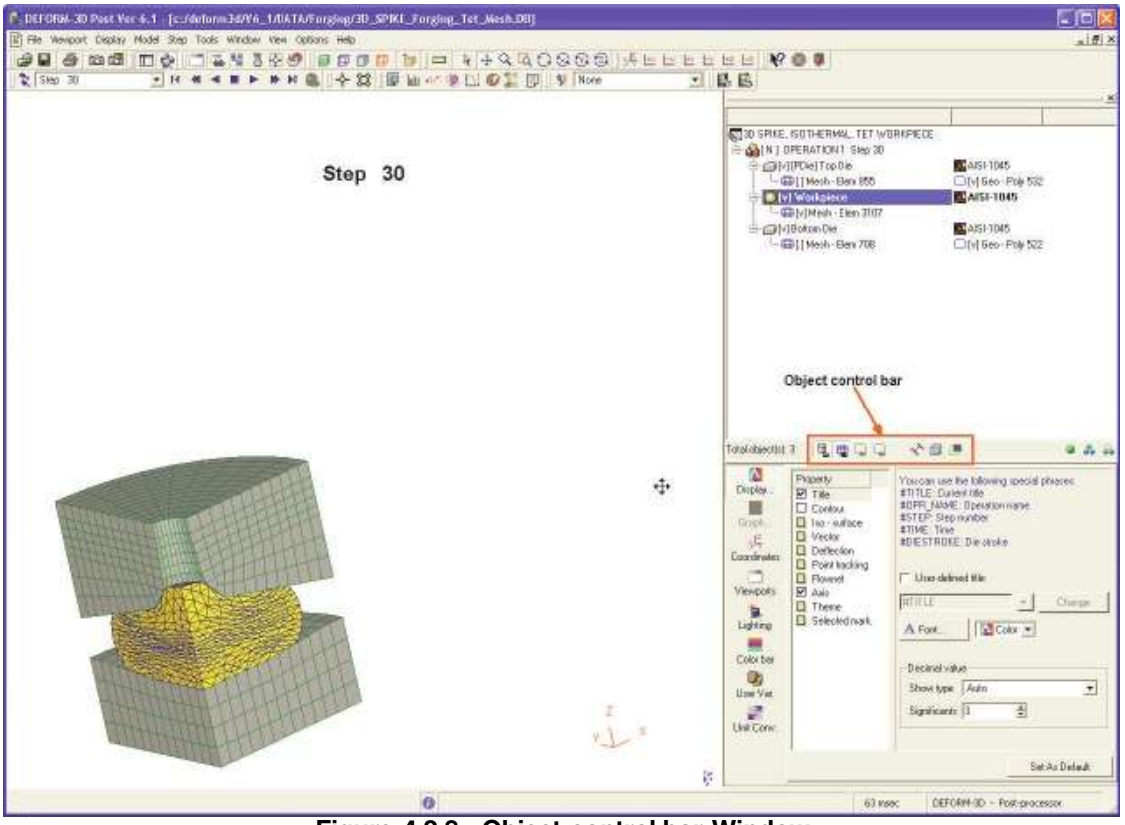


Figure 4.2.8: Object control bar Window

 Toggle object on/off (displays mesh, geometry, or both, depending on which is available and selected. Defaults to mesh if neither is selected)

 Toggle mesh on/off

 Toggle geometry on/off

 Toggle item on/off – may be any item, such as slicing, a curve or graph, etc.

 Display contact nodes

 Toggle transparency

 Toggle backfacing

4.3. Post-Processing Summary

The post-processor controls window is divided into two sections. The top sections consist of links to several sub-windows which access virtually all post-processor data display and extraction functions. The bottom sections contain hot-link icons which display several of the most commonly used functions.

4.3.1. Simulation Summary

This is very similar to the old DEFORM™ simulation summary. (see Figure 4.3.1)The step can be selected using the step list and then the object can be selected using the up/down arrow buttons in the object field. The concept of Operation Number is going to be added to DEFORM™-3D, until then all the operations have the number 1. After changing the object, if the step changes,the object does not change. The up/down arrow keys can be used to browse through the step list.

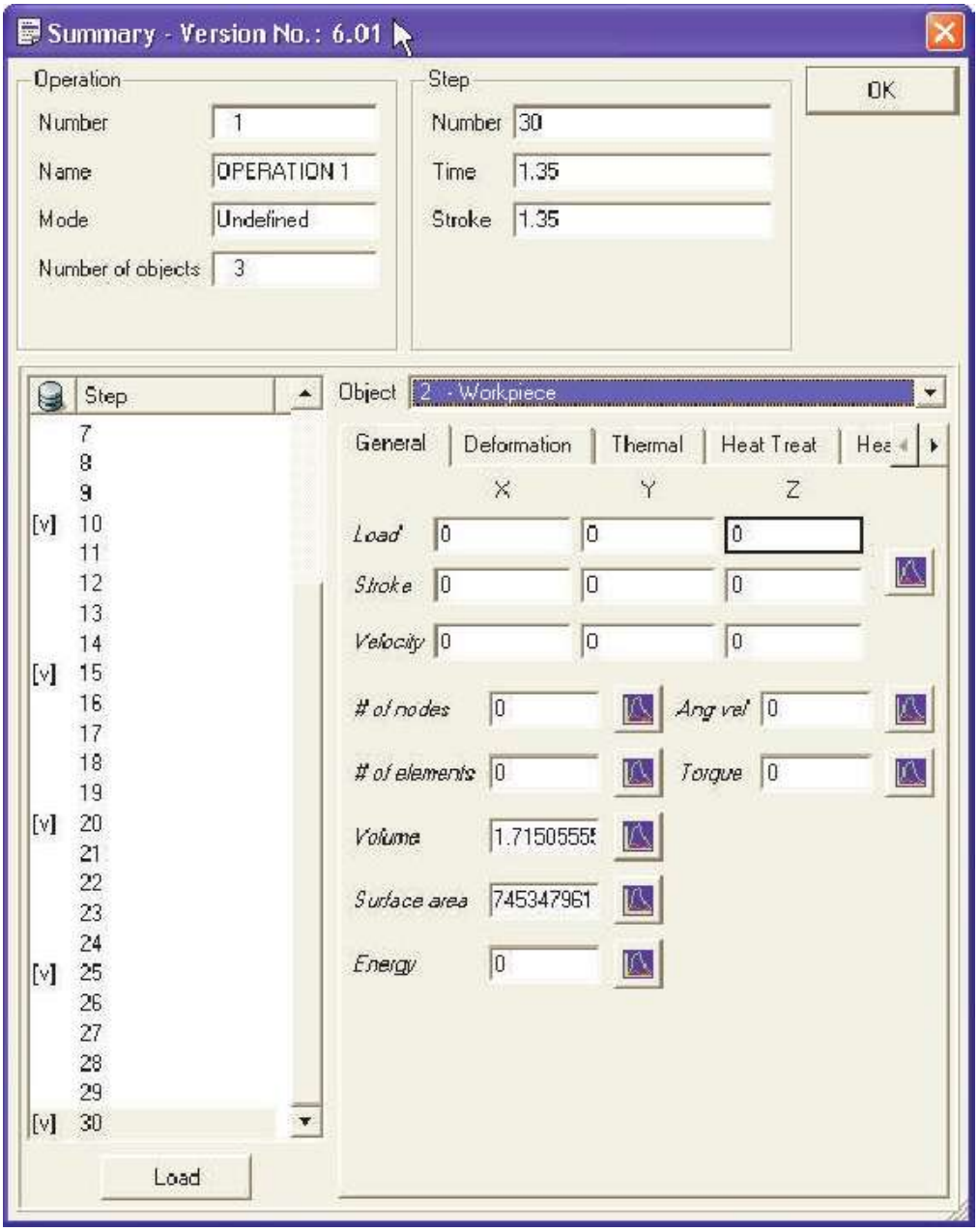


Figure 4.3.1: Simulation Summary Window

Certain characteristic data, such as press loads, principle die velocities, and maximum and minimum values of state variables are stored for every simulation step, whether complete data is stored for that step or not. This summary data can be viewed in the simulation summary window.

4.3.2.State Variable

To plot state variable plots, the variable and then the component of the variable must be selected(see Figure 4.3.2). Then the scaling type global (min/max of all objects for all simulation steps), local (min/max of all objects for the particular step), and user-defined (which defaults to the global values) must be selected. For user-defined, the min/max values can be entered in the input fields provided to the right of the window. Then the type of contour (line/shaded/vector) and the class of objects to be plotted can be selected, and finally specific objects can be selected/omitted by toggling them on/off in the object list. Clicking on the **OK** button plots the state variable in the current viewport.

The following state variables can be plotted in the post-processor:

Analysis variables:

- Minimum distance
- Contact time
- Folding angle
- Surface expansion ratio
- Surface area

Tool Wear

- Interface temperature
- Sliding velocity
- Interface pressure
- Wear pressure (current)
- Wear depth (total)

Deformation

- Coordinates
- Damage
- Displacement
- Strain effective
- Strain nodal
- Strain total
- Strain rate Stress
- Velocity
- Back stress
- Normal pressure

Thermal

- Temperature

Microstructure

- Volume fraction
- Grain Model
- Hardness
- Cooling time
- Cooling rate

Diffusion

- Dominant Atom

Heating

- Voltage

Property

- Material group

User

- User nodal variables
- User element variables
- User variables

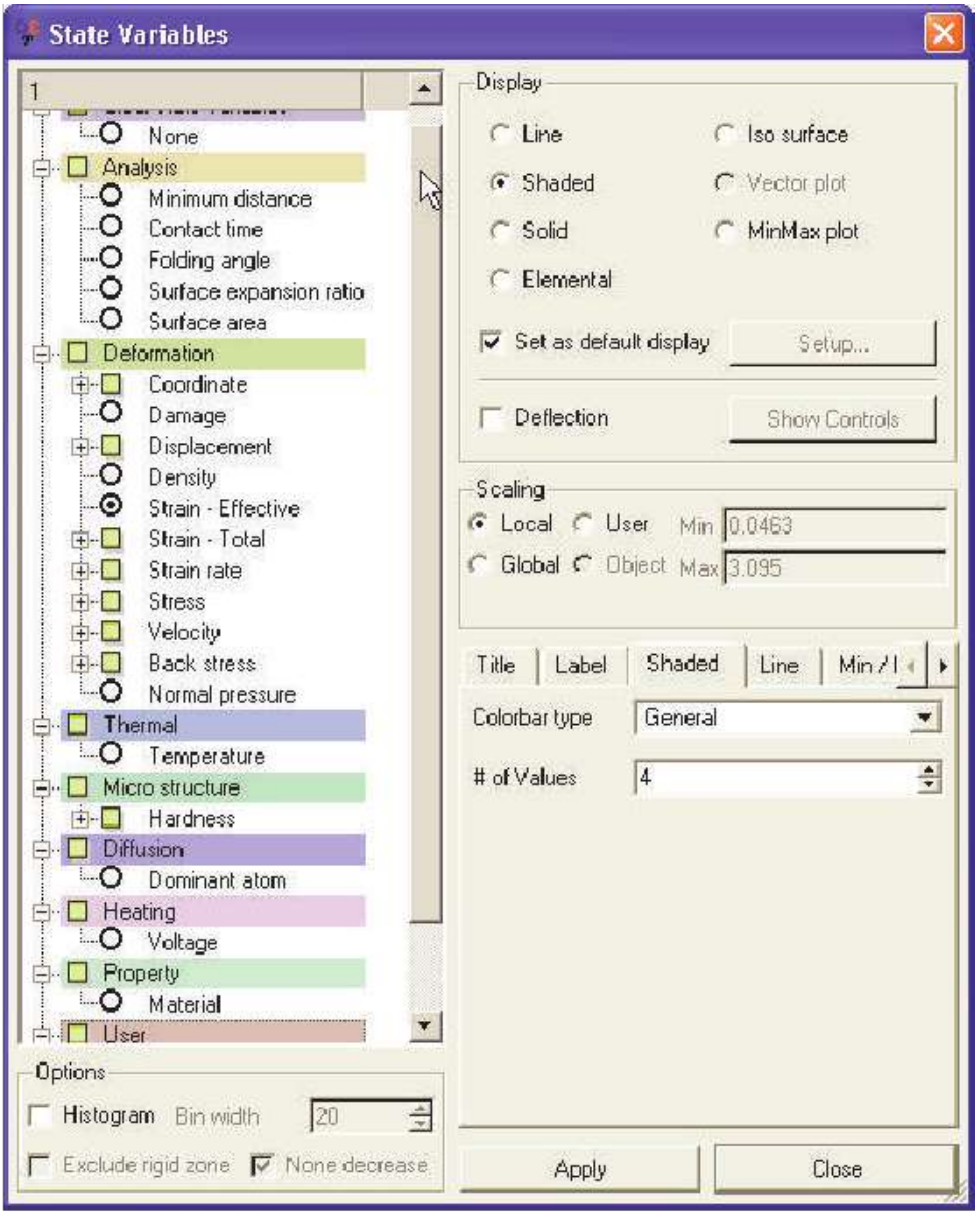


Figure 4.3.2: State Variable Plots Window.

Plot Type

Line contour

Shows the state variable selected as a line contour plot.

Shaded contour

Shows the state variable selected as a shaded contour plot.

Solid contour

This is the same as a line contour plot with solid colors in between the lines.

Elemental contour

This plot type colors each element a distinct color based on value at the element without interpolating values between elements.

Iso-surface

This plot type shows only the surface of a particular value defined by the slider bar on the bottom of the state variable window.

Vector plot

This plot type shows the variables as vectors with a color associated with the value and a direction.

Deflection

This plot type will distort the object (elastic body only)

Histogram

See Figure 4.3.3 as an example.

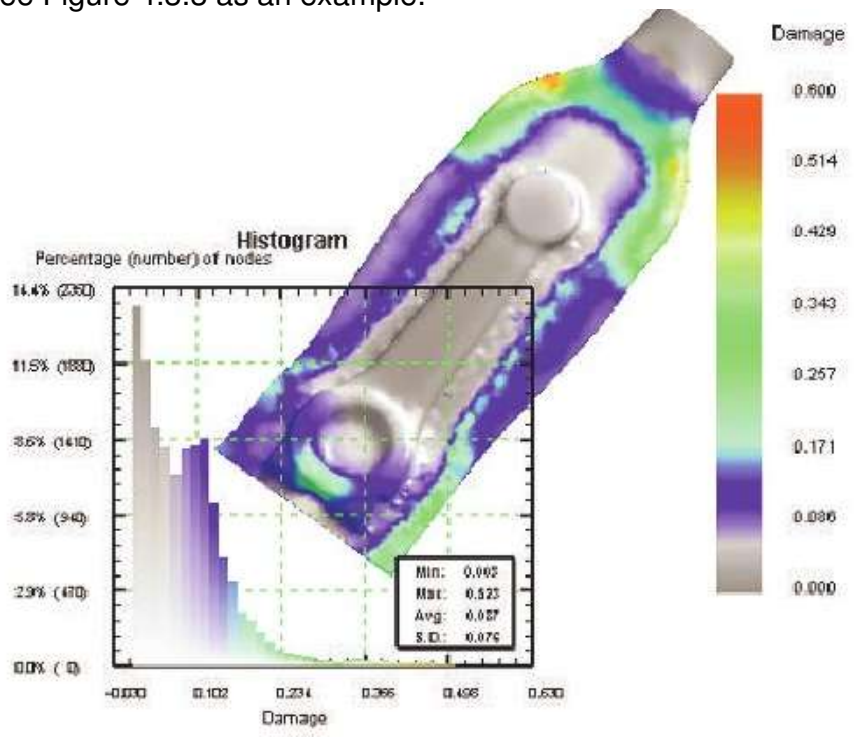


Figure 4.3.3: An example shaded contour plot with a histogram distribution.

Scaling

Local (current step)
There is a different minimum and maximum value for every step.

Global (all steps)
The same minimum and maximum value will be used for every step.

Global (user defined)
The same minimum and maximum value will be used for every step, which the user can define.

Global (each step)
There is a different minimum and maximum value for every step, which the user can define.

Object Limits
The Object Limits allows the user to set different minimum and maximum values for different objects. This is only activated for line contours.

Interpreting state variables

Damage

Damage generally relates to the likelihood of fracture in a part. The specific definition of damage is dependent on the method of calculation selected in the pre-processor. Damage is NOT a good indicator of fracture in tooling. Stress components should be used for die failure analysis.
Damage, particularly the Cockcroft-Latham damage model (the default damage model in DEFORM) has been shown to be a good indicator of certain types of tensile ductile fracture in cold parts (cracking due to deformation by stretching, such as chevron or surface cracking in extrusions, or cracking on the outside surface of an upset). It is not a good indicator of fracture in compression (such as splits perpendicular to die motion due to extreme heading reduction).
The damage value at which fracture initiates varies substantially from material to material, and can even vary for a given material with different annealing treatments. However, for a given material with a given annealing treatment, critical damage value at fracture is reasonably repeatable.

Damage value can be used in two ways:

1. Evaluating alternatives: In problem solving a job that is known to fracture, or in analyzing a job where ductile fracture is known to be a risk. Several alternatives can be analyzed in DEFORM. The alternative with the lowest damage value is the best alternative for minimizing the likelihood of fracture.

2. Comparing a design to a known critical value. Critical damage values can be estimated from prior experience with a given material on a part that is known to fracture. Running a DEFORM simulation of a process known to cause stretch cracking in the part will give an upper bound value for damage. Running a simulation on a part made of the same material that is known *not* to crack will give a good lower bound value. The ideal part is a marginal process, that is: one that cracks occasionally, but not on every part. If the peak damage value from the DEFORM analysis corresponds with the fracture site on the part, this will give a good estimate of the critical value. Designs with a damage value 10% to 20% or more below this value should be safe from fracture, if material and anneal conditions are the same.

Displacement

For small deformation problems only, plots the nodal displacement value. For large deformation, the displacement since the last remesh will be plotted. This variable is primarily intended for die deformation analysis.

Density

For powder or porous materials, plots the relative density distribution. Theoretical maximum limit is 1.0.

Strain

Strain is a measure of the degree of deformation in an object. A detailed description of strain is available in any standard text on mechanics of materials, metal forming analysis, or plasticity. A brief description of nomenclature will be given here.

The measure of strain used in large deformation analysis, including DEFORM is *true strain*, which differs slightly from the well known *engineering strain* presented in typical engineering applications.

Engineering strain is defined as

$$\frac{change\ in\ length}{original\ length}.$$

which is a good approximation for small deformations, but loses accuracy when deformations become large.

For large deformation analysis, it is better to use true strain, which is defined as the sum of a large series of arbitrarily small strain increments. Integrating

this over the total change in length gives $\epsilon = \ln \frac{l_f}{l_0}$ where ϵ is the true strain, l_0 is the initial length, and l_f is the final length.
Extending this deformation into three dimensions, we can assume a cube with initial dimensions x_0, y_0, and z_0. Components of strain in the *x*, *y* and *z* directions can be defined incrementally as $\epsilon_x = \frac{dx}{x_0}$, $\epsilon_y = \frac{dy}{y_0}$, and $\epsilon_z = \frac{dz}{z_0}$ where *dx*, *dy*, and *dz* are incremental deformations in the x, y, and z directions.

Shearing components of strain can be defined incrementally as $\epsilon_{xy} = \frac{dy}{x_0}$, $\epsilon_{yz} = \frac{dz}{y_0}$, and $\epsilon_{zx} = \frac{dx}{z_0}$ where ϵ_{xy} , ϵ_{yz} , and ϵ_{zx} are shear strains in the xy, yz, and zx planes, respectively.
In a two dimensional analysis, the yz and zx components of shear strain are assumed to be zero.
Through various mathematical techniques which are beyond the scope of this discussion, it is possible to define so-called ``principal axes" on which all components of shear strain are zero. The strains measured along these axes are termed ``principal strains."
It is frequently useful to have a single characteristic strain value to describe the degree of deformation. DEFORM uses a value common to metal forming analysis known as the effective or Von-Mises strain

$$\bar{\epsilon} = \frac{\sqrt{2}}{3}\sqrt{(\epsilon_1 - \epsilon_2)^2 + (\epsilon_2 - \epsilon_3)^2 + (\epsilon_3 - \epsilon_1)^2}$$

where ϵ_1 , ϵ_2 , and ϵ_3 are principal strains, and $\bar{\epsilon}$ is the effective strain.
For porous materials, there is also a volumetric component of strain. This change in volume relative to initial volume is stored as mean strain. For elasto-plastic materials, the strain values stored are only the inelastic parts of the total strain. Elastic parts are available through the specification in the simulation controls window.

Strain Rate

Strain rate is a measure of the rate of deformation with respect to time. The units are strain per second where strain is a dimensionless value. The components of strain rate are defined in the same manner as the components of strain.

Stress

Stress is defined as the force acting on a unit area of material. Assume a unit cube of material. Forces (or stresses) acting on the faces of the cube can be resolved into normal (perpendicular to the face) and shear (along the face).

Shear stresses can further be resolved into two components along arbitrary orthogonal axes. Thus, the complete stress state can be defined by three normal stress components σ_x, σ_y, and σ_z, and six shear components σ_{xy}, σ_{yx}, σ_{yz}, σ_{zy}, σ_{zx}, and σ_{xz}.
Through equilibrium conditions it may be shown that the components acting in the same plane are equivalent. That is $\sigma_{xy} = \sigma_{yx}$, $\sigma_{yz} = \sigma_{zy}$, and $\sigma_{zx} = \sigma_{xz}$. Thus, the complete stress state can be represented by 3 normal components and 3 shear components.
As with strain, by mathematical analysis it is possible to orient three orthogonal axes such that the shear component of the stress along these axes is 0. The resultant normal stresses acting on these axes are termed principal stresses.
DEFORM uses the von Mises stress to define the characteristic ``effective stress''. The effective stress $\bar{\sigma}$ is defined as

$$\bar{\sigma} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2}$$

where σ_1, σ_2, and σ_3 are the principal stresses. For most metals, the effective stress is indicative of the onset of plastic flow.

Velocity

The velocity option plots nodal velocity at each step. Vector plots display magnitude and direction. Magnitude is indicated by vector length and color. Contour plots display only magnitude, where contour color indicates velocity magnitude.

Normal Pressure

Normal pressure is the force per unit area on the surface of an object. This is computed on the surface of slave objects. The range of values is from no stress to compressive stress. Tension cannot be maintained without some form of sticking applied between the surfaces.

Temperature

The temperature plot displays nodal temperature at each step.

Volume Fraction

If transformation calculations are performed, the volume fraction of the selected phase component of a mixture material is displayed.

Grain Size

If the grain calculations have been turned on in the simulation controls, the corresponding variables such as hardness, cooling time and cooling rate can be examined using corresponding variables.

Hardness

Predicted hardness based on the hardness method selected in the pre-processor will be displayed.

Dominant atom

The concentration of the dominant atom (usually carbon) in a diffusion calculation is displayed.

User Variables

User defined nodal variables can be stored using user defined FORTRAN subroutines. Refer to the section on user subroutines for more information. Two classes of variables are available in DEFORM system. First class of user variables, can be defined along with the model definition in the preprocessor, and user can use his own user routines to update these values at the nodal and element levels. Second class of user variables are available at the post processor level, as user defined Post variables. For the second class of variables user need not rerun the simulation, but a variety of user variables can be computed based on the model results. Tool wear computation that have been added from 3DV61 belong to these second class of variables.

4.3.3. Point tracking

In Point Tracking you can track 200 of points over the course of a simulation and view how they compare with each other. First select the desired starting step from the Start Step list in the Point Tracking window. Now select the object from the object table. Once you have done this, click on the **Define Material Points** button. You can now add the points that you wish to track in the Display window.(see Figure 4.3.4)



Figure 4.3.4: Point tracking data window.

Adding Points
Points can be added to track by clicking in the intended location with the left mouse button when this button is selected.

Deleting Points
Points can be deleted by clicking the third icon from the top in the Point tracking data window (See Figure 4.3.4).

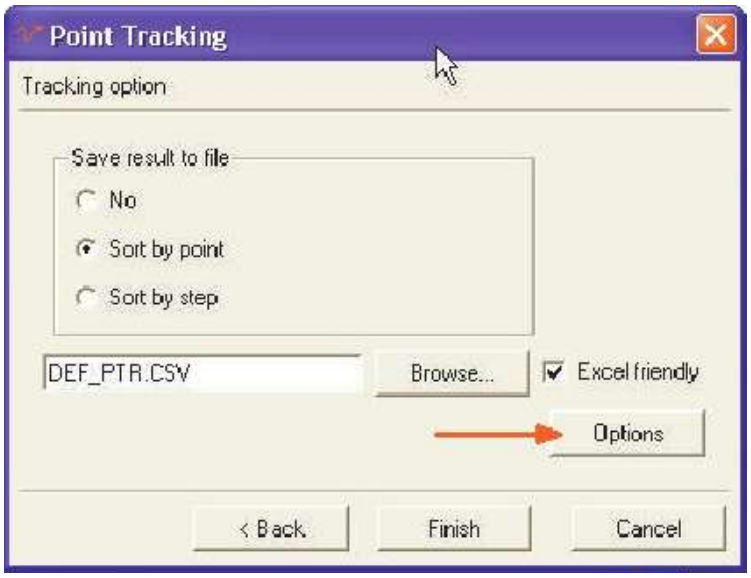


Figure 4.3.5: First window of point tracking wizard.



Figure 4.3.6: First window of point tracking wizard .

Point tracking can be performed in either the Cartesian coordinate system or in a cylindrical coordinate system. The variables that can be shown in cylindrical

format are the strain components, the stress components and the strain rate components. The results from point tracking can be saved in an Excel friendly format, (see Figure 4.3.5), and from version 3DV61 user can selectively save these variables. (see Figure 4.3.6)

4.3.4. Load stroke curves

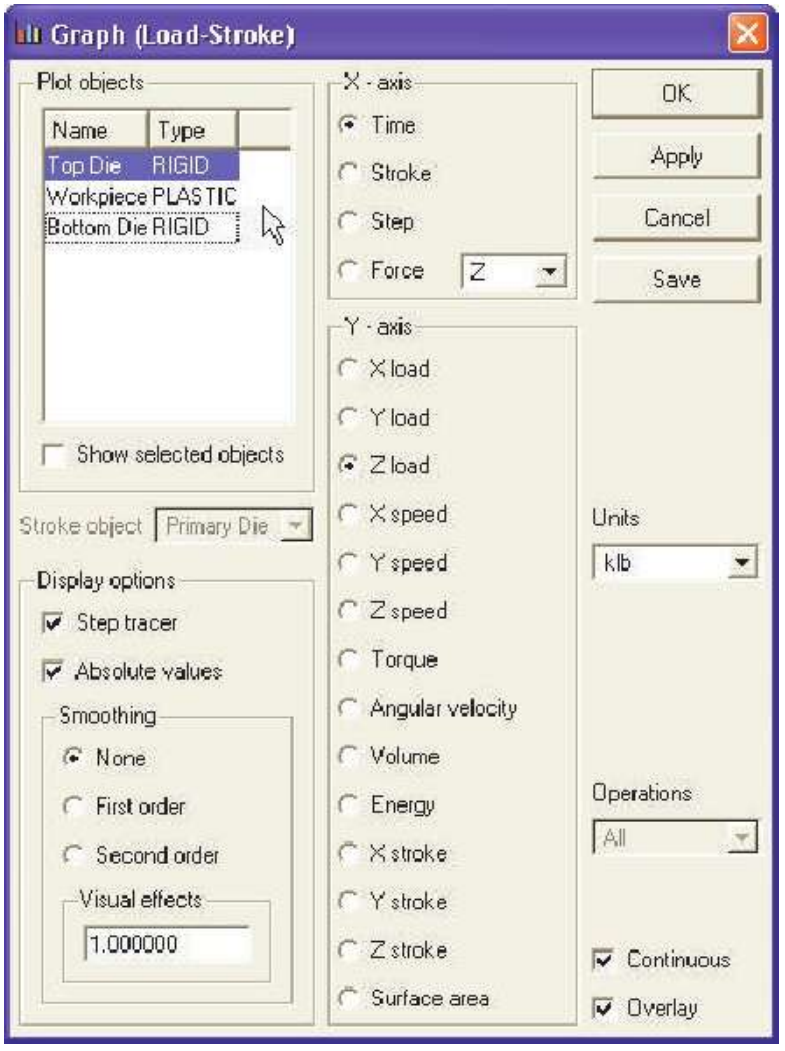


Figure 4.3.7: Load-stroke window.

The graphs window is used to generate load, speed, torque, angular velocity, and volume vs. time (or stroke) plots for the object(see Figure 4.3.7). Multiple plots can be generated on the same graph. If time is used as the x-axis, then the graph can be used for selecting steps. Clicking on a point on the graph will load the nearest saved step from the database.
A note on volume plots: due to various factors, some volume loss is unavoidable in FEM analysis. However, if volume loss exceeds more than about 1% of total volume, this should be a cause for concern.

Causes of volume loss and their remedies include:

- Corner cutting during remeshing. This is characterized by large drops in volume at remeshing steps. If elements around a relatively tight corner are too large, the portion of the element which penetrates the corner will be lost on remeshing. This can be controlled forcing a finer mesh in that region with mesh density windows, and by using the Maximum Interference Depth remesh trigger.
- Excessive hydrostatic stress or too small a volume penalty constant. This is characterized by a steep gradient in the volume curve between remeshings. Check that flow stress data is using the correct units. For problems such as high ratio extrusion with extremely high hydrostatic stress, it may be necessary to increase the volume penalty constant by an order of magnitude or two. Increasing the penalty constant may lead to convergence problems, so a balance must be struck.
- Volume compensation (*Preprocessor* → *Objects* → *Properties*) can be used to control volume loss during remeshing.
- Polygon sub stepping can be used to limit volume loss during simulation

4.3.5. Coordinate Systems

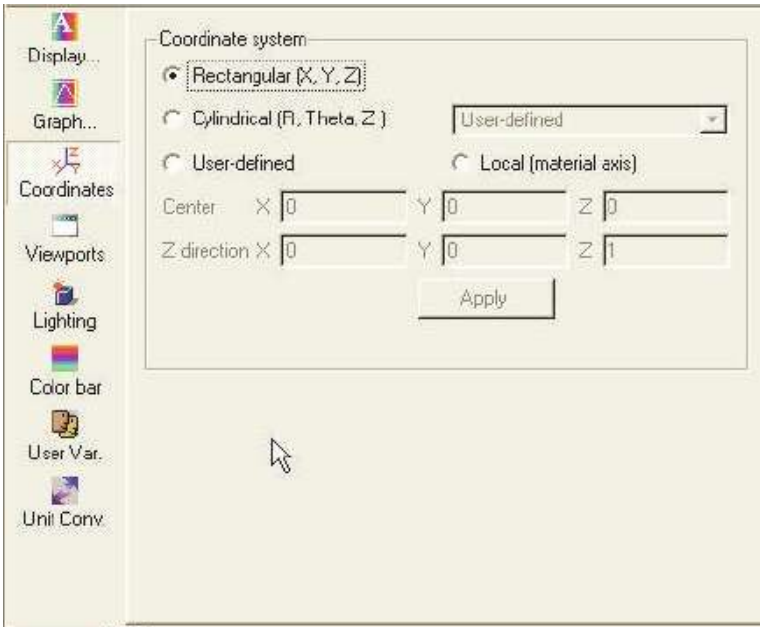


Figure 4.3.8: Coordinate system definition.

There are currently two types of coordinates systems for which stress, strain and strain rate components can be viewed: Cartesian and Cylindrical. Cartesian coordinates are the default setting, however the Coordinates button in the display properties window allows the user to switch to Cylindrical Coordinates (See

Figure 4.3.8). Clicking the Coordinates button brings up a window that allows the user to select the cylindrical coordinates. The user needs to define by z-axis for the cylindrical coordinates. This axis is defined through a single point on the z-axis and one vector direction. This works for contour plots and for point tracking. The cylindrical coordinates won't be viewed until the user selects either a stress, strain or strain rate coordinate.

4.3.6. Step Selection & Manipulation



Figure 4.3.9: Step manipulation buttons.

The display window is used to graphically display simulation results. Results for each step are selected from the step list. DEFORM™-3D allows a new method of selecting and moving between steps. The primary way of selecting a step from the step list is using the step list pulldown and clicking on the desired step (See Figure 4.3.9). The Step List can also be traversed using the Step Manipulation Buttons. These buttons and their function are explained in the following section.

Step Manipulation Buttons

Rewind

 Will rewind the step list back to the first saved step.

Reverse

 Will rewind the step list back one saved step

Forward

 Will move to the next saved step in the step list.

Fast Forward

 Will fast forward to the last saved step in the step list.

Play (Forward)

 Will display the steps one by one until the last step is displayed, at which point it will return to the first step and continue.

Play (Reverse)

 Will display the steps one by one in reverse order until the first step is displayed, at which point it will return to the last step and continue.

Stop

 Stops the playing (forward or backward) of steps.

4.3.7. Steps list

The step options such as adding and subtracting steps from the Step List are accomplished from within this window. By default when DEFORM™-3D loads up, only positive steps are displayed in the step list. (see Figure 4.3.10)

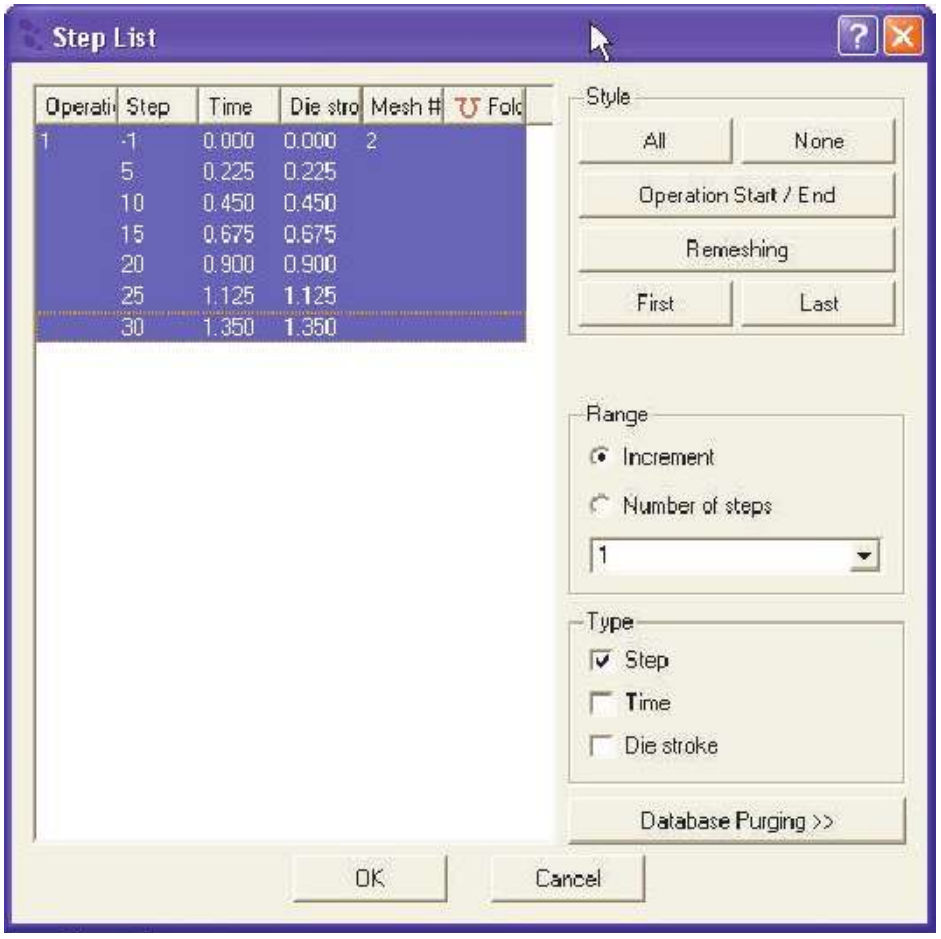


Figure 4.3.10: Step Selection Window.

Step
The Step list will show all steps that are to be used in the Post-Processor highlighted.

Selection
The selection method for the step list in the Post-Processor

All
All steps will be selected for use in the Post-Processor.

None
No steps are selected for use in the Post-Processor.

Remeshing
Remeshing will select/deselect the remeshing steps from the step list for use in the Post-Processor.

Increment
This is the previous DEFORM™ method of step selection. The increment of steps is out of the total number of steps, not just the saved steps.

Increment Saved
This will select an increment of steps, but the increment will be out of the saved steps. For example: An increment value of 5 in Increment saved mode will select every 5 saved steps between the starting and ending step for inclusion into the Post-Processor

Increment
The Increment specified here will be used in the Increment and Increment Saved step selections.

Start Step
The starting step for the increment range.

End Step
The ending step for the increment range.

Increment
The increment to be used in the range specified through the starting and ending steps.

4.3.8. View Changes within Viewport

The view of any object within a viewport can be altered using the dynamic zoom, zoon window pan and rotate buttons. These buttons all use the first mouse button and a key combination to manipulate the view.

Rotation

This can be done by simultaneously holding down the combination of left mouse button and CTRL button.

Pan

This can be done by simultaneously holding down the combination of left mouse button and SHIFT button.

Dynamic Zoom

This can be done by simultaneously holding down the combination of left mouse button and ALT button.

4.3.9. Coordinate System Selection

Object Coordinate System

The object coordinate system is displayed in the lower right corner of the data display box. When this option is selected all constrained rotation about an axis (X, Y, or Z) is applied relative to the object coordinate system. This is the default rotation method.

Screen Coordinate System

The coordinate system used for object rotation can be either the screen coordinates or the object coordinates. When the screen coordinates are selected all constrained rotation about any axis (X, Y, or Z) is in relation to the screen. This can be set by deselecting “use object coord for rotation” as seen in Figure 4.3.11

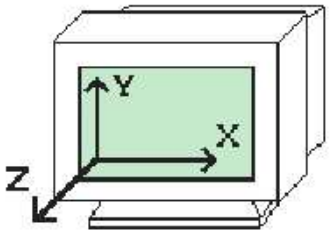


Figure 4.3.11: Screen coordinate system.

4.3.10. Rotation

Rotation can be done in either unconstrained, or constrained X, Y, or Z. Unconstrained rotation will allow rotation about all three axes, while constrained rotation will restrict rotation about either the X, Y, or Z axis. The axis for constrained rotation can either be used using the screen coordinates or the object coordinates.

When the first mouse button is pressed, a red circle appears on the screen. This represents a sphere and it's center is at the center of rotation of the object. The current rotation mode is also shown in the top left corner of the display. Moving the mouse while holding the first button rotates the part and also shows a green line, which represents the current mouse position (it represents an arc on the sphere). Once the desired rotation has been achieved, release the mouse button. Bringing the mouse back to the start of the arc will return the object to its original position. It will take some practice to get the feel for rotating the object. When dealing with an object that has a large number of elements, such as an object that has been mirrored.. The speed of rotation can become very slow. To enhance the speed, one can view a cube that represents the object while the rotation is taking place. Once the rotation is complete. The cube will be replaced with the original object. To use this option, first hold down the **SHIFT** key, then press the first mouse button and follow the procedure outlined above.

4.3.11. Coordinate Axis View

You may select any of the following views by clicking on one of the following buttons. Each button will change the view of the objects in the display window to the selected axis view. For example, clicking on the XY View button will change the display window so the XY plane is displayed in the viewport.

4.3.12. Point Selection

Any point on a graph can be selected to obtain the corresponding data values for that point.

Ruler

 This tool allows the user to measure any distance between two points by clicking consecutively on both points.

4.3.13 Multiple Viewports

Viewport Option

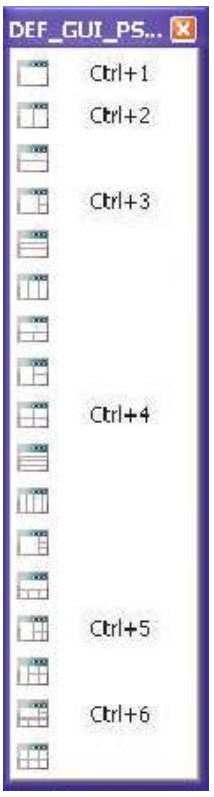


Figure 4.3.12: Viewport options.

When DEFORM™3D loads up a database, the single Viewport mode is the default mode for the display of the starting step of the database. The Viewport Option pulldown is located in the Viewport->Multi window and shows the current layout. A new mode can be selected by clicking on the desired viewport option. (see Figure 4.3.12)

By clicking the multi-link icon,  , the user can toggle whether changes in one viewport affects another viewport.

4.3.14. Nodes

The **Nodes** window will display information on the nodes of the currently selected object. This information includes **Position**, **State Variables**, and **Boundary Conditions**. A different node can be selected through the **Node** text box or by selecting a node graphically in the display window. The information will change as the step number displayed changes in the **Display** window.
(see Figure 4.3.13)

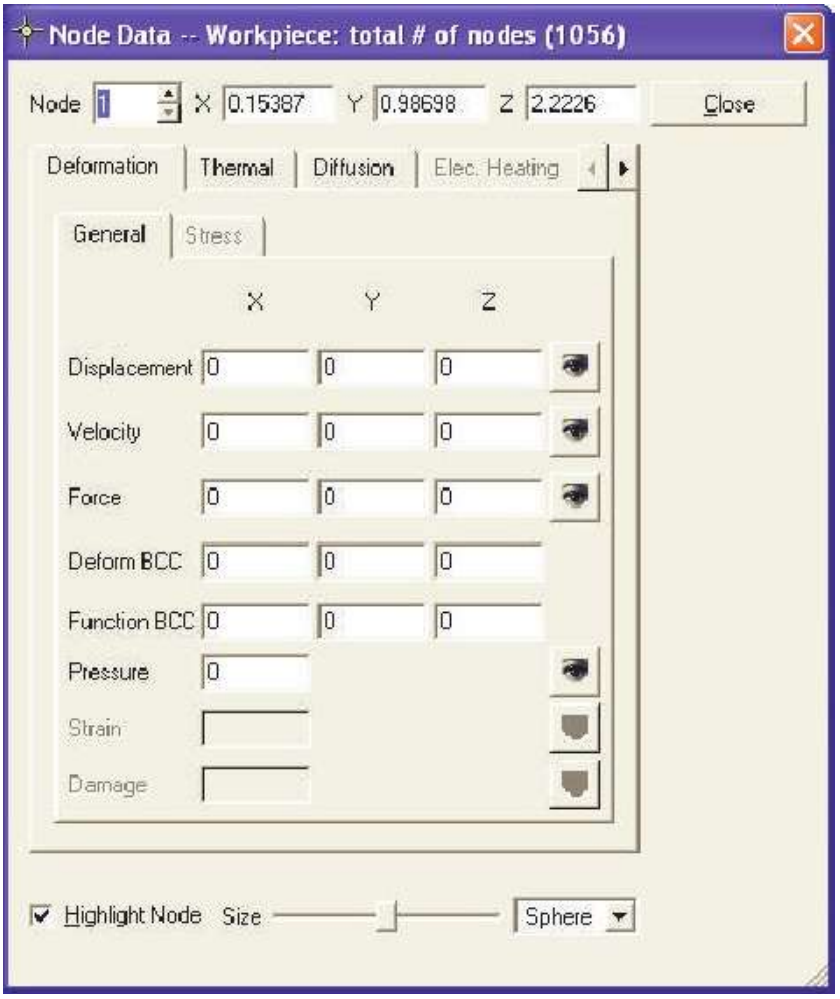


Figure 4.3.13: Nodes data window.

4.3.15. Elements

The **Elements** window will display information on the elements of the currently selected object. A different element can be selected through the **Element** text box or by selecting an element graphically in the display window. The information displayed will change as the step number displayed changes in the **Display** window. (see Figure 4.3.14)

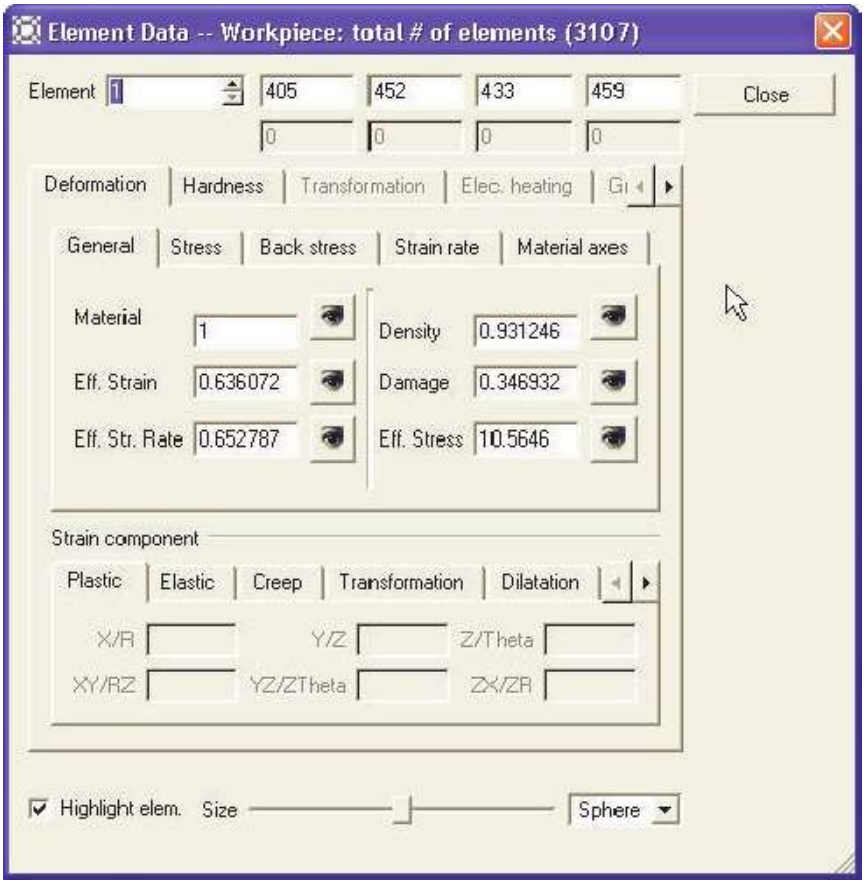


Figure 4.3.14: Elements data window.

4.3.16. Viewport

The viewport options can all be changed from within the Viewport Options window. These options will only be applied to the current Viewport in the Display Window. (Figure 4.3.15, Figure 4.3.16)

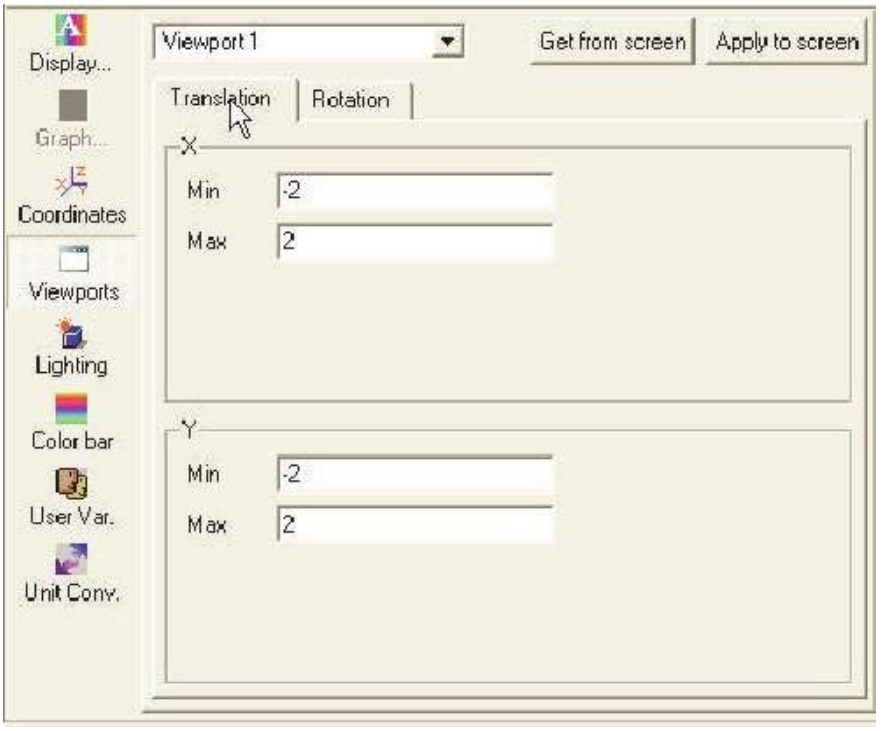


Figure 4.3.15: Viewport settings window (translation).

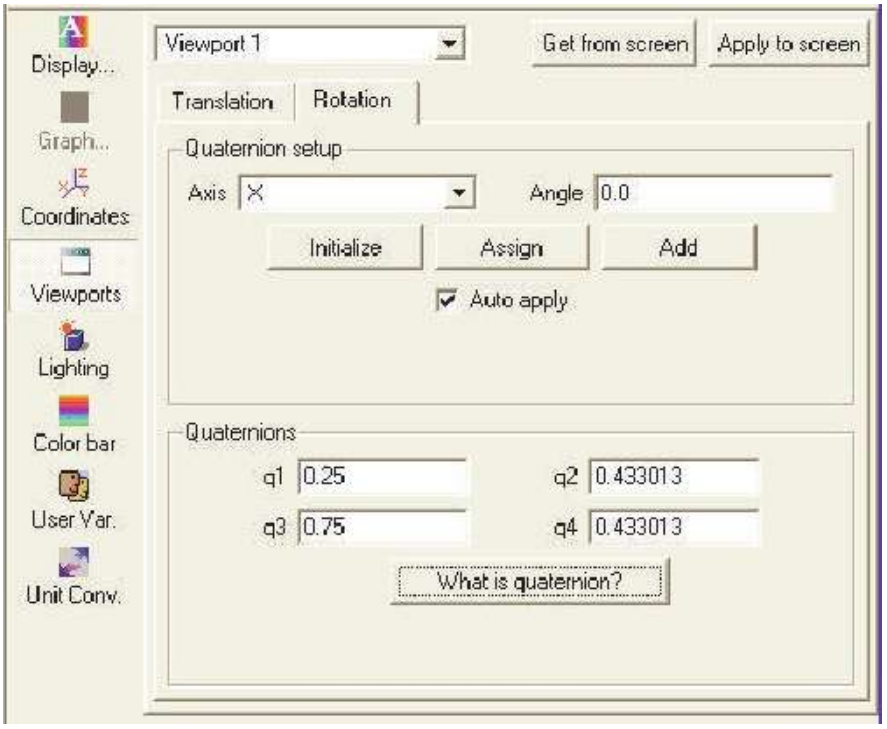


Figure 4.3.16: Viewport settings window (rotation).

4.3.17. Data Extraction

This utility in the post processor allows user to extract any model variable for a given object, at a given step in to a text file. (see Figure 4.3.17)

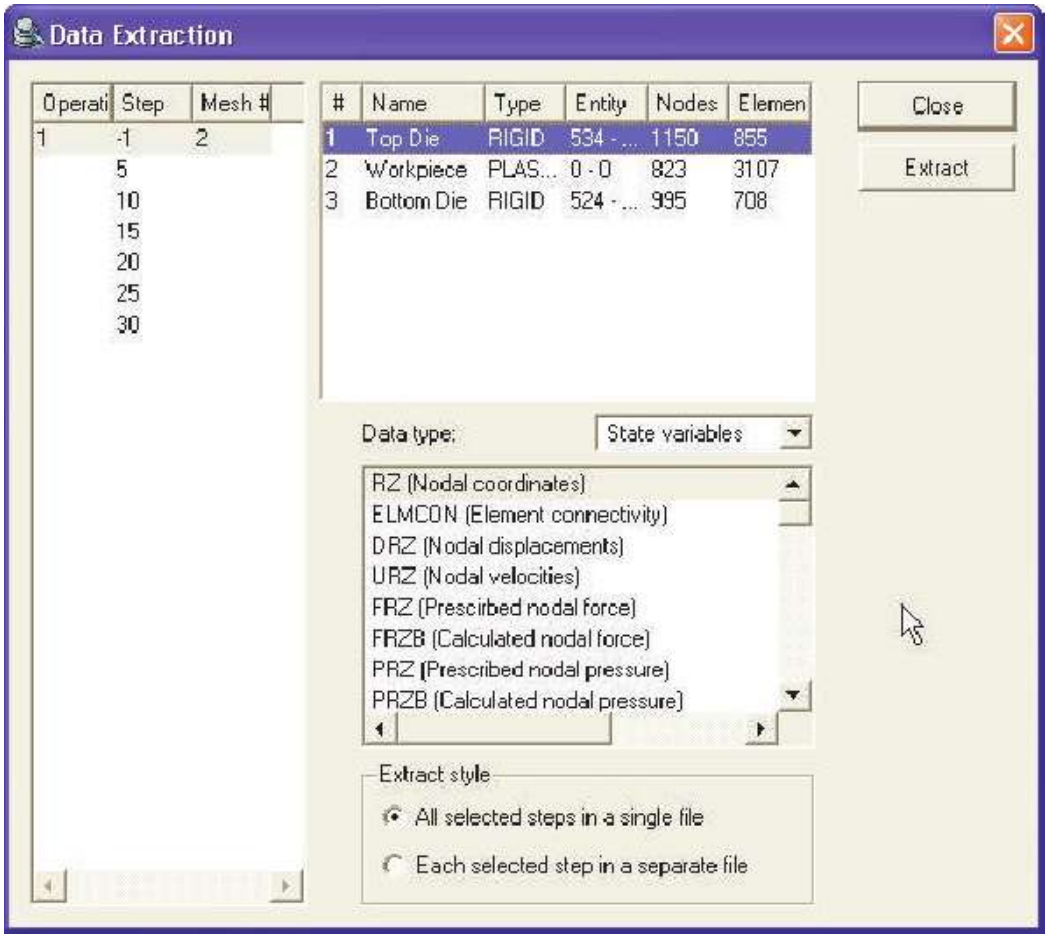


Figure 4.3.17: Data Extraction window.

Data Type
Information can be written out in a keyword format to an output file. The following items listed below can be extracted.

Object Data

Furthermore, specific keywords from Object data can be selected, as well as the objects that the data is to be extracted for.

Output
The information can be extracted to either a single file or separated into multiple files.

Steps
Single steps or all of the steps can be selected. To select a specific step, highlight the step in the step scroll down menu towards the right side of the screen. If 5 or 6 steps are desired and every step in the database is not desired. Go to the Steps option in the Post-processor select the steps desired. Go back into the data extraction window and select all for steps.

Files
The information can be extracted to one file or multiple files. A good time to implement this option, is when information for more than one step is desired. Data can be written to one file, or multiple files labeled name0001.DAT, name0002.DAT etc…

Data File
The is the name of the file, by default it is labeled DEFORM.DAT. This file can be renamed, or browse can be used to find a existing file.

Extract
Once the desired information has been selected, press the extract button to extract the information

View
By clicking on the view option, the file that is located in the Data file box can be viewed.

4.3.18. Flownet

The flownet dialog allows the user to place some form (2D or 3D) of a grid onto the object and let the simulation track the deformation of the grid throughout the deformation. This is an excellent way in which to visualize any potential irregularities in the grain structure or to view potential surface defects. The flownet dialog window is seen in Figure 4.3.18. The following actions are required sequentially to view a flownet:

1. Select the starting and ending step (See Figure 4.3.18). Click Next when finished. By omitting steps, the time to compute the flownet calculations can be reduced but the user should not omit steps if the entire process is of interest.

2. Select the grid type to be used (See Figure 4.3.19). Either 2D or 3D grids can be used. The 2D grids are less time-consuming for the same grid size since less information is required in two-dimensions. For this example, the grid selection is chosen and Next is clicked

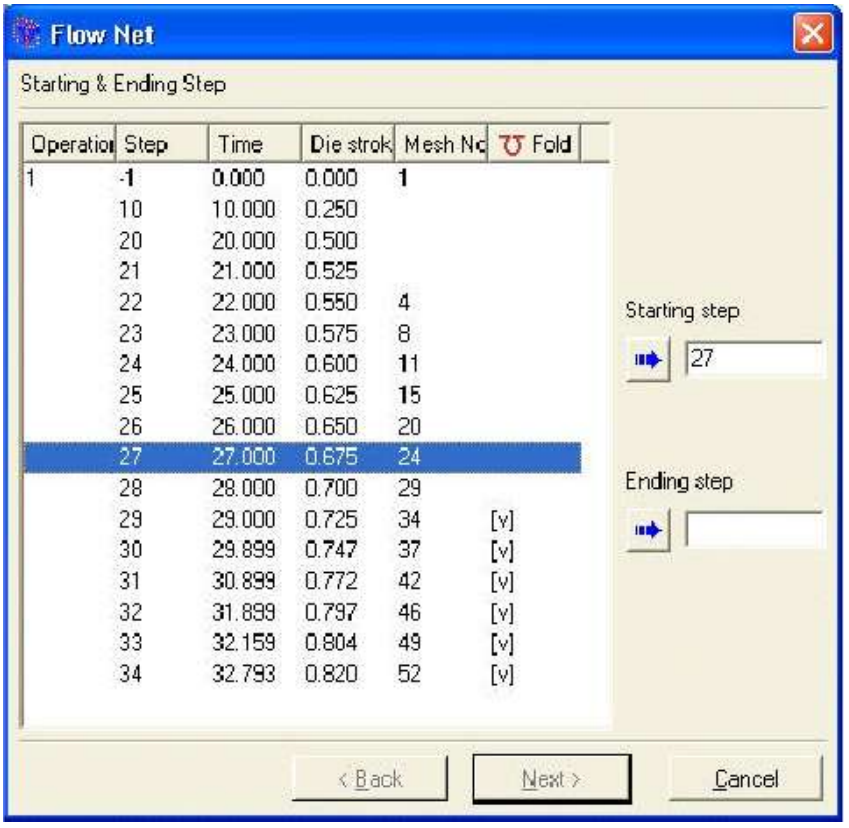


Figure 4.3.18: Step selection dialog for flownet.

3. For the 2D grid, a plane has to be defined by using a slicing plane. The plane selection method is similar to the method used by the slicing dialog. After specifying a plane to be used, click Next. In other methods, there are comparable methods for defining the regions where a flownet is valid. (see Figure 4.3.20)

4. At this time, the density of the grid needs to be specified. In the case of the grid , the number of grid can be set for a regularly spaced grid. By selecting preview the grid can be seen before calculating for all steps. After a desired grid is obtained, click Next.(see Figure 4.3.21)

5. At this time, advanced options are available such as saving either the beginning or end pattern (Figure 4.3.22). This is useful if a flownet from a different database is to be output to. Click Next when finished and the flownet will be calculated.

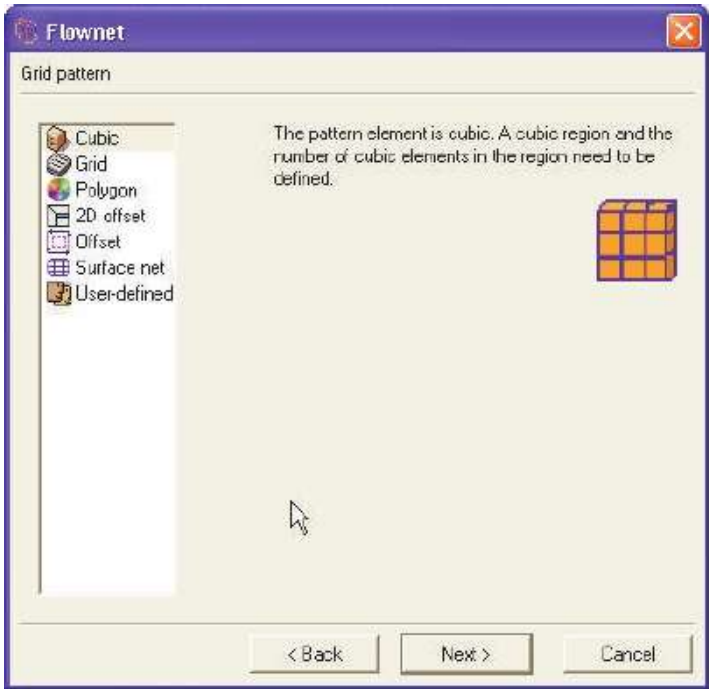


Figure 4.3.19: First window of point tracking wizard.

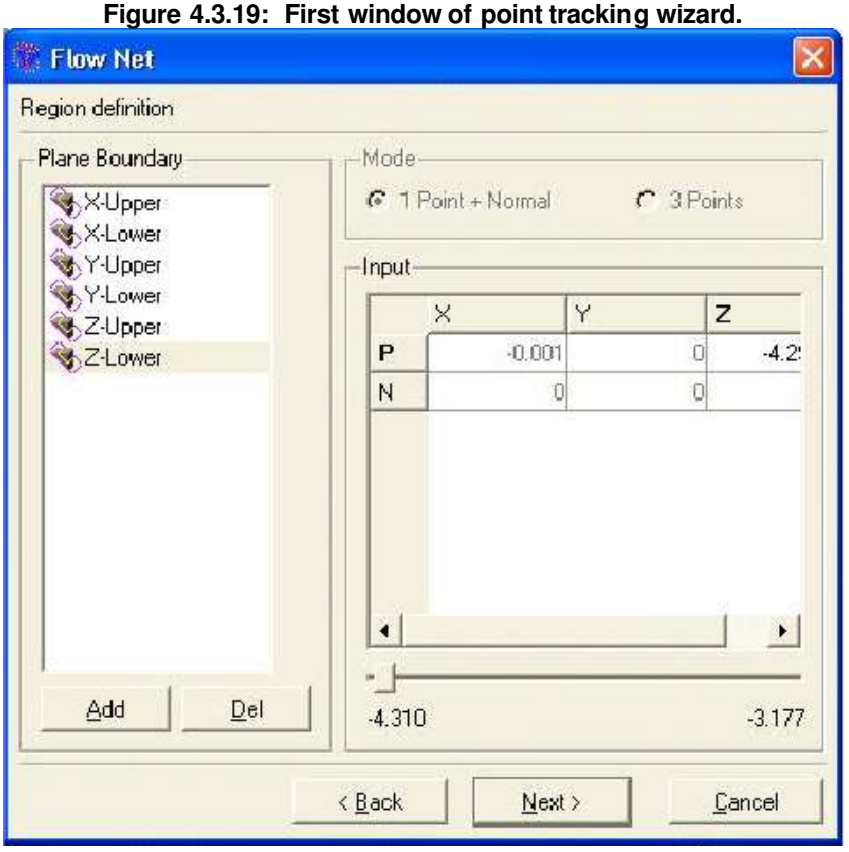


Figure 4.3.20: Region definition for flownet.

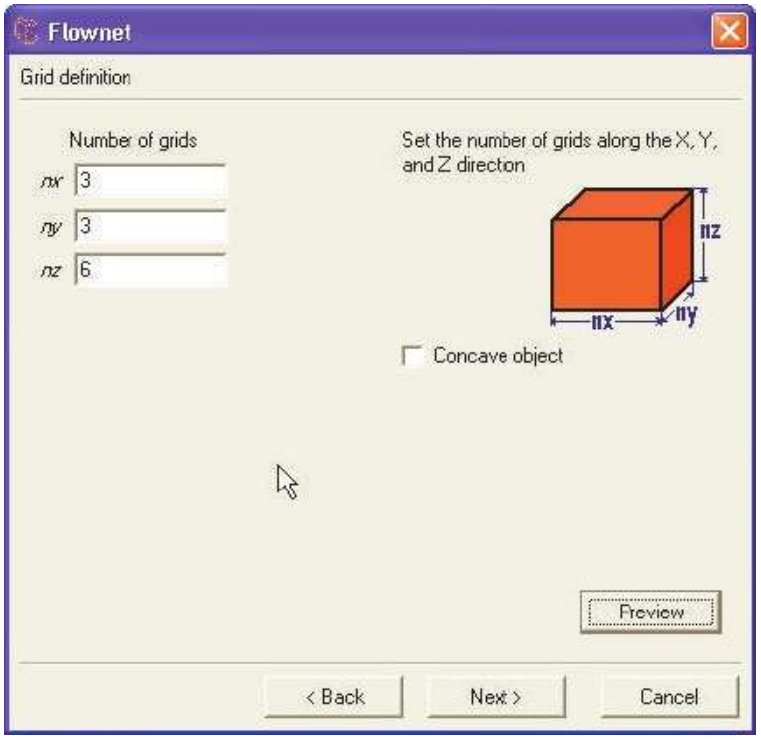


Figure 4.3.21: Grid definition for the flownet.

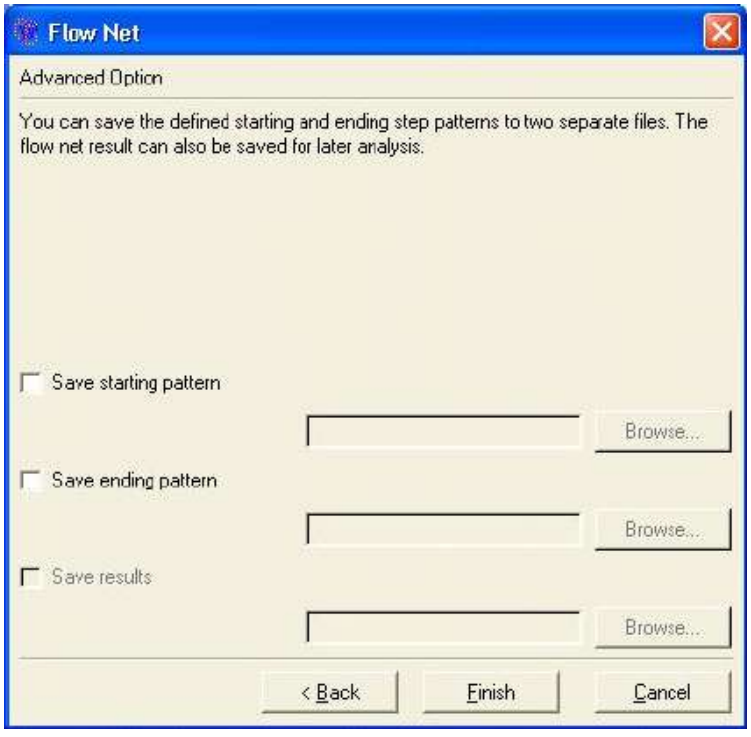


Figure 4.3.22: Advanced options of the flownet.

4.3.19. Mirroring

The mirroring dialog is seen in Figure 4.3.23. A part that has symmetry can be viewed as an entire part by selecting the Add button and clicking on the symmetry planes in the display window. To remove the mirrored section, select delete button and click on any mirrored section. Any section can be removed except for the original section. (see Figure 4.3.24)

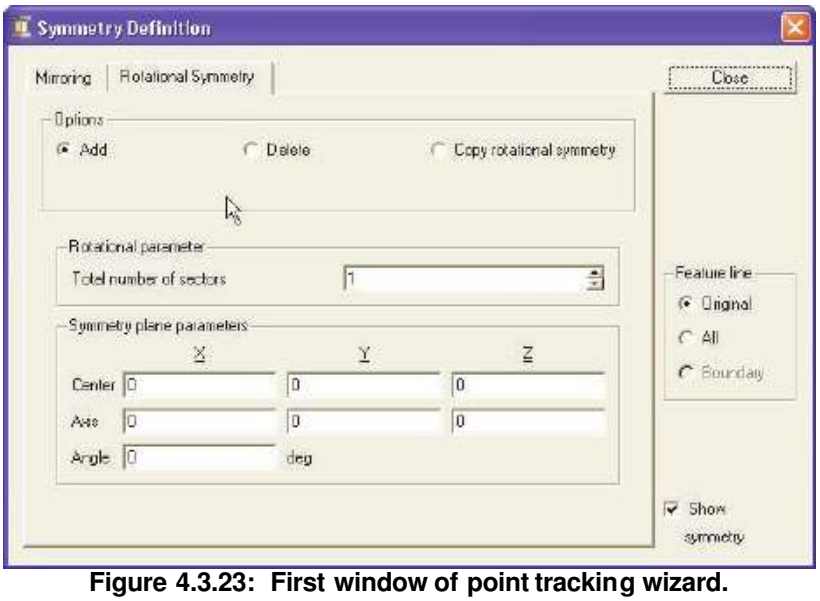


Figure 4.3.23: First window of point tracking wizard.

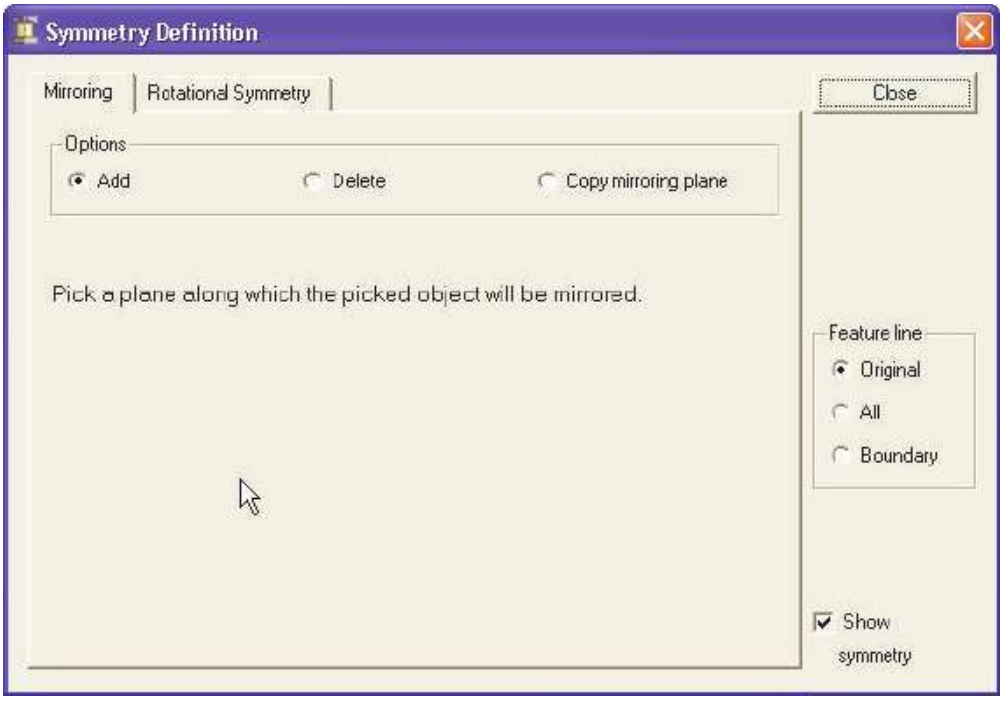


Figure 4.3.24: Mirroring dialogs.

4.3.20 Animation controls and saving.

The model results can be displayed as a continuous set of images and animation files in standard formats for presentations. These features can be accessed from the controls and settings icons . Animation controls (see Figure 4.3.25) let the users play thru the model results as a process data, and the settings dialogs let the users to save the images in a defined location (see **Figure 4.3.26**), in a specific format and resolution (see **Figure 4.3.27**). From version 3DV61, this functionality has been extended to include AVI, WMV and windows standard power point file format as well (see **Figure 4.3.27**and

Figure **4.3.28**) Please note that the MPEG-4 (compressed) AVI format will be generated correctly only
if the appropriate Microsoft MPEG-4 Video Codec is installed. This Codec is not included in the DEFORM installation. The codec can be downloaded from various Internet sites, including “http://www.afreecodec.com/win/920/microsoft-mpeg-4-v123-vki-codec/”.

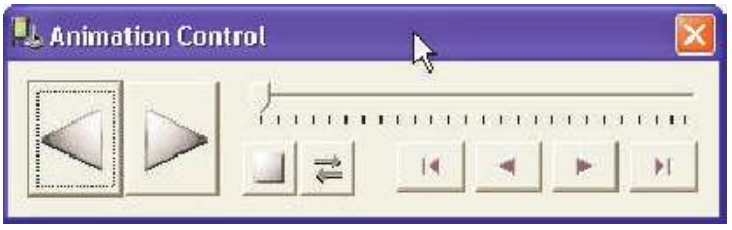


Figure 4.3.25: Animation control dialog.

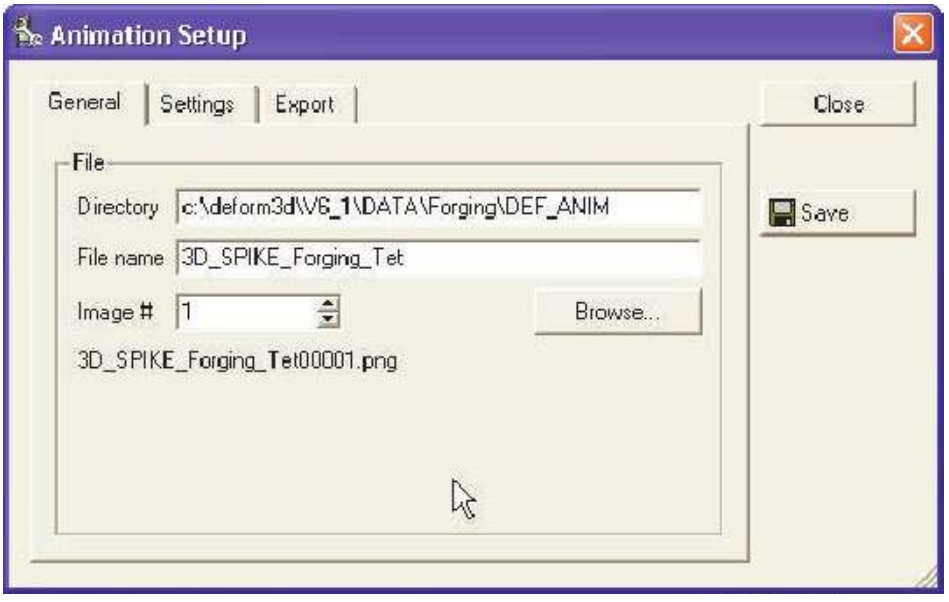


Figure 4.3.26: Animation setup dialog.

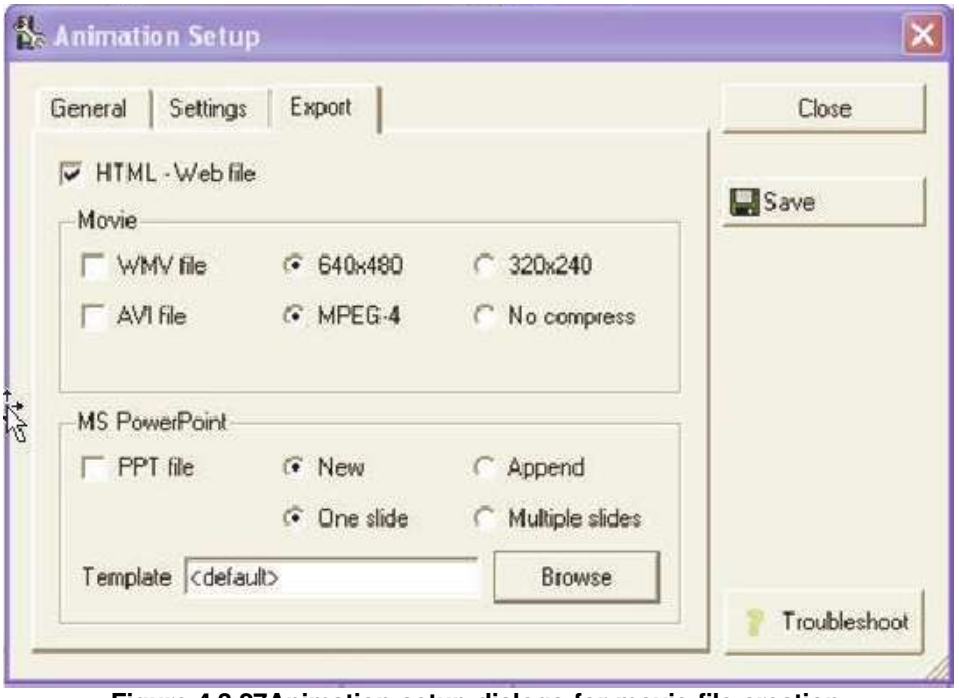


Figure 4.3.27Animation setup dialogs for movie file creation.

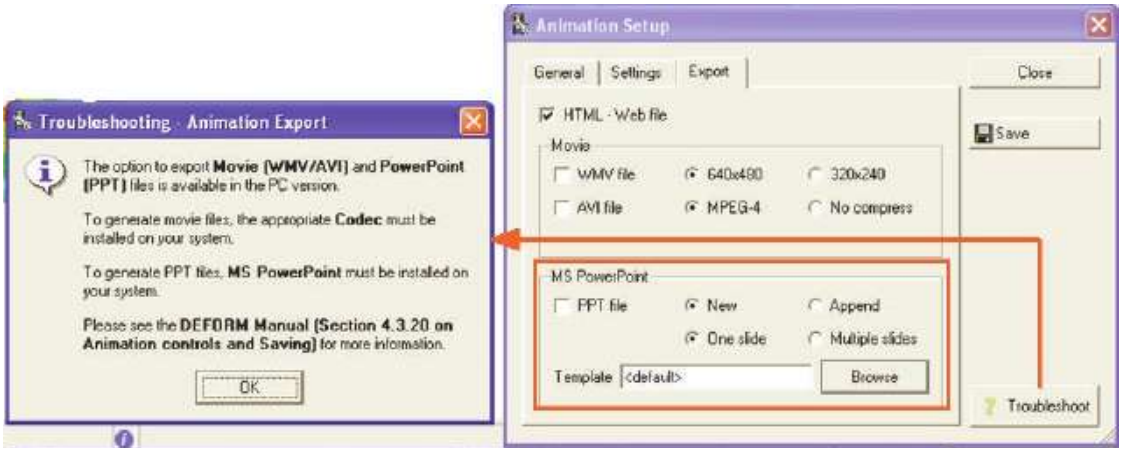


Figure 4.3.28Animation setup dialogs for PowerPoint file creation

Chapter 5: Elementary Concepts in Metalforming and Finite Element Analysis

Definition of stress, strain, and strain rate

Stress is the measure of force applied to a unit area of material. This variable is of importance in forming in that material deform (change shape) different amounts depending on how much stress they are under. There are several definitions of stress.

Engineering Stress - force per unit area measured on the original undeformed shape.

True Stress - force per unit area measured on the deformed shape. These two definitions are shown in Figure 5.1 as a comparison. In general, the true stress is more interesting for the engineer in analyzing a forming process. True stress will indicate plastic yielding and other issues with better accuracy than engineering stress.



Figure 5.1: Demonstration of concept of stress.

Strain is a measure of the total accumulated deformation a region of material has undergone.

Mathematically, the two definitions of this are seen as follows:

Engineering Strain = (Change in Length) / (Original Length)

True Strain = Sum of incremental strains = ln ((final length) / (initial length))

In Figure 5.2, the calculated strains for both an upset and a tensile test are shown. Note that the Engineering strain gives rather round numbers for doubling or halving the length of a test specimen. The advantage of true strain is that it is a more accurate measure of the actual length change in the material and is used to determine stress in DEFORM.

Strain rate is a measure of the instantaneous rate of deformation a region of material is experiencing. This quantity measures a rate of change of the strain at a point of a material per unit time.

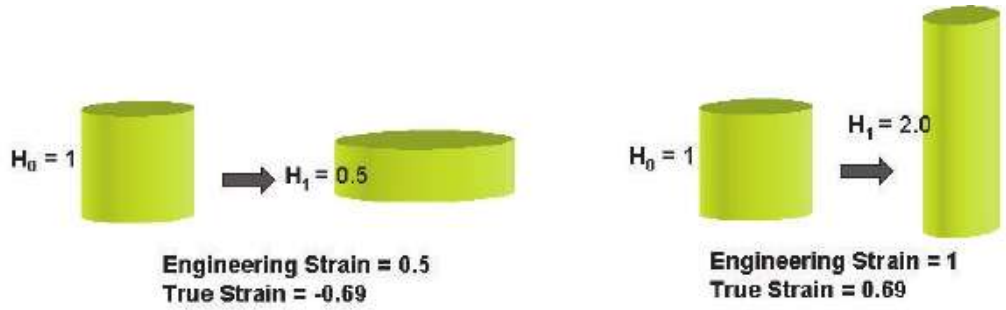


Figure 5.2: Demonstration of concept of strain.

In Figure 5.3, the engineering stress and strain are shown as closed form expressions for a compression test. Note that both of these quantities fail to account for the increase in area of the test specimen due to barrelling.

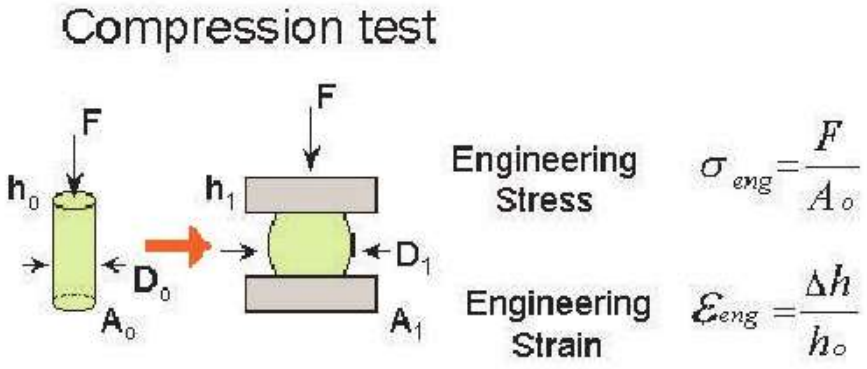


Figure 5.3: Engineering stress and strain for a compression test.

In Figure 5.4, the true stress and strain are shown as closed form expressions for a tension test. Note that both of these expressions account for the change in the cross-sectional area of the specimen during the test (assuming incompressible material). This gives a better measure of the actual state of stress and strain the material is under.

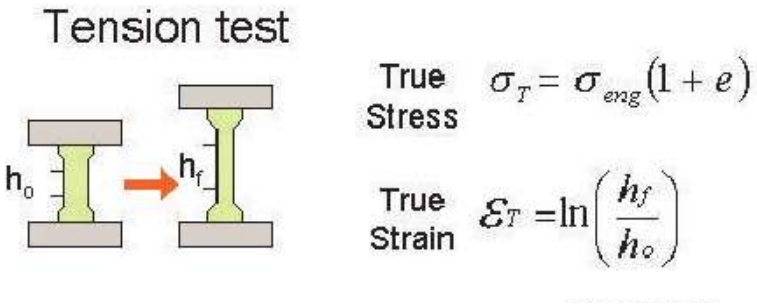


Figure 5.4: Stress and strain defined for tension test.

All materials have a characteristic stress-strain curve that determines how the material behaves structurally. For most isotropic metals, this behavior is of the general shape seen in Figure 5.5. Note that the top and bottom curves are the engineering stress-strain curves for a material and the middle curve is the true stress-strain curves for a given material. The true stress-strain curve is the same for either tension or compression, but they are not the same in terms of engineering stress-strain.

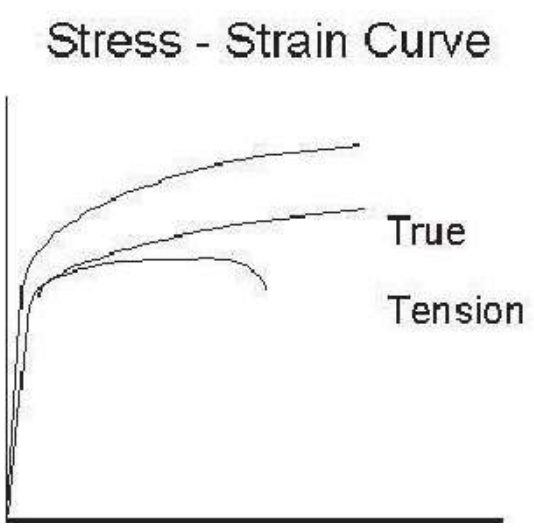


Figure 5.5: Stress-strain curve.

For simplification, the stress-strain curve is divided into two regions. The steeply sloped region at very low strain values is known as the elastic region. In the elastic region, since strains are very low, when the material is unloaded and the forces removed, the material returns to its original shape. In Figure 5.6, an object is deformed in uniaxial tension. The change in length is shown and the corresponding position on the stress-strain curve is shown. In this case, the deformation is completely elastic. After the tensile forces are removed, the material will return to the original shape.

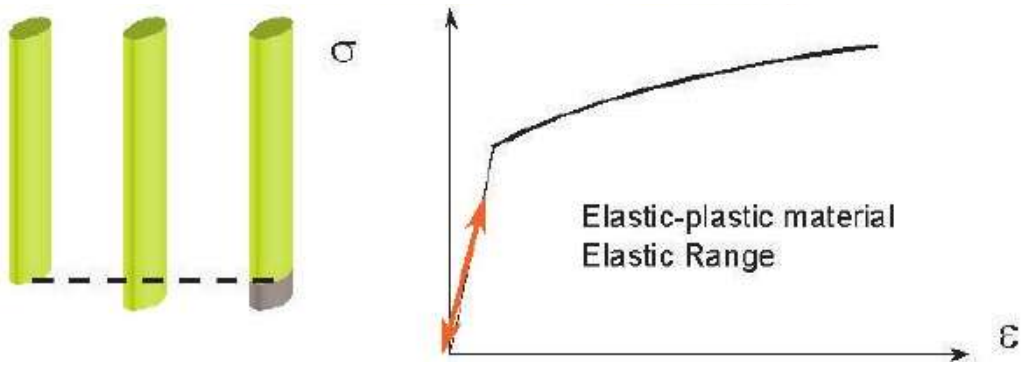


Figure 5.6: Diagram of elastic deformation.

The second region of a stress-strain curve is known as the plastic region. This region comprises strains just above the elastic range and appear on the curve as the less steeply sloped region on curve. In this region, the material does not recover any the deformation that occurs. The only recovery that occurs is the accumulated elastic strain. In Figure 5.7, a specimen under tension deforms first elastically and then plastically. The loading curve in Figure 5.7first follows the elastic loading curve and then follows the plastic curve. When the material is unloaded and the forces are removed from the specimen, the material follows the elastic curve down. When the material is completely unloaded, the deformation left over is the permanent deformation of the body.

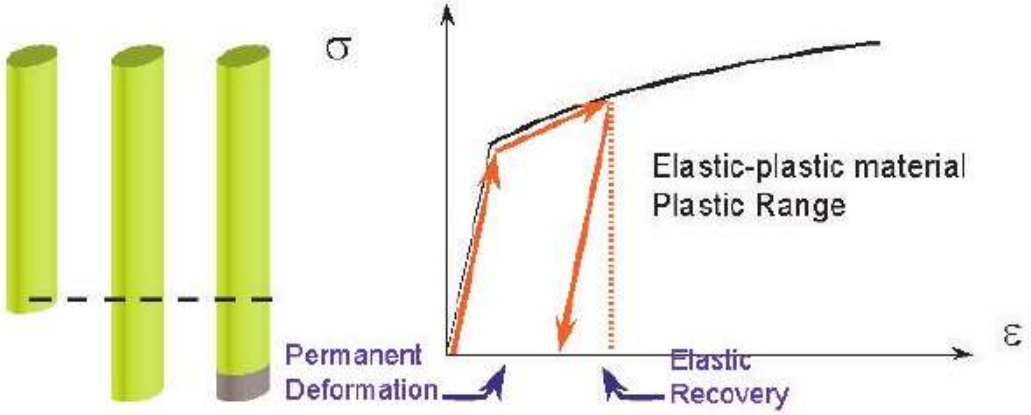


Figure 5.7: Diagram of plastic deformation.

In DEFORM, the stress and strain used in the stress-strain curves are the known as the effective stress and effective strain. The equations for these values are seen below in Figure 5.8.

Effective (von Mises) Stress $\bar{\sigma} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_z-\sigma_y)^2+(\sigma_y-\sigma_z)^2+(\sigma_z-\sigma_x)^2+6(\tau_{xy}^2+\tau_{yz}^2+\tau_{zx}^2)}$

Effective Strain $\bar{\varepsilon} = \frac{\sqrt{2}}{3}\sqrt{(\varepsilon_1-\varepsilon_2)^2+(\varepsilon_2-\varepsilon_3)^2+(\varepsilon_3-\varepsilon_1)^2}$

Figure 5.8: Equations for effective stress and strain.

In DEFORM, the concept of flow stress is used. The idea of flow stress is important in the case of incremental plasticity. As a material is deformed plastically, the amount of stress required to incur an incremental amount of deformation is given by the flow stress curve (which corresponds to the plastic region of the true stress-true strain curve. In Figure 5.9, the concept is shown visually.

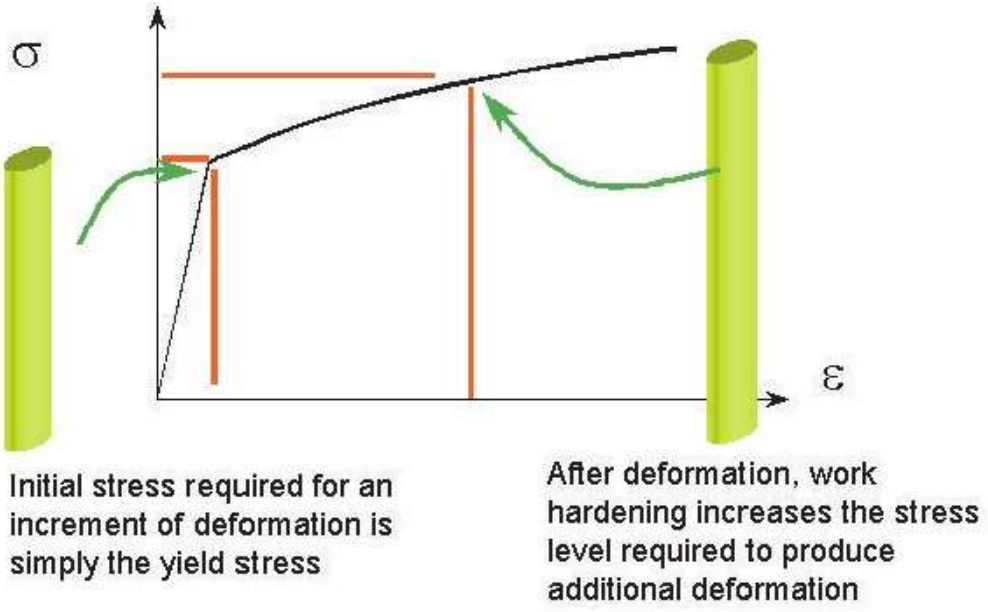


Figure 5.9: Introduction to flow stress concept.

Flow stress is strongly dependent on several state variables, among these are accumulated strain, instantaneous strain rate and current temperature. As seen in Figure 5.10, the flow stress curves can vary strongly with these state variables. So, it is important to account for these variables to accurately determine the behavior of the material.

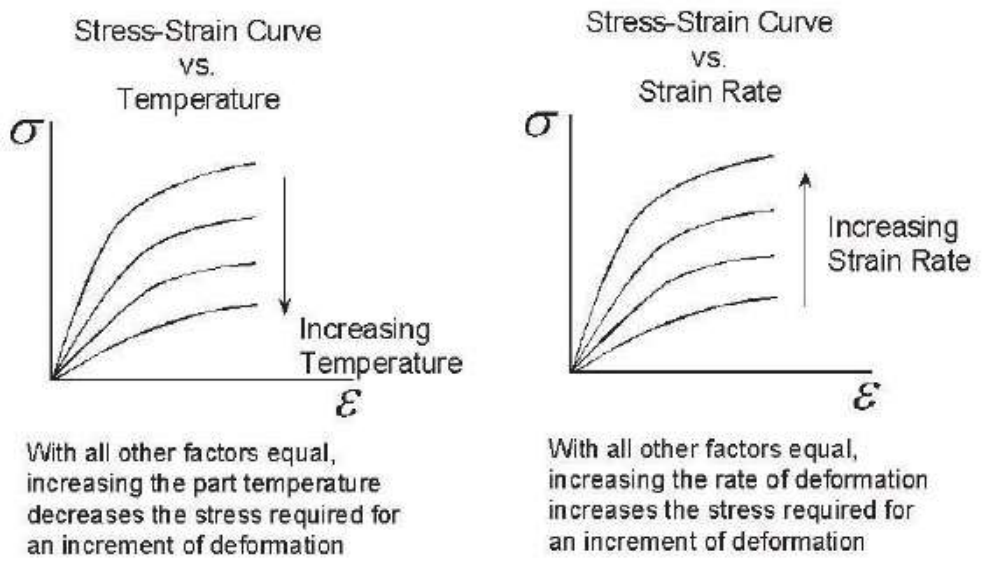


Figure 5.10: Variation of flow stress with strain rate and temperature.

In the case where elastic deformation can be neglected, as in the case of bulk metal forming, the Levy-Mises flow rule can be used to relate the stress tensor to the strain rate tensor. This flow rule is shown in Figure 5.11. The coefficient, λ, is a function of state variables and the material. This relation allows one to express stress in terms of rate of deformation.

$$\sigma_{ij} = \frac{1}{\dot{\lambda}} \dot{\varepsilon}_{ij}$$

Figure 5.11: Levy-Mises flow rule.

In calculating metal flow, the minimum work rate principle is a cornerstone for accurate calculation. This principle is defined below:

Minimum Work Rate Principle: the velocity distribution which predicts the lowest work rate is the best approximation of the actual velocity distribution.

This principle states that the material should always flow in the path of least resistance. This is shown in Figure 5.12 where there are three different upset cases and the amount of friction between the work piece and the tool determines the flow pattern of the material. In the case of no friction, there is no resistance for the material from flowing straight out uniformly. In the case of high friction, there is much resistance from flowing outward, so a barreling behavior is observed.

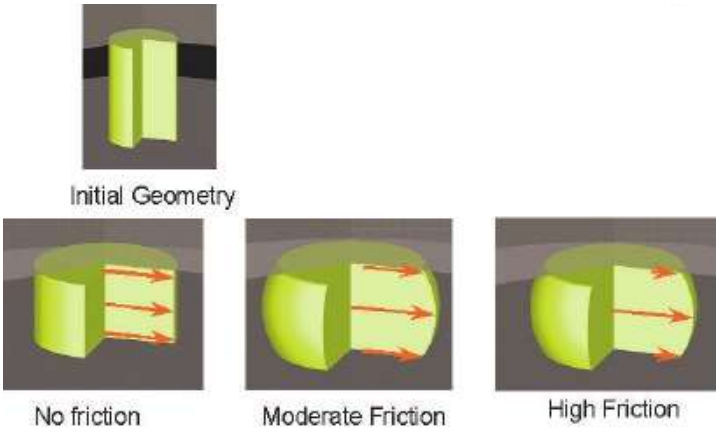


Figure 5.12: Minimum work rate principle. Note for each case, the actual velocity is the one which incurs the lowest work rate for the work piece.

This minimum work rate principle can be expressed mathematically as the following functional form seen in Figure 5.13. The top equation is simply a balance of the body forces (1st term) versus the surface tractions (2nd term). The manner in which this equation is solved for the velocities, is seen in the 2nd equation. The velocities are solved by solving for when the variation in the functional is stationary. Note that there is an extra term that maintains incompressibility in the solution. This is done by integrating the volumetric strain rate and multiplying by a large constant. Since the total solution should be zero, the solution will tend to maintain a low volumetric strain rate to keep this integral value low.

$$\pi = \int_V \bar{\sigma}\,\dot{\bar{\varepsilon}}\,dV - \int_S F_i\,u_i\,dS$$

$$\delta\pi = \int_V \bar{\sigma}\,\delta\,\dot{\bar{\varepsilon}}\,dV - \int_S F_i\,\delta u_i\,dS + K\int_V \dot{\varepsilon}_V\,\delta\dot{\varepsilon}_V\,dV = 0$$

plastic work — *surface traction work* — *change in volume (multiplied by a large "penalty" constant K)*

Figure 5.13: Functional equation for minimum work rate principle.

In order to obtain a closed form solution for complex shapes, we need to resort to mathematical tricks such as FEM. The introductory theory for this is discussed in the following section.

Introduction to FEM theory

The principle of FEM theory is divide and conquer. First, one must divide the problem into small little subproblems that are easily to formulate and after the entire problem has been divided and formulated, they must all be carefully combined and then solved. The manner in which a problem is divided is through a process called meshing. In Figure 5.14, an axisymmetric body is being upset between two flat dies. There is a grid that has been superimposed on the figure of the work piece. This grid is the mesh that represents the body being deformed. Each rectangle represents a portion of material; in this case each rectangle corresponds to a ring, for which the equation in Figure 5.13can be easily solved. Each rectangle is called an element and the intersection of any grid lines is called a node. An element corresponds to a region of material and a node corresponds to a discrete point in space.

The solution for the equations in Figure 5.13 are the velocity at each node, which are shown as vector arrows the right side of Figure 5.14. In additions to the bottom equation of Figure 5.13, there are boundary conditions that should be specified in order to provide a unique solution to the problem. In this problem, the velocity of the top set of nodes is determined by the downward speed of the die as well as the friction model between the work piece and the die. The boundary condition on the left side of the die is specified as a centerline condition meaning that the nodes are not allowed to move either right or left. The bottom nodes also have a symmetry condition meaning that they are not allowed to move up or down. These three boundary conditions allow the mesh to behave as the actual part.

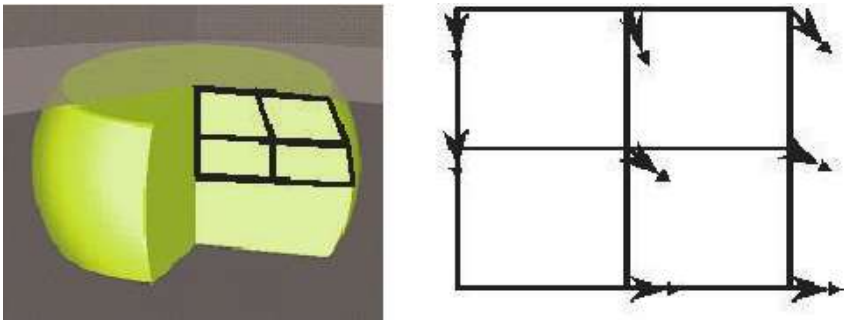


Figure 5.14: 2D mesh of upset test. (Note: This mesh is extremely coarse for the sake of clarity).

When the velocity at node points have been determined, their coordinates need to be updated. The manner in which we update the nodal coordinates is by integrating the velocity over the time step of the current step. In Figure 5.15, the nodal positions are shown as updated from the previous stage. A simple principle of note are clear from this figure:

If the nodal velocities change direction or magnitude over very small time periods, a small time step size is required to correctly predict this behavior.

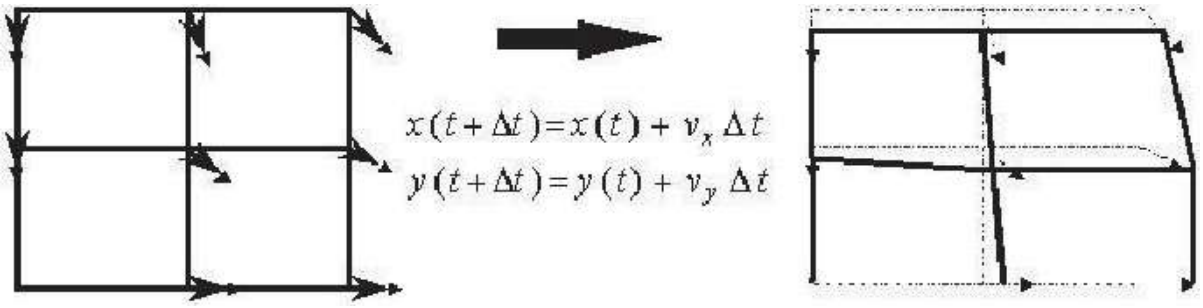
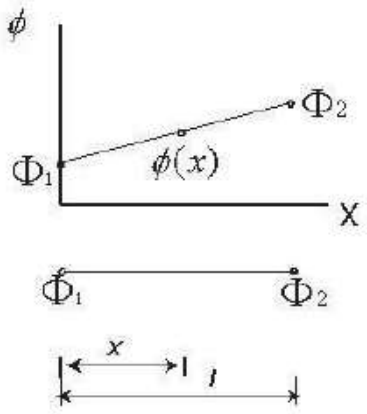


Figure 5.15: Updating nodal coordinates after a completed calculation.

The problem that now must be addressed is how to solve the equation in Figure 5.13 over a discrete set of points since the nodal values define the velocities only at discrete locations. The way in which to solve this problem is to define shape functions over the elements as a manner of providing a velocity field that satisfies the compatibility requirement (continuous over the entire body). Figure 5.16 shows a general equation for a shape function whose purpose is to define the velocity profile over an element based on the nodal velocities. A one-dimensional case is shown in Figure 5.16 as a simple linear function. The advantage of an equation of this form is that compatibility is maintained when the same nodes for any shared element edge, define the velocity over that edge.



Generalized shape function equation:

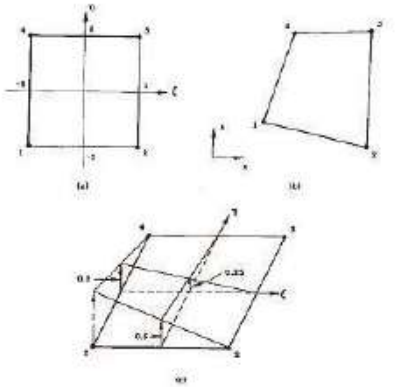
$$\phi = \sum N_i \Phi_i$$

Figure 5.16: Description of a shape function.

Figure 5.17 also shows the case of a two-dimensional element.

$$\phi(x) = N_1\Phi_1 + N_2\Phi_2$$

$$\phi(x) = \frac{x}{l}\Phi_1 + \frac{l-x}{l}\Phi_2$$



REF: fig. 6.5, Pg. 97 of Kobayashi, Oh, and Altan's *Metal Forming and Finite Element Method*

Figure 5.17: Description of shape function.

After all the equations for the elements have been written out, they must be combined into a single set of simultaneous equations. This process is shown in Figure 5.18. At the end, using a Newton-Raphson iteration method, the updated velocity can be solved for by solving a simultaneous set of equations. Once this velocity update is solved for, it is applied to the current velocity and velocity update is solved again.

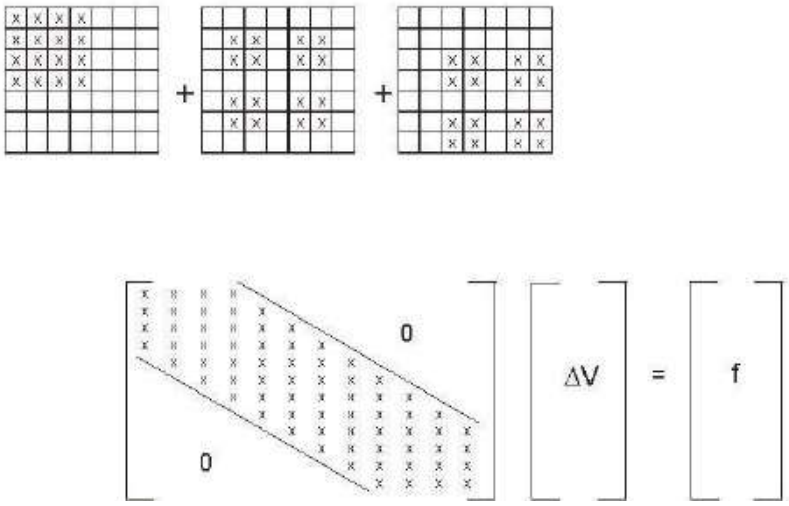


Figure 5.18: Construction of a stiffness matrix to solve for velocities.

The general FEM solution process is given below:

1. Input Geometry & Processing Conditions
2. Generate Initial Guess of Velocity Field single step:
3. Calculate Element Behavior Based on Velocity Field & other variables (strain, temp, etc)
4. Calculate Force boundary conditions based on Velocity Field
5. Assemble and solve the matrix equation
6. Calculate the error
7. If error is too large, apply correction to velocity field and go to step 3. otherwise, continue to step 8.
8. Update Geometry
9. Calculate temperature change for this step
10. Calculate new press velocity if necessary
11. If stopping criteria has been reached, END.otherwise, go to step 3 and repeat the process

This concludes this short summary of the FEM process in metalforming. For more information on FEM theory, please consider reading the following list of references:

1. Kobayashi, S., Oh, S.I. and Altan T. *Metalforming and the Finite-Element Method*. Oxford University Press. 1989.

2. Przemieniecki, J.S. *Theory of Matrix Structural Analysi*s. Dover Publications. 1968.

3. Zienkiewicz, O.C. and Taylor, R.L. *The Finite Element Method*. McGraw-Hill. 1989.

Chapter 6: User Routines

This chapter explains the various user routines available in the DEFORM system for both the FEM engine and the post-processor. Examples on how to use each type of routine, how to compile the code, and how to run the modified FEM engine and post-processor are also covered.

The FEM engine user defined routines can be used for many different purposes during a simulation. Currently user routines exist for flow stress definition, movement control, calculation of user nodal values (USRNOD), calculation of user element values (USRELM), damage models and many other specialized needs. In the post-processor, user defined post-processing routines can be used to calculate field variables using the steps stored in the database. To implement the user routines you must have a FORTRAN compiler installed on your system or the user may compile the routines on the SFTC website (Windows only).

User-Defined FEM Routines

User-Defined FEM Routines are FORTRAN subroutines in which the user can change internal routines within the DEFORM FEM engine to achieve very specialized functions within DEFORM. These subroutines can then be compiled and linked to provide object code to generate a custom built FEM engine. The user subroutines are grouped in to different fortran source files based on their functionality. These are text files containing all the available FORTRAN subroutines including all the common blocks with all the variables explained in comments. To compile this file, run the script file DEF_INS.COM (on unix), select the routine (fem or user defined post) you would like build, and select the platform (like hp, dec, linux etc..) for which you would like to build for. At this point, the FORTRAN files will be compiled and liked to the object code named DEF_SIM.OBJ (on unix). This will then generate a new FEM engine, named DEF_SIM.EXE. (and the corresponding parallel versions P4and P4P as well). As shown in Figure 6.1a similar structure has been provided for PC environment as well. This whole process is shown in Figure 6.1b for unix and PC environments.

Currently user routines exist for flow stress definition, movement control, calculation of two nodal values (USRNOD), calculation of element values (USRELM), and for other models. For example, there are many different methods for a user to control the movement of a rigid body within DEFORM, e.g. constant velocity, mechanical press, hammer press movement, speed as a function of time. However, there are some cases where a slightly more specialized movement control is required, such as movement based on variation of state variables of the work piece. This can be performed using user-routines since these variables are available when the movement of the rigid die is calculated.

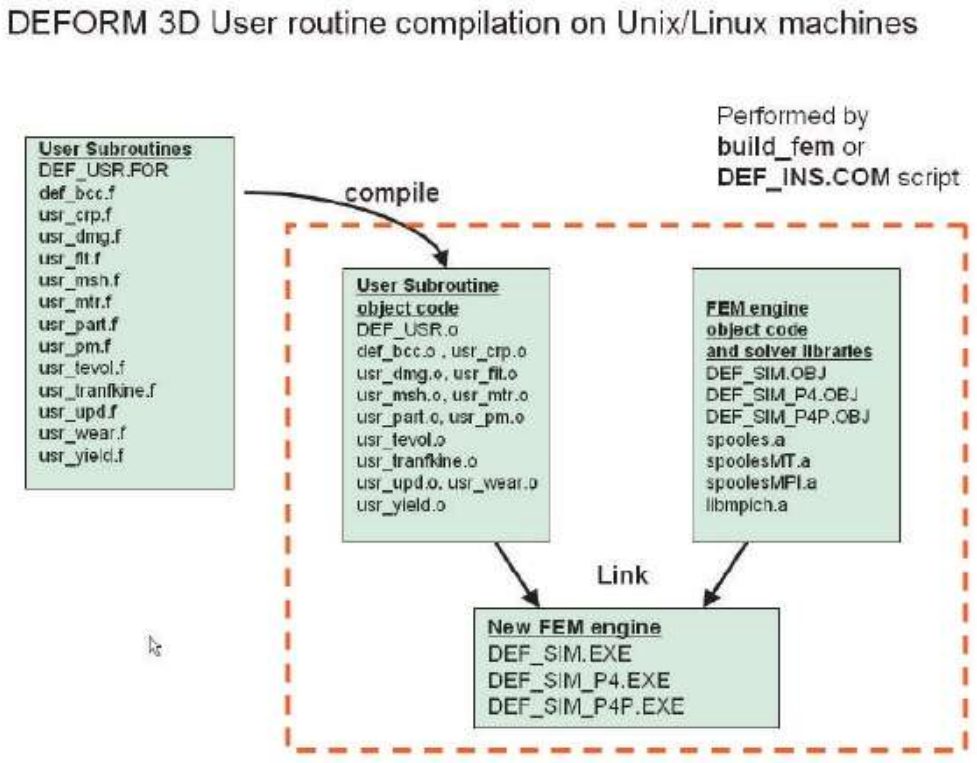


Figure 6.1: Description on how to compile/link a new FEM engine on UNIX

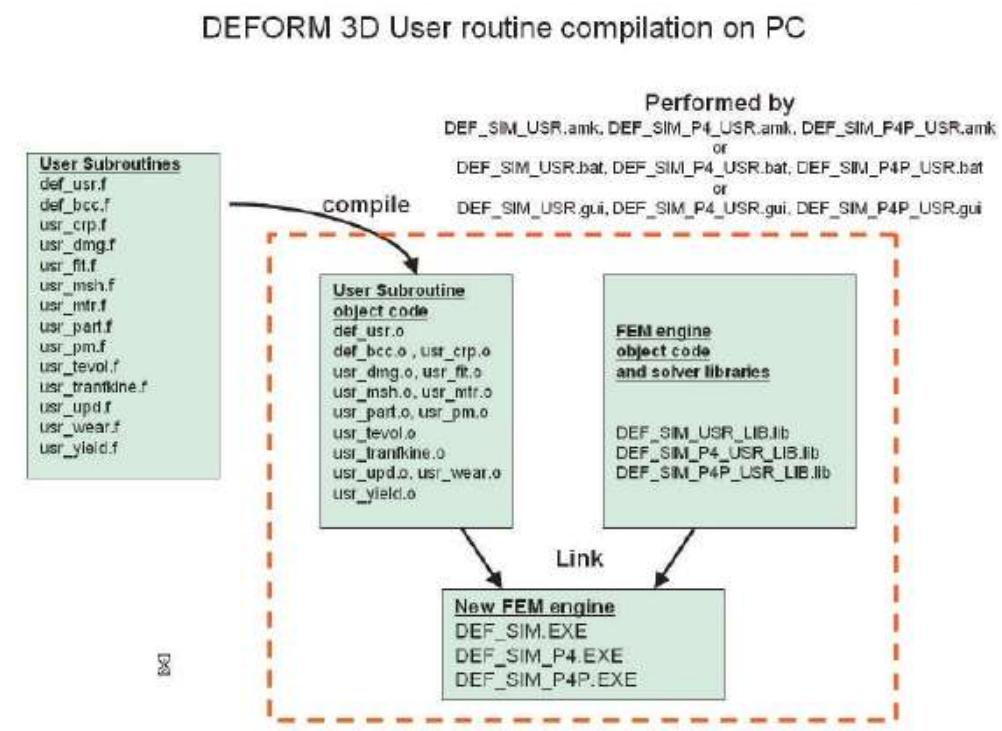


Figure 6.1b: Description on how to compile/link a new FEM engine on PC

Options for Linux Kernels

The supported kernel are (use the 'uname -a' to get these details)
LINUX_2.4.20-28.7 (old redhat version) using Absoft v7.5
LINUX_2.6.8-24.25 (Suse92 version, can be used on Suse10.1, 10.2 as well) using Absoft v9.0
LINUX_2.6.9-55.0.2.ELSMP (Centos/Redhat Enterprise Linux 4) using Absoft v10.0

The following can be used as guidelines to select the options for Linux operating system when prompted by DEF_INS.COM script

(three options available are 'linux', 'absoft90_linux', 'centos_linux' respectively for the above mentioned kernels)

If user is using any other variations of the kernels, support is subjected to delay and can not be guaranteed. If user is evaluating kernels different from the above list, please note that compatibility can become an issue. And the following can be general guidelines, in selecting the compatible kernel numbers for use with Deform user routine compilation.

kernel number starts with 2.4.20 it needs to be 2.4.20-28+
kernel number starts with 2.6 is needs to be 2.6.8-24+(Suse92+) or 2.6.9-55 + (Centos 4)

Summary of subroutines and calling structure of user-defined FEM routines

Here is a list of the different subroutines available to the user. With each, routine is a brief description of its purpose and the frequency of it being used by DEFORM.

1. USRRAT
Description: This routine allows the user to define a routine to calculate incubation time and change of volume. This routine is called in the transformation algorithm. This routine is called after each converged step.

2. INCUBT
Description: This routine is added for convenience to complement USRRAT. This routine is called whenever INCUBT is called. (available in ***usr_tranfkine.f*** file)

3. USRMTR
Description: This routine allows the user to calculate flow stress of a material. This routine is called at the beginning of each iteration. (available in ***usr_mtr.f*** file)

4. UFLOW#
Description: This routine is one of the many flow stress calculation routines. This routine allows the user to store many different flow stress routines in the DEF_USR.FOR and specify which routine is called in the Pre-Processor (available in ***usr_mtr.f*** file). This routine is called each time USRMTR is called.

5. USRDSP
Description: This routine allows the user to calculate the die speed of a rigid object that has movement defined as a user model. This routine is called at the beginning of each step. (available in ***usr_dsp.f*** file)

6. DIESP#
Description: This routine is called by USRDSP as a means of segregating movement routines in the same manner as the flow stress functions. This routine is called whenever USRDSP is called. (available in ***usr_dsp.f*** file)

7. USPM
Description: This subroutine allows the user to specify parameters for densification of a porous material model. This routine is called before stiffness matrix generation which means it's called at the beginning of each iteration. (available in ***usr_pm.f*** file)
8. USRUPD
Description: This is the user defined nodal and element variable subroutine. This routine allows the user to calculate special state variables and store them for each node and element. These variables can be viewed in the Post-Processor or be used during the simulation in flow stress calculation. User Nodal variables are updated only at the end of a converged step. User Element variables are updated at the beginning of each iteration and at the end of a converged step. The purpose for this is that the most current user element variable can be fed into the user-defined flow stress routine. (available in ***usr_upd.f*** file)

9. USRCRP
Description: This routine is used to define creep rate and its derivative as a routine. This routine is available for only elasto-plastic materials. This is called upon the beginning of each iteration. (available in ***usr_crp.f*** file)

10. USRMSH
Description: This routine is used as a general purpose routine that has access to many internal variables within DEFORM. This routine is advocated when other routines cannot satisfy the needs of the user. This routine is called at the beginning and end of each step. (available in ***usr_msh.f*** file)

11. USRDMG
Description: This routine allows the user to define a special damage model as a FORTRAN routine. This routine is called at the end of each step once per element. (available in ***usr_dmg.f*** file)

12. USERWEAR

Description: This routine allows the user to define a special wear model as a FORTRAN routine. This routine is called at the end of each converged step once per element. (available in ***usr_wear.f*** file)

13. USR_TRNF_KINE
Description: This routine allows the user to define a transformation as a FORTRAN routine. This routine is called at the end of each converged step once per element. (available in ***usr_tranfkine.f*** file)

User-Defined Post-Processing Routines

In the post-processor, user defined post-processing routines can be used to calculate field variables using the steps stored in the database. The manner in which these values are computed is based on creating a shared library file to compute variables based on the variables stored in the database. To implement user-defined post-processing variables,

The file in the USR (UNIX) or UserRoutine (Windows) subdirectory of DEFORM-3D should be copied to a local directory and the file PSTUSR.FOR or pstusr3.f can be edited. The manner in which to edit this file is discussed in a later section of this document.

After editing, this file can be compiled and linked as a shared object file (UNIX/LINUX) or a dynamically-linked library (Windows).

This shared library can be called by going to the User Variable tracking window seen below. To select the library, click on the library tab and select the library that was built. After this, go back to the tracking window and press the Track Data button.

After the data has been tracked, go to the State Variables window and select the USR tab and then the state variables can be plotted to User Variables.

Note: User-defined Post-Processor routines can only use data that was saved in the database. If a very coarse step increment was made, variables having a cumulative effect can have considerable error.

6.1. User defined FEM routines

This section contains a description of the different FEM user routines available in the current release of DEFORM-3D. The skeletal code for user routines is stored

in different fortran files (see Figure 6.1) which has the FORTRAN functions that the FEM engine calls if a user routine is to be used. The user routines calculates the specified values and returns output values

User defined data (USRDEF)
The user defined data (USRDEF) field in the pre-processor can be used to stored data that can be used to specify parameters for the user-routines. This data can be defined in the *Simulation Controls, Advanced Controls* menu as shown in Figure 6.2. In the user-routines the following code lets the user access the USRDEF values common block through the variable IUSRVL. This data can be accessed from any type of user routines. This data is defined for a given model, not specific to an object or object type.

```
      CHARACTER*80 IUSRVL
      COMMON /IUSR/ IUSRVL(10)
```

To read and write data to the USRDEF variable the following sections of code can be used.

```
C     C     TO READ DATA (10 RESERVED LINES)
C
      READ(IUSRVL(LINE NUMBER),*) DATA1,DATA2,DATA3...
C
C     TO WRITE DATA (10 RESERVED LINES)
C
      WRITE(IUSRVL(LINE NUMBER),*) NEWDATA1, NEWDATA2, NEWDATA3 ...
```

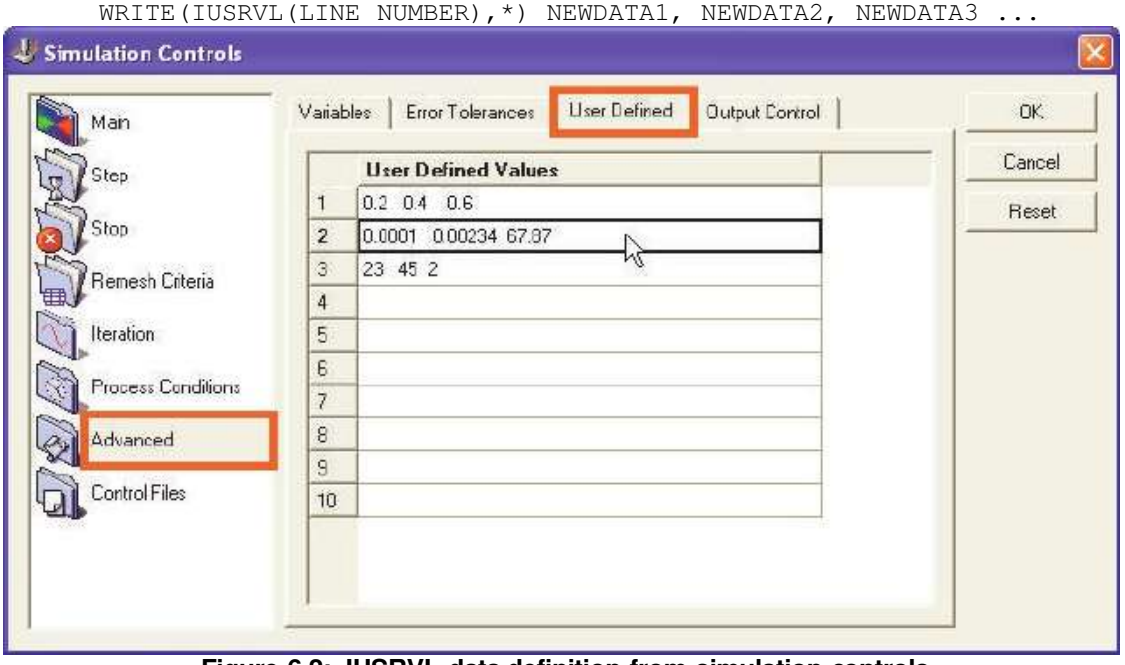


Figure 6.2: IUSRVL data definition from simulation controls

User defined flow stress routines (USRMTR)
If the flow stress models in DEFORM are not applicable for a process, a user

defined flow stress can be calculated during the simulation. The flow stress can be a function of strain, strain rate, temperature, user node and user element variables. The flow stress subroutine should return the following information :

```
   YS         =   FLOW STRESS
   YPS        =   DERIVATIVE OF FLOW STRESS W.R.T. TEPS
   FIP        =   DERIVATIVE OF FLOW STRESS W.R.T. EFEPS

where

   TEPS       =   EFFECTIVE STRAIN
   EFEPS      =   EFFECTIVE STRAIN RATE
```

A maximum of 100 flow stress routines can be defined in this program. In the pre-processor *Material Properties* the flow stress (FSTRES) type selected should be *Advanced* and the routine number to be used should be specified for each material group which uses the user routine. (see Figure 6.3a and Figure 6.3b) This routine number (NPTRTN)is passed to the user defined flow stress subroutine **USRMTR** to control branching to the specified **UFLOW** module.

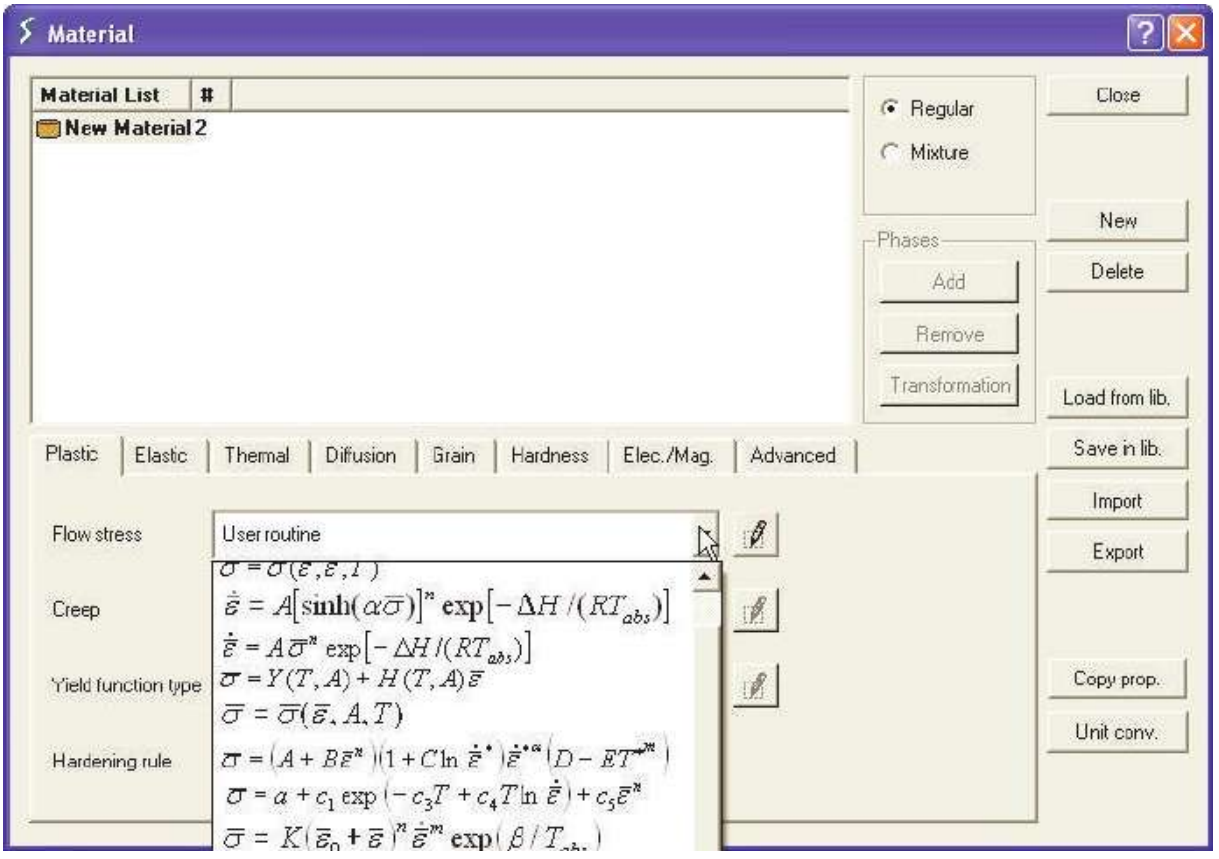


Figure 6.3a: Defining user defined flow stress routine information in Preprocessor

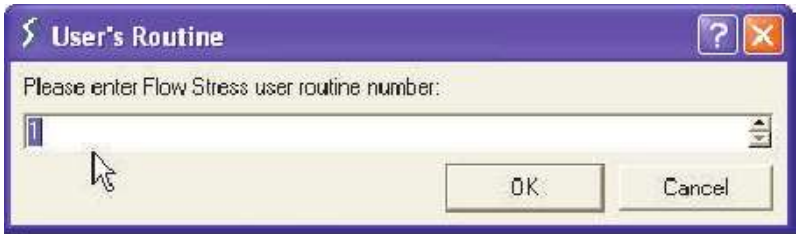


Figure 6.3b: Defining user defined flow stress routine number in Preprocessor

Examples of using the user defined flow stress subroutine are given below :

1. The flow stress depends on the strain rate sensitivity index (PEM) and on the effective strain rate (EFEPS).

```
PEM = 0.1
 YS  = 10. * (EFEPS)**PEM
 FIP = 10. * PEM * (EFEPS)**(PEM-1.)
 YPS = 0.
```

2. The flow stress depends on the strain index (PEN), strain rate sensitivity index (PEM), the effective strain (STRAIN) and the effective strain rate (EFEPS). The value of effective strain can be the element strain or from a user defined state variable. In the example given below the effective strain comes from a user defined state variable which stores the current strain. This example also illustrates the concept of using the user defined state variables to calculate flow stress.

```
STRAIN = USRE1(1)
  IF (STRAIN.LE.0.) STRAIN = 1.E-5
  PEN = 0.15
  PEM = 0.1
  YS  = 10. * STRAIN**PEN* (EFEPS)**PEM
  FIP = 10. * STRAIN**PEN* PEM * (EFEPS)**(PEM-1.)
  YPS = 10. * PEN * STRAIN**(PEN-1.) * (EFEPS)**PEM
```

The UFLOW routine is called 5 or 9 times per iteration for each element (tet or brick, respectively). The calling sequence for a single element is:

1. Guess the velocity of each node in the element
2. At each integation point (4 or 8 for tet or brick) calculate the strain rate.
3. Evaluate the flow stress at each integration point using the following values
   - Strain rate at integration point
   - Temperature at integration point at the beginning of the step
   - Strain = Strain at beginning of step + (Strain Rate * Time Step)
4. Evaluate the flow stress at the center of the element using the following values
   - Strain rate at the center of the element

- Temperature at the center of the element at the beginning of the step
- Strain = Strain at the beginning of the step

This sequence is repeated for each element. The stiffness matrix is generated and solved using these values. The solution yields a velocity correction vector, (which gives the velocity error norm) and the difference between internal and boundary forces (which gives the force error norm). When these two values have converged, the step data is written to the database, and temperature and microstructure calculations are performed, then the process is repeated from step 1.

User has access to a range of nodal and elemental data in addition to user defined variables. Comments provided in the routines explains all the variables and their meaning. Some of these are indicated here.

```
C    INPUT :
C
C      NPTRTN    =   FLOW STRESS NUMBER
C      TEPS      =   EFFECTIVE STRAIN
C      EFEPS     =   EFFECTIVE STRAIN RATE
C      TEMP      =   TEMPERATURE
C      ALSO VARIABLES IN /ELMCOM/
C
C    OUTPUT :
C
C      YS        =   FLOW STRESS
C      YPS       =   DERIVATIVE OF FLOW STRESS W.R.T. TEPS
C      FIP       =   DERIVATIVE OF FLOW STRESS W.R.T. EFEPS
C
      COMMON /USRCTL/ DTK,KOBJ,ISTATUS,KSTEP
C
C     COMMON /USRCTL/
C     DTK    : TIME INCREMENT
C     KOBJ   : OBJECT NUMBER
C     KSTEP  : Step Number (N)
C     ISTATUS: 0 - the begain of the step
C              1 - the end of the step
C
      COMMON /ELMCOM/ RZE(3,8),URZE(3,8),STSE(6),EPSE(6),EFEPSE,EFSTSE,
     +               TEPSE,RDTYE,TEMPE(8),USRE1(100),USRE2(100),
     +               DTMPE(8),NODEE(8),KELE,KONP
C
C     COMMON /ELMCOM/
C     RZE    : NODAL POINT COORDINATES (four corner nodes)
C     URZE   : NODAL POINT VELOCITY    (four corner nodes)
C     STSE   : STRESS TENSOR
C     EPSE   : STRAIN RATE TENSOR
C     EFEPSE : EFFECTIVE STRAIN RATE
C     EFSTSE : EFFECTIVE STRESS
C     TEPSE  : TOTAL EFFECTIVE STRAIN
C     TEMPE  : FOUR NODAL TEMPERATURE
C     RDTYE  : RELATIVE DENSITY
```

```
C     USRD1  : USER DEFINED STATE VARIABLES (INPUT: AT the Beginning of STEP
N)
C     USRD2  : USER DEFINED STATE VARIABLES (OUTPUT: At the End of the STEP
N)
C     NODEE  : CONNECTIVITY OF THE ELEMENT
C     KELE   : ELEMENT NUMBER
C     KONP   : NODE NUMBER PER ELEMENT
C
C     WHEN (ISTATUS.EQ. 1) --> USRE2/USRN2 should be updated here
C     KELE  > 0            --> Element data is active
C     KNODE > 0             --> Node Data is active
```

User defined movement control (USRDSP)

DEFORM supports definition of the die movement for machines which cannot be controlled using the movement mechanisms given in the DEFORM system. The die speed routines are a functions which are called the USRDP subroutines based on the function number specified in the Object, Movement controls window as shown in Figure 6.4

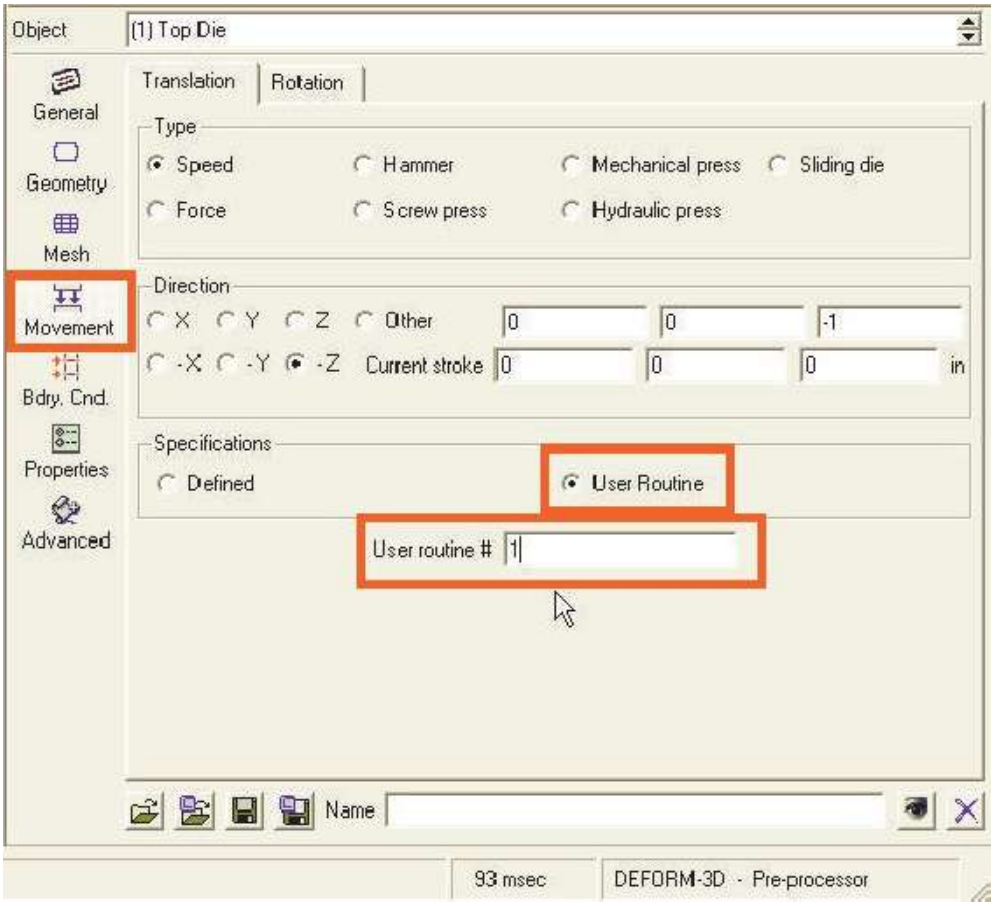


Figure 6.4: User defined die speed settings in Preprocessor.

The die movement can be a function of the following variables :

```
C     INPUT
C
C       TIME     = THE SIMULATED PROCESS TIME
C       PDIS     = PRIMARY DIE DISPLACEMENT
C       VX,VY,VZ = DIE SPEED IN X, Y & Z DIRECTIONS, RESPECTIVELY
C       STRKX,STRKY,STRKZ = CURRENT DIE STROKE IN X, Y & Z
DIRECTIONS,
C                 RESPECTIVELY
C       FRZX,FRZY,FRZZ    = DIE FORCE IN X, Y & Z DIRECTIONS,
RESPECTIVELY
C       AVGSRT   = AVERAGE STRAIN RATE
C       SRTMX    = MAXIMUM STRAIN RATE

C     OUTPUT
C
C       UPDV     = THE UPDATED DIE SPEED IN THE SPECIFIED DIRECTION
C       UPDF     = THE UPDATED DIE FORCE IN THE SPECIFIED DIRECTION
C
C       DTIME    = CURRENT TIME STEP                (I/O)
```

The output that the user has to provide from the user routine is :

```
UPDV   = THE UPDATED DIE SPEED IN THE SPECIFIED DIRECTION
UPDF   = THE UPDATED DIE FORCE IN THE SPECIFIED DIRECTION
DTIME  = DESIRED TIME STEP
```

To use the values for AVGSRT, STRMX the primary workpiece has to be specified as the object whose average and max strain rate are required. This is the keyword PDIE(2) in *Simulation Controls, Advanced Controls* in the pre-processor.

Examples of using the user defined die movement subroutine is given below :

Example Case #1

The die speed routine in DIESP1 controls the speed based on a user specified value for the average strain rate. The value of the average strain rate is specified using the USRDEF fields.

```
C
C  THE DIE SPEED OF THIS ROUTINE IS DETERMINED BY:
C
C  WHERE SR IS THE APPROXIMATED STRAIN RATE DURING
C          AN UPSETTING PROCESS
```

```
C       HI IS THE INITIAL BILLET HEIGHT.
READ(IUSRVL(1),*) HI
STRK = STRKX*STRKX + STRKY*STRKY
STRK = DSQRT(STRK)
C
C     FIND the Current Height
C
WRITE(6,*) TMPMX
HJ = HI - STRK
UPDV = AVGSRT * HJ
```

The goal of this routine is to define a die velocity by the following equation:

$$V = \dot{\varepsilon} \ (H_{initial} - S)$$

where:

V = Output die velocity

ε = Average strain rate

$H_{initial}$ = Initial billet height

S = Current die displacement

At the beginning of this code, a variable is fetched from our USRDEF fields, the initial height of the billet.

```
READ(IUSRVL(1),*) HI
```

The die displacement can be computed by the following equation:

$$S = \sqrt{D_x^2 + D_y^2}$$

The stroke is computed by the following code:

```
STRK = STRKX*STRKX + STRKY*STRKY
STRK = DSQRT(STRK)
```

Example Case #2

In the case of a screwpress, the rotational energy is converted to translational motion to form a part. The process ends when the energy

stored in the flywheel runs out or when the clutch on the drive mechanism is disengaged. Each step, the amount of energy may change due to energy being consumed by deforming the work piece. The total energy is an intial condition and the change in the current energy needs to be computed each step by,

$$E_O = E_I - \Delta E$$

where:

E_O = Energy after previous step

E_I = Energy at current step

ΔE = Change in energy over previous step

The change in energy can simply be calculated by the following equation,

$$\Delta E = \frac{F_I d_I}{\eta}$$

where:

ΔE = Change in energy over previous step

F_I = Die force over previous step

d_I = Distance traveled over previous step

η= The efficiency of the process.

Based on the current energy, the translational speed of the die can first be computed by calculating the rotational speed of the flywheel,

$$\omega = \sqrt{\frac{2E_O}{I}}$$

where:

ω = Rotational speed of the flywheel.

E_O = Energy of current step.

I = Moment of inertia

Using the rotational speed, the translational speed can be simply

determined by considering how the spindle shaft is threaded,

$$V_O = \omega(\pi d \sin\theta_t)$$

where:

V_O = The output translational velocity of the die.

ω = Rotational speed of the flywheel.

d = diameter of the spindle

θ_t = pitch angle of the spindle threads

This can be implemented in the following code:

```
      DATA ENERGY/ 10000.0/
      eff = 0.2
      MI = 10
      PI = 3.14159
      diam = 1.0;
      pitch = 0.1;

C     This calculates the change in the energy between steps
      e_change = (FRZY * STRKY) / eff

C     This updates the energy value
      ENERGY = ENERGY - e_change

C     This makes sure that the energy doesn't go negative
      if(ENERGY.LT.0) THEN
      ENERGY = 0.0
      endif

C     This computes the rotational speed based on the current energy
      rot_spd = SQRT((2*ENERGY)/MI)

C     This converts angular speed to rotations per second
      rot_spd = rot_spd / (2 * PI)

C     This calculates the tranlational velocity of the screw press
      V_out = rot_spd * PI * diam * sin(pitch);

C     This updates the value
      UPDV = V_out
```

User defined node and element value (USRUPD)

The user can implement subroutines which can calculate nodal and elemental values (up to 100) during the simulation for each node/element of the objects in the simulation. The inputs are all state variables and the outputs are the values for USRNOD, and USRELM. The variables can also be used in the flow stress routines to model flow stress as a function of new state variables.

The advantage of using these variables instead of doing the same procedure using user defined post-processing is that these values are calculated for each step in the database whereas user defined post-processing is only for the steps that are stored in the database.

Data that is passed to the user variable subroutine are stored in COMMON blocks as detailed below :

```
C
      COMMON /USRCTL/ DTK,KOBJ,ISTATUS,KSTEP
C     DTK    : TIME INCREMENT
C     KOBJ   : OBJECT NUMBER
C     KSTEP  : Step Number (N)
C     ISTATUS: 0 - the begain of the step
C              1 - the end of the step
C
      COMMON /ELMCOM/ RZE(3,8),URZE(3,8),STSE(6),EPSE(6),EFEPSE,EFSTSE,
     +                TEPSE,RDTYE,TEMPE(8),USRE1(100),USRE2(100),
     +                DTMPE(8),NODEE(8),KELE,KONP
C     RZE    : NODAL POINT COORDINATES (KONP corner nodes)
C     URZE   : NODAL POINT VELOCITIES  (KONP corner nodes)
C     STSE   : STRESS COMPONENTS
C     EPSE   : STRAIN RATE COMPONENTS
C     EFEPSE : EFFECTIVE STRAIN RATE
C     EFSTSE : EFFECTIVE STRESS
C     TEPSE  : TOTAL EFFECTIVE STRAIN
C     TEMPE  : NODAL TEMPERATURES
C     USRE1  : USER ELEMENT VARIABLES (INPUT: AT the Beginning of STEP
N)
C     USRE2  : USER ELEMENT STATE VARIABLES (OUTPUT: At the End of the
STEP N)
C     NODEE  : CONNECTIVITIES OF THE ELEMENT
C     KELE   : ELEMENT NUMBER
C     KONP   : NODES PER ELEMENT
C
      COMMON /ELMCOM3/ TEPS_NE(8),EFEPS_NE(8),DAMG_NE(8),STS_NE(6,8)
C     TEPS_NE     : Nodal eff. strain of the surrounding nodes
C     EFEPS_NE    : Nodal eff. strain rate of the surrounding nodes
C     DAMG_NE     : Nodal damage factor of the surrounding nodes
C     STS_NE      : Nodal stress components of the surrounding nodes
C
      COMMON /NODCOM/ RZN(3),URZN(3),DRZN(3),TEMPN,DTMPN,USRN1(100),
     +                USRN2(100),KNODE
C     RZN    : Nodal Point Coordinates
C     URZN   : Nodal Point Velocities
C     DRZN   : Nodal Point Displacement
```

```
C     TEMPN  : Nodal Point Temperature
C     DTMPN  : Nodal Point Temperature increment from last step to
current step
C     USRN1  : User Nodal Variables (Input: At the beginning of Step N)
C     USRN2  : User Nodal Variables (Output: At the end of Step N)
C     KNODE  : Node Number
C
      COMMON /NODCOM3/ EFEPS_NN,TEPS_NN,DAMG_NN,STS_NN(6),IELMNOD(3)
C     EFEPS_NN      : Nodal effective strain rate
C     TEPS_NN       : Nodal effective strain
C     DAMG_NN       : Nodal damage factor
C     STS_NN        : Nodal stress components
C     IELMNOD(1) = 0: Damage factor, Element definition
C                > 0:    ditto,      Node+element definition
C     IELMNOD(2) = 0: Eff. strain rate and strain, Element definition
C                > 0:          ditto,              Node+element
definition
C     IELMNOD(3) = 0: Stress components (El-plastic): Element
definition
C                > 0:          ditto,                Node+element
definition
C
C     COMMON /ELMCOM3/

C
C     WHEN (ISTATUS.EQ. 1) --> USRE2/USRN2 should be updated here or in
USRMSH.
C     Note:
C       If a user chooses to update USRE2/USRN2 in SUB. USRMSH, he/she
should also
C       copy all of USRE1/USRN1 to USRE2/USRN2 here.  When NUSRVE or
NUSRND are
C       greater than 2, more line should be added below.
C
C     KELE  > 0             --> Element data is active
C     KNODE > 0             --> Node Data is active
C
C     THE FOLLOWING EXAMPLES ARE:
C       TO STORE THE MAX PRICIPAL STRESS IN USRE2(1), AND
C       TO STORE THE STRAIN ENERGY IN USRE2(2).
C
C     At present NUSRVE and NUSRND are not passed into this routine.
 However,
C     if User Nodal and/or Elemental Variables are in use and NUSRVE or
NUSRND
C     are greater than 2, USRE1(3..NUSRVE) and/or USRN1(3..NUSRND)
should be copied
C     to USRE2(3..NUSRVE) and/or USRN2(3..NUSRND) below in the
appropriate places.
C
```

The variable USRN1 stores the nodal variables at the beginning of the step (the current value). After computing a new value for the user defined variables the results should be stored in USRN2 at the end of each step. For the element variables, USRE1 stores the values at the beginning of the step and the updated

value must be stored in USRE2.

If the variables are not being calculated, then the value stored in USRN1 and USRE1 must be copied to USRN2 and USRE2 respectively.
Examples of using the user defined nodal and element variables are given below:

1. The maximum principal stress is stored in the second user element value (USRE(2)) and the first element variable is not defined.

```
IF (ISTATUS.EQ.1.AND.KELE.GT.0) THEN
     USRE2(1)=USRE1(1)
     CALL USR_MAXPRN(STSE,PRNSTS)
     IF (USRE2(1).LT.PRNSTS) USRE2(1) = PRNSTS
ENDIF
```

2. In this example the average cooling rate (F/min) from 1300 F to 600 F is calculated and the result stored in the second user nodal variable (USRN2(2)). Here CURTIM is the current time in the simulation which can be accessed from the COMMON block CLOK.

```
COMMON /USER_DATA/ AMAX_TEMP, AMIN_TEMP, ADIF_TEMP
  DATA   AMAX_TEMP, AMIN_TEMP, ADIF_TEMP
 $       /  1300,      600,       700  /
  IF (ISTATUS.EQ.1.AND.KNODE.GT.0) THEN
     IF (TEMPN.LT.AMAX_TEMP.AND.USRN1(1).EQ.0.AND.
 $       TEMPN.GT.AMIN_TEMP) THEN
        USRN2(1) = CURTIM
        USRN2(2) = 0
     ELSE IF (TEMPN.LT.AMIN_TEMP.AND.USRN1(2).EQ.0) THEN
        USRN2(1) = CURTIM - USRN2(1)
        USRN2(2) = ADIF_TEMP/((CURTIM-USRN1(1))/60)
     ELSE
        USRN2(1) = USRN1(1)
        USRN2(2) = USRN1(2)
    ENDIF
  ENDIF
```

3. In many situations the flow stress may be a function of a user defined variable. In the example given below the value of strain is stored in the user variable and can then be used in the USRMTR routines to calculate the flow stress of the material.

```
     IF (ISTATUS.EQ.1.AND.KELE.GT.0) THEN
C
C     Strain = time increment * strain rate
C
        USRE2(1)=USRE1(1) + DTMAXC * EFEPSE
C
C     Calculate max principal stress and if greater than current
value
C     store in the user element value
C
        USRE2(2)=USRE1(2)
        RETURN
     ENDIF
C
     IF (ISTATUS.EQ.1.AND.KNODE.GT.0) THEN
```

```
      USRN2(1)=USRN1(1)
      USRN2(2)=USRN1(2)
      RETURN
   ENDIF
```

User defined damage models (USRDMG)

User defined damage models can be implemented for calculating damage or for use with the fracture module of DEFORM where elements can be deleted when their damage values exceeds a certain value. To use the damage model select the fracture mode (FRCMOD) as User Routines in *Materials Properties, Advanced* and specify the user routine number to be called in the subroutine USRDMG. (see Figure 6.5a)

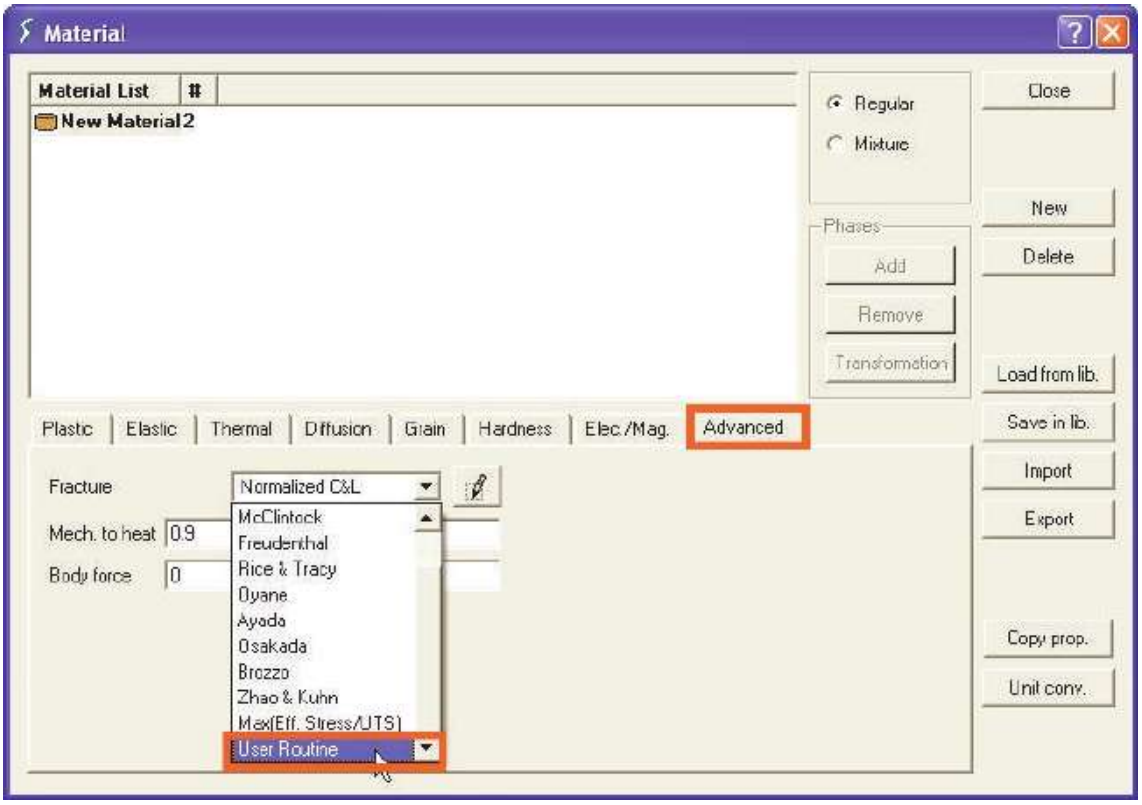


Figure 6.5a: Selecting the user defined damage model from the Material dialogs

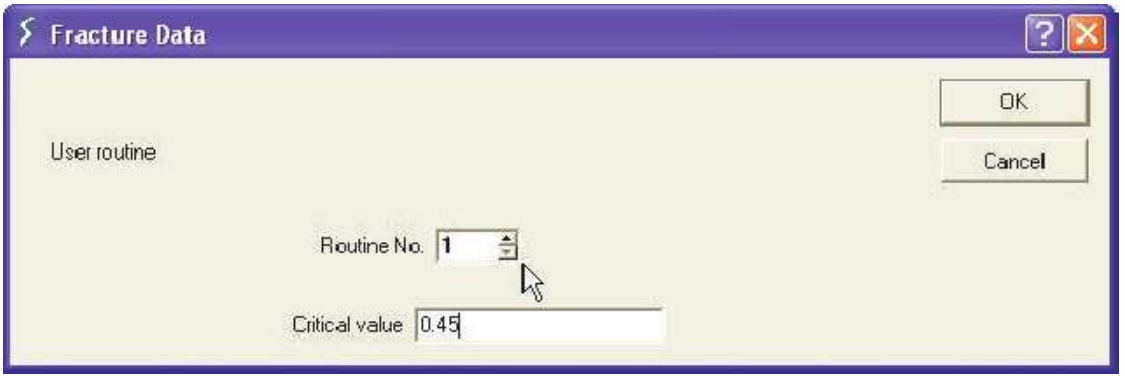


Figure 6.5b: Defining the routine number and critical value for damage

The damage routines are functions USRDM1 onwards with the inputs being as follows :

Input

```
C     NRT      =  DAMAGE MODEL NUMBER
C     STS      =  STRESS
C     EFSTS    =  EFFECTIVE STRESS
C     EFEPS    =  EFFECTIVE STRAIN RATE
C     DAMAG    =  PREVIOUS ACCUMULATED DAMAGE
C     STRLMT   =  STRAIN LIMIT
C     DTIME    =  TIME INCREMENT
```

In addition to the variables passed on as arguments, all the element common block ELMCOM variables (as seen in the USTMTR routines) can be accessed in these routines as well. (user needs to just insert the common block in to his damage routines)

The output is the new value of damage

Output

```
  DAMAG   =  NEW VALUE OF ACCUMULATED DAMAGE
```

The routine USRDM1 has an example on how the above routine is to be used. The default damage model in DEFORM is the Freudenthal criterion as follows. Note that after the calculation is made, the computed damage value is returned to the variable DAMAG. The calculation is performed only if two criterion are met:

1. The effective strain rate is above the limiting strain rate and
2. The effective stress is greater than zero.

Note that if the calculation is skipped, the current step value will be the same as the previous step value.

Here is the sample code included with the DEFORM-3D package:

```
C This routine calculates the accumulated damage based on
```

```
C the Freudenthal criterion for each object.
C
      IF (EFEPS.LE.STRLMT) GO TO 10
      IF (EFSTS.LT.0.) GO TO 10
C
      DAMAG = DAMAG + EFSTS*EFEPS*DTIME
C
   10  RETURN
```

User defined general routine (USRMSH)

This user routine is recommended when no other routine will accomplish what is desired. The flexibility of this routine makes this a very powerful option but often other routines will accomplish the same task in much less effort. This routine is called once at the beginning and end of each step. Also, it is called once for each object in the simulation. For example, if there are 3 objects in a simulation, this routine will be called three times (once for each object) at the beginning of each step and three times at the end of each step.

A list of the input variables is taken from the code and is listed as follows:

```
C
C     All FIELD VARIABLES CAN BE CHANGED IN THIS ROUTINE!
C
C     IMPROPER CHANGE MADE IN THIS ROUTINE WILL CAUSE PROBLEMS
C     IN THE ANALYSIS.
C
C     PLEASE USE THIS ROUTINE WITH CAUTION!!
C
C     This routine will be called at the beginning of the step and
C          at the end of the step
C     Object with FEM mesh will be passed to this Routine
C     REAL*8 array
C     RZ(3,NUMNP):   Nodal Coordinateis
C     DRZ(3,NUMNP):  Nodal displacemnts
C     URZ(4,NUMNP):  Nodal Velocities, and pressures (for tets only)
C     TEMP(NUMNP) :  Nodal temperatures
C     DTMP(NUMNP) :  Nodal temperature change in the step
C     FRZA(3,NUMNP): Nodal external forces
C     FRZB(3,NUMNP): Nodal reaction forces
C     PRZB(3,NUMNP): Calculated nodal pressures
C                    3rd component is normal pressure
```

```
C               vector sum of first two components gives traction
C               tangential to the surface
C         Note:   PRZB values for the rigid meshed dies are the
C         ----    interpolated values from the contacting workpiece.
C       for example traction 'TRACT' at node 'NODE' can be extracted as
C       TRACT = DSQRT((PRZB(1,NODE))**2.D0+(PRZB(2,NODE))**2.D0)

C       EFSTS(NUMEL):  Effective stress
C       EFEPS(NUMEL):  Effective strain rate
C       TEPS(NUMEL) :  Total plastic strain
C       RDTY(NUMEL) :  Relative element Densities
C       STS (6,NUMEL): Stress tensor components (Engineering definition)
C       EPS (6,NUMEL): Strain rate components (Engineering definition)
C       DCRP(6,NUMEL): Creep rate components (Engineering definition)
C       TSRS(LSTSR,NUEML): Strain components
C       Note) LSTSR is  total number of strain components defined by
Pre-processor.
C         Elastic, Plastic, Creep, Transformation plasticity,total
strain- 6 components
C         Thermal volumetric, Transformation volumetric - 1 component
C
C          Ex1) If elastic and plastic strain components are selected,
then LSTSR = 6+6
C          Ex1) If elastic,total and thermal volumetric strain
components are selected,
C               then LSTSR = 6+6+1 = 13
C
C       DAMG(NUMEL)   : Damages
C       USRVE(NUSRVE,NUMEL): User defined Element Variables
C       NUSRVE: Number of User defined Element Variables
C                 (Must be declared in the Pre-Processor)
C       USRVN(NUSRND,NUMEL): User defined nodal Variables
```

```
C      NUSRND: Number of User defined nodal variables
C                (must be declared in the Pre-processor)
C
C      ATOM(NUMNP):   Dominating Atom Contents
C      HEATND(NUMNP): Nodal Heat Source
C      WEAR(3,NUMNP): nodal wear parameters(for meshed objects)
C         WEAR(1,N): Interface temperature (in Deg. Absolute)
C         WEAR(2,N): Sliding velocity
C         WEAR(3,N): Interface pressure
C             Note:
C             ----   (WEAR(1:3,N) and PRZB components are computed
C                    for rigid meshed die nodes when
C                    tool wear models are turned on in Pre processor
C                    in the inter object data definition.
C      EVOL(NUMEL):   Elemental volume
C
C      MICRO-STRUCTURE RELATED VARIABLE
C
C        Available ONLY for Heat Treat applications
C
C        HDNS(2,*):    Hardness
C        VF(NTMATR,*): Volume Fraction
C        VFN(NTMATR,*): Transformation Starting Volume Fraction
C        DVF(NTRELN,*): Transformation Volume Fraction Change Ammount
C        TICF(NTRELN,*): Incubation Time
C        GRAIN(NGRNVAL,*): Grain Size
C
C      CURTIM: Current Time
C      KSTEP:  Current Step Number
C      DTMAXC: Time Step
```

```
C
C      Integer*4 Integer Array
C
C      NBCD(3,NUMNP): Nodal Boundary Condition
C                     0- Traction specified  => FRZA
C                     1- Prescribed Velocity => URZ
C                     2- Normal pressure Specified  => PRZA
C      NBCDT(NUMNP): Temperature Boundary Condition
C                     0- Prescribed Nodal heat
C                     1- Prescribed Nodal Temperature
C
C      NONP :         Nodes per element (4/8 for each tet/brick)
C      NOD(NONP,*):  Element Connectivities (global node numbering)
C      MATR(NUMEL):  Material group number
C      NBDRY(4,NUMFAC):   Boundary node list (Global node numbering)
C
C      Interger*4 Integer Variables
C
C      KOBJ : Current Object number
C      NUMEL: Total number of elements of KOBJ
C      NUMNP: Total number of nodes of KOBJ
C      NDSTART: Starting node number of KOBJ
C      NDEND  : Ending Node Number of KOBJ
C
C      NEDGE: Total number of oundary edges of KOBJ
C      NTMATR: Total number of Materials
C      NTRELN: Total Number of Inter-materail relations
C      NGRNVAL: Number of Grain-related Variables
C      NROUTINE: User Controlled Routine number (?)
C      AVGSRT: Average Strain Rate
```

```
C      SRTLMT: Limiting Strain Rate
C
C      ISTATUS: 0 -> Called at the beginning of each step prior to the
analysis
C               1 -> Called at the end of each step prior to writing to
database
C
C      IUSRFLAG: An integer flag at user's disposal
C      NWEAR_CMP : Number of parameters computed for tool wear
computations
C
C      NODAL DEFINITION FOR DAMAGE FACTPR
C        IELMNOD(1) = 0 -- NOT DEFINED
C                   > 0 -- DEFINED
C        DAMG_NP(NUMNP)   : Nodal damage factor
C      NODAL DEFINITION FOR EFFECTIVE STRAIN
C        IELMNOD(2) = 0 -- NOT DEFINED
C                   > 0 -- DEFINED
C        EFEPS_NP(NUMNP)   : Nodal eff. strain rate
C        TEPS_NP(NUMNP)    : Nodal eff. strain
C      NODAL DEFINITION FOR STRESSES IN ELASTOPLASTIC OBJECT
C        IELMNOD(3) = 0 -- NOT DEFINED
C                   > 0 -- DEFINED
C        DAMG_NP(NUMNP)   : Nodal damage factor
C
      COMMON /USRFLAG/ IUSRFLAG
C      IUSRFLAG: An integer flag at user's disposal
```

No output is required for this routine.

Example #1: Applying uniform distributed heating to a meshed object

In this example, we will apply heat to each element and to make the heating uniform, we will weight the heat to the volume of each element.

Here is an outline of the procedure in which this can be done.

1. A variable, HHH, is first defined that stores the heat rate per unit volume to be applied. The value in this example is 100. In SI units this is N*mm/mm^3. In English units this is klb*in/in^3.

2. Loop over the elements in the meshed object.

3. Inside the loop, calculate heat rate per node by multiplying by volume and then dividing by the number of nodes.

4. Loop over each node for a given element and add the heat rate to each individual node.

Here is the code that performs this:

```
C
C  The following example shows how the NODAL HEAT is added.
C  HHH is the heat rate per unit volume provided by the user.
C
      HHH = 100.0

      DO 500 L = 1, NUMEL
        HT = EVOL(L)*HHH/NONP
        DO I = 1, NONP
          N = NOD(I,L)
          HEATND(N) = HEATND(N) + HT
        ENDDO
  500  CONTINUE
```

Windows building procedure for FEM routines

Files provided by SFTC:
(Even though the procedures explained here are for Absoft Fortran v90, the corresponding object files and build scripts have been provided for older versions of Abosft compilers. Additionally active licensed users can use the compilation facilities provided in the USER area of DEFORM web site)

(1) Libraries
- DEF_SIM_USR.lib
- DEF_SIM_P4_USR.lib
- DEF_SIM_P4P_USR.lib

(2) Project files
- DEF_SIM_USR.gui

· DEF_SIM_P4_USR.gui
· DEF_SIM_P4P_USR.gui

(3) Sample fortran source files
def_usr.f
def_bcc.f
usr_crp.f
usr_dmg.f
usr_fit.f
usr_msh.f
usr_mtr.f
usr_part.f
usr_pm.f
usr_tevol.f
usr_tranfkine.f
usr_upd.f
usr_wear.f
usr_yield.f

PROCEDURE

Synopsis:

Compile def_usr.f and link this library with the DEF_SIM_USR_LIB.lib or DEF_SIM_P4_USR_LIB.lib or DEF_SIM_P4P_USR_LIB.libprovided by SFTC. As result, an executable file DEF_SIM.exe or DEF_SIM_P4.exe or DEF_SIM_P4P.EXE will be produced. SFTC provides two project files and one FORTRAN file that can be used as templates.

Procedure:

If you can find Compile_DEF_SIM_USR.bat in the current directory, you can compile the user routine by simply click on that batch file, and copy the DEF_SIM.exe or DEF_SIM_P4.exe or DEF_SIM_P4P.EXE to the folder where DEFORM3D has been installed.

To build DEF_SIM.exe follows these steps:

(1) Double click DEF_SIM_USR.gui , Absoft Pro Fortran compiler will open automatically.

(2) Click on icon or in the menu bar click on Tools->Build to build DEF_SIM.exe.

(3) Copy DEF_SIM.exe to the DEFORM3D/V6_1 directory (do not forget to make a backup copy of the original DEF_SIM.exe).

To build DEF_SIM_P4.exe follows these steps:

(1) Double click DEF_SIM_P4_USR.gui , Absoft Pro Fortran compiler will open automatically.

(2) Click on icon or in the menu bar click on Tools->Build to build DEF_SIM_P4.exe.

(3) Copy DEF_SIM_P4.exe to the DEFORM3D/V6_1 directory (do not forget to make a backup copy of the original DEF_SIM_P4.exe).

To build DEF_SIM_P4P.exe follows these steps:

(1) Double click DEF_SIM_P4P_USR.gui , Absoft Pro Fortran compiler will open automatically.

(2) Click on icon or in the menu bar click on Tools->Build to build DEF_SIM_P4P.exe.

(3) Copy DEF_SIM_P4P.exe to the DEFORM3D/V6_1 directory (do not forget to make a backup copy of the original DEF_SIM_P4P.exe).

Compiling user routines for UNIX platforms

After the FORTRAN code has been modified a new FEM engine can be build using the INSTALL3D script which is located in the $DEFORM3_DIR or using the script build_fem which is in the $DEFORM3_DIR/USR directory. If the INSTALL3D program is being used, the code in $DEFORM3_DIR will be altered (you must have write permissions). If you wish to build a local copy of the FEM engine code then copy all files from the $DEFORM3_DIR/USR directory to your local directory. Then run the script build_fem using the following command:

> build_fem

This builds a new copy of the FEM engine DEF_SIM.EXE in the local directory. After this process is completed, simulations using the new user defined routines can be run using the local copy of the FEM engine. Since the script 'build_fem' depends on correct identification of the operating system name and version number, when attempted on a system different from what was compiled at SFTC, 'build_fem ' may not always work. Under these conditions, user can directly use the script 'DEF_INS.COM' and specify the name of the machine and module name for building user defined binaries for DEF_SIM executables.

Running the modified FEM engine for UNIX platforms

If the FEM engine is built in the DEFORM directory with user routines, all users have access to the same user routines. If a local copy of the FEM engine is to be built and run then the DEF_ARM.COM script has to be copied to the local directory and the calls to DEF_SIM.EXE have to be modified to call the local copy of the program. In place of the calls

```
$DEFORM3_DIR/EXE/DEF_SIM.EXE
```

replace it with

```
./DEF_SIM.EXE
```

where ./DEF_SIM.EXE is the local copy of the FEM engine. When simulation jobs are submitted using the GUI or text based main program, the alias DEF_ARM is used to start the script DEF_ARM_CTL.COM which in turn runs DEF_ARM.COM.

When using a local copy of the FEM engine, copy DEF_ARM_CTL.COM to a local directory and change the alias for DEF_ARM to point to the local copy of DEF_ARM_CTL.COM. This alias is defined in the $DEFORM3_DIR/CONFIG.COM and a line in the .cshrc after the source $DEFORM3_DIR/CONFIG.COM redefining the alias should work. In the .cshrc file the following modifications can be made.

```
setenv DEFORM3_DIR '/disk1/deform/3d/v61'
source $DEFORM3_DIR/CONFIG.COM
#
# Old alias
#
alias DEF_ARM $DEFORM3_DIR/COM/DEF_ARM_CTL.COM
#
# New alias which is a local copy of DEF_ARM_CTL.COM
#
alias DEF_ARM $HOME/DEF_ARM_CTL.COM
```

Also in the local copy of DEF_ARM_CTL.COM the following calls should be replaced.

```
$DEFORM3_DIR/COM/DEF_ARM.COM
```

with the local version of

```
$HOME/DEF_ARM.COM
```

After this has been done simulations can be run using the local copy of the FEM engine and jobs can be started using the user-interface.
One problem that can occur is that if the .cshrc has an exit for non-interactive shells and the new definitions are after this, then they will never be defined when running a simulation. Place the new command to run the local copy of DEF_ARM just after the definitions for the regular version of DEFORM.

Running the modified FEM engine for Windows platforms

If the FEM engine has been built for Windows, the only way in which to utilize it is to swap it with the current engine. The current engine with be located with the current installation of DEFORM-3D usually in a directory such as

C:\DEFORM3D\V6_1. If it is not there, look for a file named DEF_SIM.EXE. (and the corresponding parallel P4 and P4P versions)

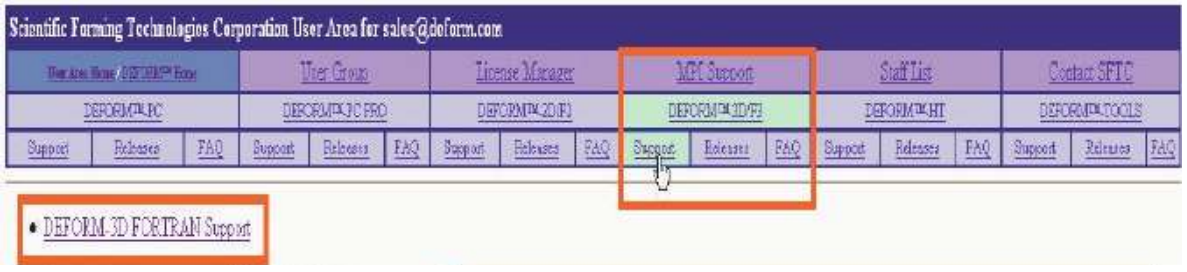
Compiling User Routines on the DEFORM Support Website for Windows Machines

There is now the possibility to compile user routines for Windows platforms using the SFTC website. The location of the website is as follows and is available to only current and active customers of DEFORM:
Go to
http://support.deform.com/3d/support/fortran/

Or go to the DEFORM web site and select the user area



Select DEFORM3D > Support > DEFORM-3D FORTRAN Support as follows



And select the correct DEFORM 3D version and follow the prompts to submit the user routine files and download the compiled binaries along with compiler messages and user source files (in zipped pack)

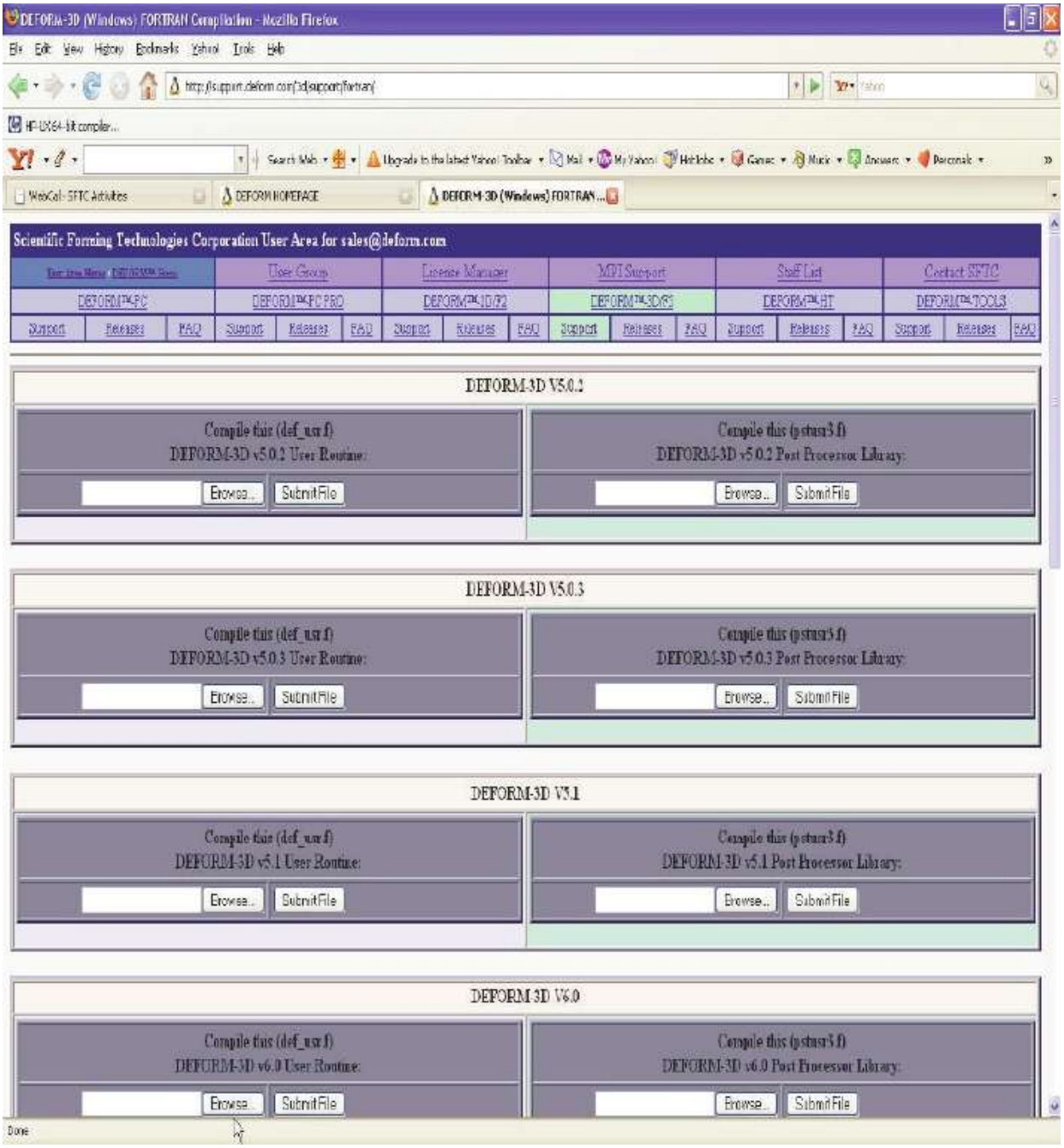


Figure 6.6 : The DEFORM webpage sequence for compiling user routines.

Figure 6.6 shows the webpage where the user routines are compiled. There are various versions of DEFORM listed on the website and the appropriate version

should be selected. The left half of the webpage is the user FEM routines that can be compiled and the right half are the user post-processing routines that can be compiled. To compile a user routine, click the appropriate Browse... button and load the fortran file that should be compiled. After this, click the Submit File button and the website will guide you to download a zip file. This zip file will contain the following information:

1. The original fortran file you submitted.
2. A message file with the output of the compiler/linker.
3. A .dll file or an .exe file if the compilation was successful. If not, please refer to the error message in output file from the compiler/linker.

For more information on user routines, please refer to the section on user routines in the manual.

6.2. User defined post-processing routines

User defined post-processing can be used to generate plots of user variables after running a simulation. It uses the steps that are stored in the database and any type of variable can be plotted in the post-processor for these steps. The calculation of this variable is done using a FORTRAN program PSTUSR.FOR (UNIX) or pstusr3.f (Windows) which is stored in the $DEFORM3_DIR/USR (UNIX) directory or the <DEFORM-3D installation directory>/User Routine/PostProcessor directory (Windows).

It is important to note that these variables do not affect the results of the simulation. There are 10 user-defined routines, each of these can be used to calculate 20 different user variables. After the code for these variables has been specified the user defined routines have to be compiled. The compilation procedure for UNIX and Windows differs and is outlined below. The end result of the compilation creates a shared dynamic library for the GUI post-processor. Some applications of user variables are to evaluate the micro-structure after the simulation, to predict the hardness, yield strength of different regions of the forged part, to evaluate failure using a critical damage value, calculate the cooling rate, etc...

User defined post-processing (USRVAR)
The user defined post-processing routines have to be written using FORTRAN in the file PSTUSR.FOR or pstusr3.f. When user variable tracking is done the functions in this FORTRAN program are called for the steps that have been selected in the post-processor. The user function is evaluated at each node/element of the object for which the variables are tracked. The user subroutine is called at the beginning of tracking to get the variable names, then at the first step to get the initial values for all variables and then called for all the

steps present in the database being tracked. There are three phases in the tracking process :

PHASE 1:
The user variable function is called with the INIT flag set to to "0". This is done once before tracking is started. During this phase the variables names (VNAME) should be defined so that they can be displayed in plots on the screen. The variables for which "VNAME" is defined are tracked.
For easier identification purpose, it is recommended that proper name or descriptions be assigned to "VNAME". This can be done when the user subroutine is called with INIT = 0 as shown below:

```
IF(INIT.EQ.0) THEN
  VNAME(1) = 'User Example  -1'
  VNAME(2) = 'User Example  -2'
  VNAME(3) = 'User Example  -3'
  VNAME(4) = 'User Example  -4'
  VNAME(5) = 'User Example  -5'
  RETURN
ENDIF
```

PHASE 2:
The user variable function is then called with INIT flag set to "1". This is the second phase in which all user variables have to be initialized to their starting values. This is called at the first step in the list of steps which are being tracked (ISTEP equals to the starting step) for each node/element in the object.

```
IF(INIT.EQ.1) THEN
  VAR2(1)= (STS(1)+STS(2)+STS(3))/3.
  VAR2(2)=EFEPS
  VAR2(3)=0.0
  VAR2(4)=0.0
  VAR2(5)=0.0
  RETURN
ENDIF
```

If the initial value of a variable is zero the this is defined using the following code (Here variable 5 is used)
VAR2(5) = 0.
If the maximum value of a variable is being tracked (for example temperature) then this value can be set to (Here variable 5 is used).
VAR2(5) = -10000
and then checked and increased if temperature at any step is greater than this.

PHASE 3:
The user variable function is called with the INIT flag set to "2" in the phase in which all calculations of the user variables are done. This function is called for all steps for each node/element of the object and the user is expected to update the values of the state variables based on the inputs passed to this program. The inputs to the function are:

```
C                              *** INPUT ***
C
C     TNOW          : Current time
C     DTMAX         : Max time step when set in simulation
controls
C                   : TNOW between successive steps can also
be
C                   : used to compute time step size.
C     RZ            : Element center coord.
C     TEMP          : Temperature
C     EFEPS         : Effective strain rate
C     TEPS          : Total accumulated strain
C     EFSTS         : Effective stress
C     DAMGE         : Damge factor
C     RDTY          : Relative density
C     STS           : Stress tensor
C     EPS           : Strain rate tensor
C     TSR           : Strain tensor
C     WEAR(5)       : WEAR(1)=Interface temperature on master
object nodes
C                   : WEAR(2)=Sliding velocity on master
object nodes
C                   : WEAR(3)=Interface pressure on master
object nodes
C     VAR1(1-95)    : Initial state variables
C     VAR2(1-95)    : Updated state variables (OUTPUT)
C     VNAME(1-95)   : Name for each variables
C     ISTEP         : Step number
C     INIT          : Flag for Different Operations
C                     0 - Define characteristic of the
subroutine
C                     1 - Initialize User Defined Variables
C                         (Initial Step)
C                     2 - Calculate User Defined Variables
C                         (Subsequent Steps)
```

```
C     IOBJ         : Object number in current object
C     NUMNP        : Number of nodes in current object (when
INIT=1,2)
C     NUMEL        : Number of elements in current object
(when INIT=1,2)
C     ICURNE       : Current node/element number in object
(when INIT=1,2)
C                    (depending on tracking at node or
element)
```

1. In the example given below the maximum mean stress is stored in the first variable, the maximum strain rate is stored in the second variable.

```
IF(INIT.EQ.2) THEN
  STSM = (STS(1)+STS(2)+STS(3))/3.
  IF(STSM.GT.VAR1(1)) THEN
    VAR2(1)=STSM
  ELSE
    VAR2(1)=VAR1(1)
  ENDIF
  IF(EFEPS.GT.VAR1(2)) THEN
    VAR2(2)=EFEPS
  ELSE
   VAR2(2)=VAR1(1)
  ENDIF
  VAR2(3) = VAR1(3)
  VAR2(4) = VAR1(4)
  VAR2(5) = VAR1(5)
ENDIF
```

In the statement to check for INIT = 2, each user variable is checked with some criterion, if it meets this then the value is updated, else the new value is set the same as the previous value VAR1. It is very important that if the value is not updated, the value is set to the VAR1 value as this allows tracking of variables across remeshing steps where data is interpolated from an old mesh to a new mesh.
Since this calculation is done for each step and for each node/element the code should be written efficiently. You should not open close files for each subroutine call as this can degrade performance. The /DATA/ statements in FORTRAN can be used to store parameters and the /COMMON/ block fields can be used to hold static data. If files are to be opened then use UNIT numbers from 91..99 for opening these files as opening other unit numbers might clash with files opened in other parts of the program.
The mesh number should always be incremented when there is a change in the mesh. If a remeshing step has been purged from a database then user-variable

tracking will not work with the database.

Compiling post-processing user routines on UNIX machines

After the code has been modified the new TEXT based post-processor and the shared dynamic library for the GUI post-processor have to be generated using the INSTALL3D program which is located in the > build_pst This builds a new copy of the post-processor DEF_PST3.SL in the local directory, a shared library that can be used with the GUI post-processor.

After this process is completed then the new variables can be accessed from the GUI post-processors. In the GUI based post-processor, when tracking of user variables is done, a post-processor database (PDB) is generated for the values of the tracked variables. If the same data is to be viewed again in the post-processor, after loading the database, the post-processor database can be loaded to view the data. This saves the time required to track these variables again.

Running the modified post-processor in UNIX

The GUI post-processor looks for the shared library DEF_PST3.SL first in the current working directory then in the users HOME directory under the subdirectory DEFORM3 and then in the DEFORM_DIR/USR directory. It reads this shared library and displays the list of existing variables which can be tracked. In *User Variables* dialog the routine number, the object for which tracking is to be done, and the option to track at the node or element can be selected to carry out tracking. Once tracking is carried out all the data is stored in a post-processor database (PDB) file. Variables can then be plotted by selecting the variables in the *State Variables* menu.

Compiling post-processing user routines on Windows machines

The recommended way in which to do this is to use the online facility available through the SFTC web site. To learn more about this, please consult the previous section on user-defined FEM routines. (Figure 6.6)

To generate the .dll file directly using AbSoft, please use the instruction below:

Requirements

- ABSOFT Pro Fortran 9.0 (version 7.0, 7.5 are also supported)

FILES PROVIDED BY SFTC:

1. *Files needed by the project*
- Pstusr3.f
- PC_pstusr3.f
- PC_pstusr3.als
- PC_pstusr3.xps

2. *Project file*
- USR_DEF_PST3.gui

PROCEDURE

Synopsis:

Compile *USR_DEF_PST3.gui* to generate a Dynamic-Link Library --- USR_DEF_PST3.dll

Procedure:

If you can find Compile_DEF_PST_USR.bat in current directory, user routine post processor can be compiled by double click on that batch file.

To generate USR_DEF_PST3.dll follow these steps:
1. Double click USR_DEF_PST3.gui (USR_DEF_PST3_Absoft70.gui for Absoft 7.0 compiler), Absoft Pro Fortran compiler will open automatically.
2. Click on icon  or in the menu bar Click "**Tools->Build**", to build *USR_DEF_PST3.dll*.
3. After finishing with the set up of the project, customize *pstusr3.f* and rebuild *USR_DEF_PST3.gui.*
4. Copy *USR_DEF_PST3.dll* to the **DEFORM3D**/**V6_1**/**USR** directory.

Quick Reference

General
This is a quick reference guide for common DEFORM forming simulations. It gives a quick overview for the most common problem types. For problems that are not covered here, refer to the online help manual, or contact Scientific Forming technical support at (614) 451-8330.

Cold Forming

When simulating cold forming, it is important to simulate all steps of the process, since the work hardening effects of early steps can influence behavior of later steps. The procedure for simulating a multi-station process is detailed at the end of this section.

1. Create a new problem folder (directory)
Each DEFORM problem should reside in its own folder. However, a given problem may contain many operations, all in the same database file.

2. Start the DEFORM preprocessor

3. Set Basic Simulation Controls
a. Problem Title (optional)
Descriptive title for the problem – will be displayed on the screen during pre- and post-processing.
b. Unit System
Select English or SI units. This will change many default values and affect how material data is imported.

c. Select Simulation Mode
For cold, isothermal simulations, deformation mode should be turned on (yes), heat transfer and all other modes should be turned off.

4. Define the Material
Define Plastic material properties for the workpiece material. Material data can be loaded from the DEFORM material database. Be careful to select a material with data in the temperature range you will be simulating.

5. Define the Workpiece
a. Define Object Type
For most simulations, plastic object type is suitable.

b. Define the Geometry

Geometry can be defined from an STL file, or by entering table data.
Workpiece geometry requirements:

- Geometry normals must always face outward
- Always check the geometry.

6. Mesh the Object

Typical progressions should use element sizes representative of the geometric detail of the part. When in doubt, more elements will tend to give more accurate results. Typical weights:

- Curvature=0.9
- Strain Rate=0.7
- Strain =0.5
- Temp=0

7. Define Boundary Conditions

Symmetric planes should be defined either with a velocity boundary condition or with a symmetry boundary condition.

8. Define the Tools

a. Assign Tool Names (optional)

For reference during pre- and post-processing.

b. Define Tool Geometry

Import or define geometry for the punch and die. The geometry rules detailed above for the workpiece all apply. Furthermore:

- For multi-piece tools, draw a single tool boundary defining all inserts, knockouts, etc. The geometry can be separated later for die stress analysis. Refer to the manual or contact tech support if there is more than one moving tool for a given station.
- Extend the tool geometry slightly across the centerline.
- For tools with a sliding clearance between the punch and die, increase the OD of the punch to slightly intersect the die. Refer to the manual or training material for more details
- Put a slight flat on the tip of any pointed punches.
- While it is not strictly necessary, it is convenient to make object 2 the punch or moving object.

c. Define Press Movement for the Punch

In general, press speed will not influence simulation results for cold forming simulations. A constant press speed of 10 in/sec or 250mm/sec is generally adequate. Set the direction (typically downward for punch on top). The stroke value should generally not be changed by the user.

9. Define Interobject Data

a. Inter-Object Relationships

The workpiece should be slave to both tools. For well lubricated cold forming, a friction factor of 0.08 is reasonable.

b. Object Positioning

Using interference, with the workpiece as the reference object, position both tools in contact with the workpiece. For extremely small parts, a smaller interference tolerance may be necessary to prevent excessive tool-workpiece overlap.

c. Generate Contact Boundary Conditions

The default value of tolerance is adequate.

10. **Complete Simulation Controls**

a. Define Number of Steps

- For very limited deformation such as a square-up, use 50 steps
- For average deformation, such as heading, use 100 steps
- For problems with a large amount of deformation, such as extrusion, use 200 steps

Save every 5 to 10 steps.

b. Select the Primary Die

Enter the object number for the punch(generally object #2)

c. Calculate the Stroke per Step

Estimate the distance the punch will move (total stroke). Divide this value by the number of steps, and enter this value in Stroke per Step. If you are unsure of total punch stroke, add 10 or 15 extra steps. This will overshoot the goal, and you can back up a few steps
to get the final result.

d. Set stopping controls (optional)

If you know the exact distance the punch will move, enter this value under stopping controls. If no value was entered, the simulation will stop when all steps have been completed.

e. Set substepping control

Under Advanced Step Controls, set **Strain per Step** to 0.025.

11. **Save the Data**

Save a Keyword file.
Go to Database Generation. Check the data. If there are any yellow or red flags, resolve them, then generate the database. Exit the preprocessor and start the simulation.

12. **Running a Second Operation**

a. Identifying the endpoint of the first operation

After the simulation is completed, go to the Post-Processor, and check the results. Identify the step at which the first operation will be completed, and make a note of this step number.

b. Loading Simulation Results Into the Preprocessor.

Return to the preprocessor, and load the appropriate step from the database.

c. Changing Tool Geometry

Go to the Geometry editor, delete the tool geometry (*not the whole object)*, and import or create new tool geometry for each tool.

d. Positioning Objects

From Interobject reposition the tools against the workpiece using interference.

e. Generate Contact

Initialize and generate contact boundary conditions.

f. Reset Simulation Controls

Determine total steps and stroke per step as described above.

g. Reset Stopping Stroke

If the stroke stopping control was used, reset the stroke to zero on object movement controls, and reset the stopping control under simulation controls.

h. Write The Database

Writing an old database will append data to the end of the existing database. It will overwrite any steps *after* the step that was loaded.

i. If the appropriate ending step is not saved…..

If you encounter a situation where, say, step 90 is not formed enough, step 100 is formed too much, and there are no steps saved in between, you can load step 90, change the save interval to 1 (save every step), then rerun the last 10 steps of the simulation to get the proper stopping step.

Hot Forming

When simulating hot forming, it is important to simulate all steps of the process including transfer from the furnace, and resting on the die, since the temperature loss due to transfer from the furnace and from die chilling can influence flow behavior. The procedure for simulating a multi-station process is detailed at the end of this section.

This section will outline the setup procedure for the following operations:

1) Set uniform object temperature to simulate full furnace soak.
2) Cool in air to simulate transfer from the furnace to the dies
3) Rest on dies
4) Forge
5) Repeat for multiple forging blows

1. **Create a new problem folder (directory)**

Each DEFORM problem should reside in its own folder. However, a given problem may contain many operations, all in the same database file.

2. **Start the DEFORM preprocessor**

3. **Set Basic Simulation Controls**

The simulation controls will be set for the first operation – simulating chilling during the transfer from the furnace to the press.

a. Problem Title (optional)

Descriptive title for the problem – will be displayed on the screen during pre- and post-processing.

b. Operation Name

Operation name

c. Unit System

Select English or SI units. This will change many default values and affect how material data is imported.

d. Select Simulation Mode

For simulating transfer of the workpiece from the furnace to the press, only heat transfer will be modeled, so turn on heat transfer, and turn off deformation.

4. **Define the Material**

Define thermal properties for the material. For most steels, the files STEEL_E.KEY or STEEL_S.KEY contain reasonable thermal properties. These files can be loaded from the Material Properties menu on the main preprocessor window.

5. **Define the Workpiece**

a. Define Object Type

For most simulations, plastic object type is suitable.

b. Define the Geometry

Geometry can be defined from an STL file.

Workpiece geometry requirements:

- Geometry must be defined with normals facing outward
- Always check the geometry.

c. Mesh the Object

Typical progressions should use element sizes that represent the features of the part. When in doubt, more elements will tend to give more accurate results.

Typical weights:

- Curvature=0.9
- Strain Rate=0.7
- Strain =0.5
- Temp=0

d. Define Thermal Boundary Conditions

For *solid* parts, define heat exchange with the environment on all faces *except* the symmetry surfaces.

e. Initialize Object Temperature

Set the initial object temperature to the furnace soak or preheat temperature.

6. **Complete Simulation Controls**

a. Define Number of Steps

A typical transfer operation can be simulated in 10 to 20 steps.

b. Select the Primary Die

Enter object 1 (only object defined) as the primary die.

c. Calculate the Time per Step

Divide the total transfer time by the number of steps.

7. **Save the Data**

Save a Keyword file.

Go to Database Generation. Check the data. If there are any yellow or red flags, resolve them, then generate the database. Exit the preprocessor and start the simulation.

8. **Loading Simulation Results Into the Preprocessor to Define Second Operation**

Return to the preprocessor, and load the last step from the database.

9. **Define Material Data for the tools**

Specify thermal data for the tool material

10. **Define the Tools**

a. Assign Tool Names (optional)

For reference during pre- and post-processing.

b. Define Tool Geometry

Import or define geometry for the upper die (punch) and lower die. The geometry rules detailed above for the workpiece all apply. Furthermore:

- For multi-piece tools, draw a single tool boundary defining all inserts, knockouts, etc. The geometry can be separated later for die stress analysis. Refer to the manual or contact tech support if there is more than one moving tool for a given station.
- Extend the tool geometry slightly across the centerline.
- For tools with a sliding clearance between the punch and die, increase the OD of the punch to slightly intersect the die. Refer to the manual or training material for more details
- Put a slight flat on the tip of any pointed punches.
- While it is not strictly necessary, it is convenient to make object 2 the punch or moving object.

c. Mesh the Tools

Use around 400 elements. Use user defined density with 3 at the contact surface and 1 on the back side of the die.

d. Assign Tool Material

Be sure the proper material is assigned to both the tools and the workpiece

e. Assign Tool Temperature

Set the uniform tool temperature before forming begins.

11. **Define Interobject Data**

a. Inter-Object Relationships

The workpiece should be slave to both tools. The interface heat transfer coefficient should be about 0.004 for English units, or 10 for SI units.

b. Object Positioning

Using interference positioning, position the workpiece on the bottom die. Leave the top die away from the workpiece during this operation.

c. Generate Contact Boundary Conditions

The default value of tolerance is adequate.

12. **Complete Simulation Controls**

a. Define Number of Steps

A typical die resting operation can be simulated in 10 to 20 steps.

b. Select the Primary Die

Enter object 1 as the primary die.

c. Calculate the Time per Step

Divide the total transfer time by the number of steps.

d. Define Operation Name (optional)

13. **Save the Data**

Save a Keyword file.
Go to Database Generation. Check the data. If there are any yellow or red flags, resolve them, then generate the database. Exit the preprocessor and start the simulation.

14. **Loading Simulation Results Into the Preprocessor to Define Forming Operation**

Return to the preprocessor, and load the last step from the database.

15. **Set Simulation Controls**

a. Operation Name (optional)

b. Select Simulation Mode

We will now simulate the forging operation, so turn deformation on (yes)

16. **Define Material Data for Workpiece**

Assign plastic (flow) data for the workpiece material. Be sure the material selected covers the proper temperature range, including any deformation heating and die chilling.

17. **Assign Top Die (punch) Movement**

Press behavior may play a role in results. Consult the manual or SFTC tech support for more information on press behavior.
Set the direction (typically downward). The stroke value should not be changed unless a mechanical press model is used.

18. **Define Interobject Data**

a. Inter-Object Relationships

Assign a friction factors. For lubricated hot forging, values of 0.2 to 0.3 are typical. For non-lubricated hot forging, values of 0.8 to 1.0 are typical.

b. Object Positioning

Using interference, with the workpiece as the reference object, position both tools in contact with the workpiece.

c. Generate Contact Boundary Conditions

The default value of tolerance is generally adequate

19. **Complete Simulation Controls**

a. Define the Number of Steps

- For very limited deformation, such as a square-up or buster, use 50 steps
- For average deformation, such as heading, use 100 steps
- For problems with a large amount of deformation, such as extrusion, 200 or more steps are appropriate.

b. Assign the Primary Die

The top die (or punch) should be the primary die. This will generally be object #2.

c. Calculate the Stroke per Step

Estimate the distance the top die will move (from the point it contacts the workpiece) Divide this value by the number of steps, and enter this value in Stroke per Step. If you are unsure of total punch stroke, add 10 or 15 extra steps. This will overshoot the goal, and you can back up a few steps to get the ending shape

d. Set Stopping Controls (optional)

If you know the exact distance the punch will move, enter this value under stopping controls. If no value was entered, the simulation will stop when all steps have been completed.

e. Set substepping control

Under Advanced Step Controls, set **Strain per Step** to 0.025

20. **Save the Data**

Save a keyword file.

Go to Database generation, Check the data. If there are any yellow or red flags, resolve them, then generate the database. Exit the preprocessor and start the simulation.

21. **Running a Second Operation**

a. Identifying the endpoint of the first operation

After the simulation is completed, go to the Post-Processor, and check the results. Identify the step at which the first operation will be completed, and make a note of this step number.

b. Loading Simulation Results Into the Preprocessor.

Return to the preprocessor, and load the appropriate step from the database.

c. Simulate Chilling During Transfer to Next Station

Consider the transfer time between stations. Refer to the beginning of this section for guidelines on running heat transfer simulations.

d. Changing Tool Geometry

Go to the Geometry editor, delete the tool geometry (*not the whole object)*, and import or create new tool geometry for each tool.

e. Positioning Objects

From Interobject reposition the tools against the workpiece using interference.

f. Generate Contact

Initialize and generate contact boundary conditions.

g. Reset Simulation Controls

Determine total steps and stroke per step as described above.

h. Reset Stopping Stroke

If the stroke stopping control was used, reset the stroke to zero on object movement controls, and reset the stopping control under simulation controls.

i. Write The Database

Writing an old database will append data to the end of the existing database. It will overwrite any steps *after* the step that was loaded.

j. If the appropriate ending step is not saved…..

If you encounter a situation where, say, step 90 is not formed enough, step 100 is formed too much, and there are no steps saved in between, you can load step 90, change the save interval to 1 (save every step), then rerun the last 10 steps of the simulation to get the proper stopping step.

Appendices

These appendices are topical treatment of specific information concerning DEFORM. In the future, these subjects may be shuffled into the main content of the manual and other information may be added in its place. The topics may be updated by other methods and may become out of date. Please contact Scientific Forming Technologies Corporation if there are any questions on the current status of any information.

Appendix A: Running DEFORM in text mode

DEFORM contains text based modules which can be used to set up and run simulations in automatic mode without going through the graphic user interface (GUI).

The text based preprocessor DEF_PRE.EXE can be used to assemble input data and generate a DEFORM database. It contains most of the same functionality of the graphic interface.

The job can be submitted by calling the simulation control script DEF_ARM_CTL.COM.

Assembling input data

The preprocessor can be controlled by redirecting a text input control file to the following program.

C:\deform2d\v5_1\DEF_PRE.EXE < DEF_PRE_INP.txt

Where DEF_PRE_INP.txt contains the following lines:

<CR>
2
1
DEF_COMMANDS.KEY
<CR>
E
E
Y
<CR>

Which are the user inputs if the text based system were run in interactive mode.

The file DEF_COMMANDS.KEY contains a series of “Action Keywords” which trigger the preprocessor to perform a series of options. (The file name is the users choice). Action keywords are documented separately. It also can contain standard keywords to define simulation controls (time stepping, stopping controls, etc).

An important function of action keywords is to trigger the input of other keyword files, which can include geometry definition, boundary conditions, etc.

Sample contents of the DEF_COMMANDS.KEY file. Contents of an actual file will be defined by the user.

```
KFREAD
DEFAULT.KEY
CURSIM 1
SIMNAM
Solution
DTMAX 0.001
DTPMAX 10 0.01 100
STPINC 10
KFREAD
DEF_MESH.KEY
KFREAD
DEF_EDGE_BCF.KEY
NDTMP 1 0 70.0
TMAX 10800
ENVTMP 2130
KFREAD
AIR_BC.KEY
USRSUB  1  1
GENDB 2
DEFORM_DEMO.DB
```

This file:
Reads DEFAULT.KEY which contains default problem settings
Sets the operation number to 1
Sets temperature sub stepping and save increment
Reads DEF_MESH.KEY
Reads DEF_EDGE_BCF.KEY which contains boundary condition definitions
Sets the temperature of all nodes to 70 degrees
Sets simulation time
Sets environment temperature
Loads AIR_BC.KEY which contains convection coefficients for air
Defines a user subroutine to be used
Generates a database called DEFORM_DEMO.DB.

Brief Introduction to Keywords

There are two different types of keywords that can be read by the Preprocessor: *Input keywords* and *Action keywords*. Input keywords contain data that is directly used as data for a simulation. This can be a geometry definition, convection coefficient values, or other such data. Action keywords perform certain operations when the Preprocessor is reading the data. For example, the

keyword KFREAD tells the Preprocessor to read the next line into the Preprocessor as a keyword file. This is quite useful for segregating data into different keyword files and being able to load them in a modular manner into the Preprocessor. The most commonly used Action keywords are KFREAD (keyword file reading), DBREAD (database file reading), GENCTC (generate contact based on proximity distance to dies), and GENDB (generate database). All the keywords are referenced with their specification method in the keyword reference.

Help For Optimization Users

In the case where optimization being performed, many simulations have to be run sequentially. In this case it means that the input data will have to change each time a simulation is run. This can be done by segregating the data into different keyword files and loading the all together at the initialization phase. For example, consider the case where the heat flux is different between each simulation. As long as that is the only difference between each run, the initial data can be stored in a keyword file and the heat flux data can be stored in another keyword file (or created on the fly by the optimization script). This can be done in a few easy steps:

1. Define the simulation and save it in a keyword without the heat flux definition.
2. Read the definition of heat flux in the keyword manual (keyword: ECHFLX).
3. Define a keyword file with the heat flux data. An easy cheat would be to define it in the graphical preprocessor, save it to a keyword file, extract it from the saved keyword, and modify it as necessary during run time.
4. Run a simple script file as above that loads in both keyword files and generates a database.

Running a Simulation

A simulation can be executed by running it directly from the command line calling the DEF_ARM_CTL.COM simulation control script.

The inputs are

DEF_ARM_CTL.COM *problem_id* B
Where *problem_id* is the database file name, with the .DB extension stripped. In the case above, we would run the simulation for DEF_DEMO.DB with the line:

F:\DEFORM3D\5_1\DEF_ARM_CTL.COM DEFORM_DEMO B

The *B* indicates that the simulation should be run in Batch mode.

Extracting the Results

There are two ways of performing this action: the text-based Preprocessor and the text-based Postprocessor. Which one is used depends on the desired output. The most brute-force but straightforward operation is to open the text-based preprocessor, load the last step of the database and save the data as a keyword file. This keyword file can then be parsed for any required information such as node temperatures (keyword: NDTMP). The action keyword that allows the user to read a database file is DBREAD and the action keyword that allows the user to write a keyword file out is KFWRIT that is specified the same manner as the KFREAD keyword. The keyword file can be read as in the case of assembling the input data as such:

```
DBREAD 0
DEFORM_DEMO.DB
KFWRIT
OUTPUT.KEY
```

The DEF_PRE_INP.txt would be the same where the DEF_COMMANDS.KEY file would contain the four lines above. The last step of the database would be read and stored in the keyword file named OUTPUT.KEY.

Appendix B: Inserting DEFORM™ Animations in Powerpoint Presentations

{Based on Powerpoint 97 running on Windows 98 platform – other variations may be slightly different}

Legend:

[Comments in square brackets (parentheses) are instructions]

Underlined comments are actual key strokes

The Process:

[Arrange suitable directory structure, for example, POWERPOINT file in same directory as DEFPLAY.EXE and presentation files in subdirectories immediately below]

[Open POWERPOINT file and go to page where animation is to be inserted]
insert object\create from file\browse [find DEFPLAY.EXE in the browse window and select it] ok [locate new object icon on page (perhaps drag it to the bottom left corner)]

[right click on icon]
edit package package object\edit command line… [type "DEFPLAY.EXE presentation_file_full_path –e"]
ok\file update\file exit
[where "presentation_file_full_path" is for example:
"d:\work\presentation\extrude\extrude.pre"]

[right click on icon]
action settings\mouse click\action on click\none
action settings\mouse over\action on mouse over\none

[right click on icon]
custom animation\timing\start animation\animate\1 seconds after previous event
[1 seconds is optional]
custom animation\effects\entry animation and sound\no effect
custom animation\chart effects\none
custom animation\play settings\object action\activate contents [check box "hide while not playing"] ok

[save powerpoint file:] file\save

[Some platform/powerpoint version combinations may give warnings of OLE

object that may contain viruses; this may be removed by:]
tools\options\general\general options [uncheck box “macro virus protection”]

From 3DV6.1, on PC the system supports direct saving of model results in the form of power point files movie files as explained in section 4.3.20 of this help system.

Appendix C: DETAILS OF MOVEMENT CONTROLS IN SPIN.KEY

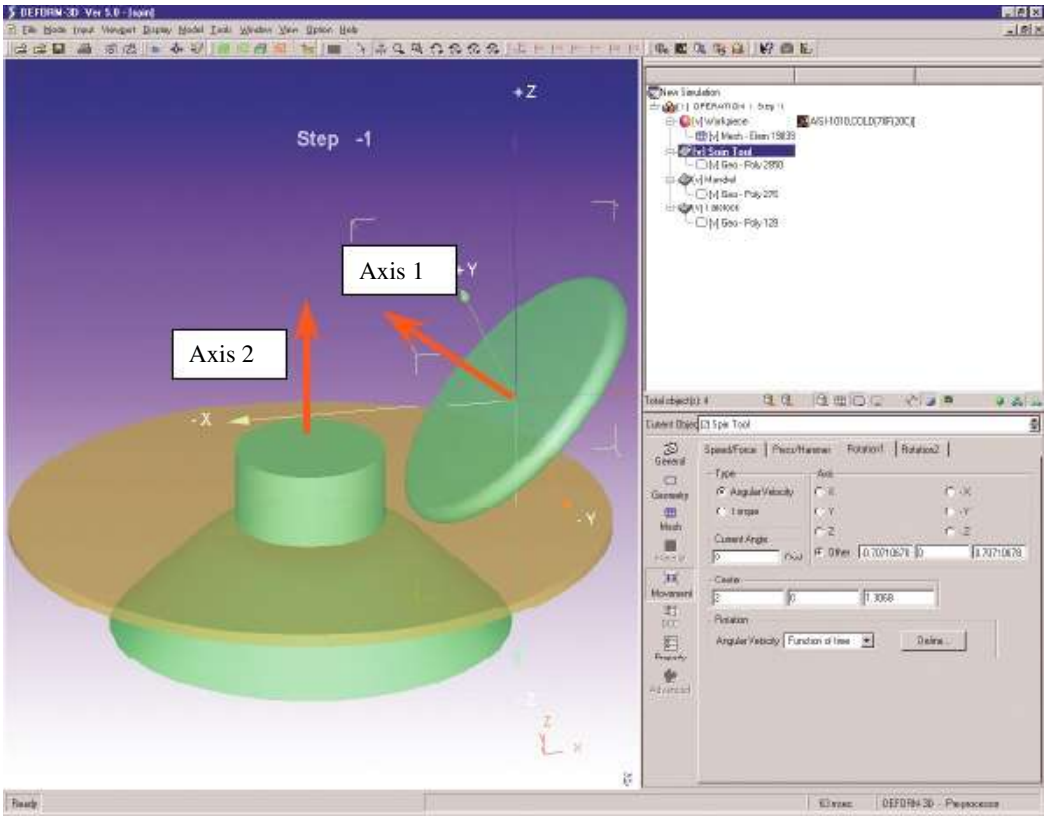


Figure c.1 : Tool arrangement in SPIN.KEY

Introduction:

The file SPIN.KEY is located in the DATA subdirectory of the DEFORM3D installation. This appendix is a write-up for how the movement controls were defined for that case.

Movement controls for spinning.

Work piece is held stationary. Tool spins around Axis 2.

Tool is initially located along X axis.

Tool translational movement towards or away from Axis2 is defined along the X axis. As the tool rotates around Axis2, the direction of translational movement is adjusted so that it is always towards/away from Axis2. (Figure c.2 illustrates the updating of this velocity vector).

Movement in the Z axis is accomplished by simultaneously moving the mandrel and tailstock.

Non-linear tool movement can be achieved by decomposing the movement path into Vx(t) and Vz(t). The Vx(t) function is assigned to the tool, and Vz(t) is assigned to the tailstock & mandrel.

The angular velocity of the spin tool should be defined such that the relative velocity between the work piece and tool is zero at the point of contact. Thus

$$r_1 \omega_1 = r_2 \omega_2$$

where ω_1 is the angular velocity of the work piece (or the tool about the work piece in the simulation) and ω_2 is the angular velocity of the tool about it's own axis. r_2 is the radius of the contact point from the center of the work piece, and r_1 is the work piece.

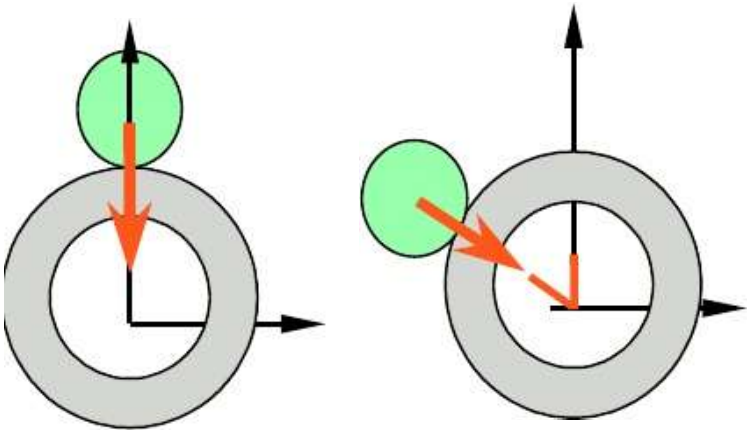


Figure c.2 : Automatic updating of radial velocity defined along an axis.

Appendix D: Data Files

Below is a comprehensive list of data files that can influence the behavior of a solution as well as a brief description of their purpose.

LAY.DAT

This data file is used for mesh consolidation for cogging simulations.

STRETCH.DAT

This data file is used to activate the check of a surface edge stretch to determine if a remeshing is needed.

AXIS.DAT

This data file allows the user to perform special options on rotating work piece simulations.

STNCMP.DAT

This file allows the user to turn off strain component interpolation.

SW2SP.DAT

This file allows the user to specify the maximum number of elements which the solver can use the Sparse solver. The purpose of this is to control the size of the problem where the sparse solver can be used since it requires more memory than the C-G solver. In the case where sparse solver is not used, the simulation will only use the C-G solver. This option should only be enabled for cases where the C-G solver can be used, i.e. single deforming plastic work piece with no load controlled dies.

NBC.DAT

This file allows the user to enable multiple contact conditions on nodes in die corners. This eliminates nodal oscillation in die corners.

ALE.DAT

This triggers the steady state solution method to be used during the simulation.

SPRING.INI

This file allows the user to enable the usage of spring-loaded dies.

SPRING2.INI

This file allows the reversible spring loading direction to be specified.

DEF_RSE.DAT

This file allows to enable special features of the Rigid Super-Element scheme to be used during a simulation.

Appendix E: 2D to 3D Conversion Utility

Purpose: (available in DEFORM 2D only)

Revolves or extrudes 2D geometry and all process variables from 2D database to 3D keyword file.

How it works:

3D elements are created directly from 2D elements. In the case of axisymmetric simulations, the elements are revolved about an axis and for plane strain, the elements are extruded along an axis. For either case, the user has the option of Brick (8 node) or Tetrahedral (4 node) elements. If there are a large number of 2D element it is recommended to use manual remeshing to remesh the 2D object with 100-400 elements. After this, interpolate the state variable. Generate a database step with the new data. The new step will be negative, which you will specify in the conversion utility.

How to run:

Unix:

From a command prompt, type M23

Windows:

Open a DOS command window and change to the problem directory (cd \deform3d\problem….).

Run the M23.COM utility \deform2d\v90\M23.COM

a. <Enter Problem ID without extension > (e.g. SPIKE)

b. Enter 3d keyword file name [SPIKE_3.KEY] hit <CR> to accept default

c Enter Step Number from the 2D database (this value may be negative)

d. Enter Object Number that is to be converted to 3D.

e. Enter number of nodes (4 for tet and 8 for brick elements)

f. Enter number of 3D planes to be created (this defines the level of discretization). 3D elements are created by sweeping 2D plane about a common axis to create wedges

g. Enter sweep angle (for axisymetric 2D simulation) or extrusion length (for plane strain 2D simulation)

h. Enter node centerline tolerance (entering 0.001 seems to work for most cases) Used to decide which nodes are centerline nodes - measurement should be made in 2D pre or post processor.

i. Select variables to copy onto 3D mesh (enter c to copy all).
Defaults should be appropriate in most cases.

Appendix F: Fracture with Element Deletion and Damage Softening

Fracture within DEFORM-3D is now available. To implement this, only a few settings are required. The first setting that is required is the critical damage value for fracture. This is specified within the material properties window -> Advanced tab (See Figure F.1). Within this window the damage criteria can be specified. By clicking the data window icon next to the criteria, a critical value can be input to the system (See Figure F.2). The critical value to use is very dependent on the material being used, the processing methods to produce the material, deformation history, etc…. The recommended way in which to use the critical value is to either determine the absolute critical value for fracture based on a known process or to reduce the damage value of a given simulated process.

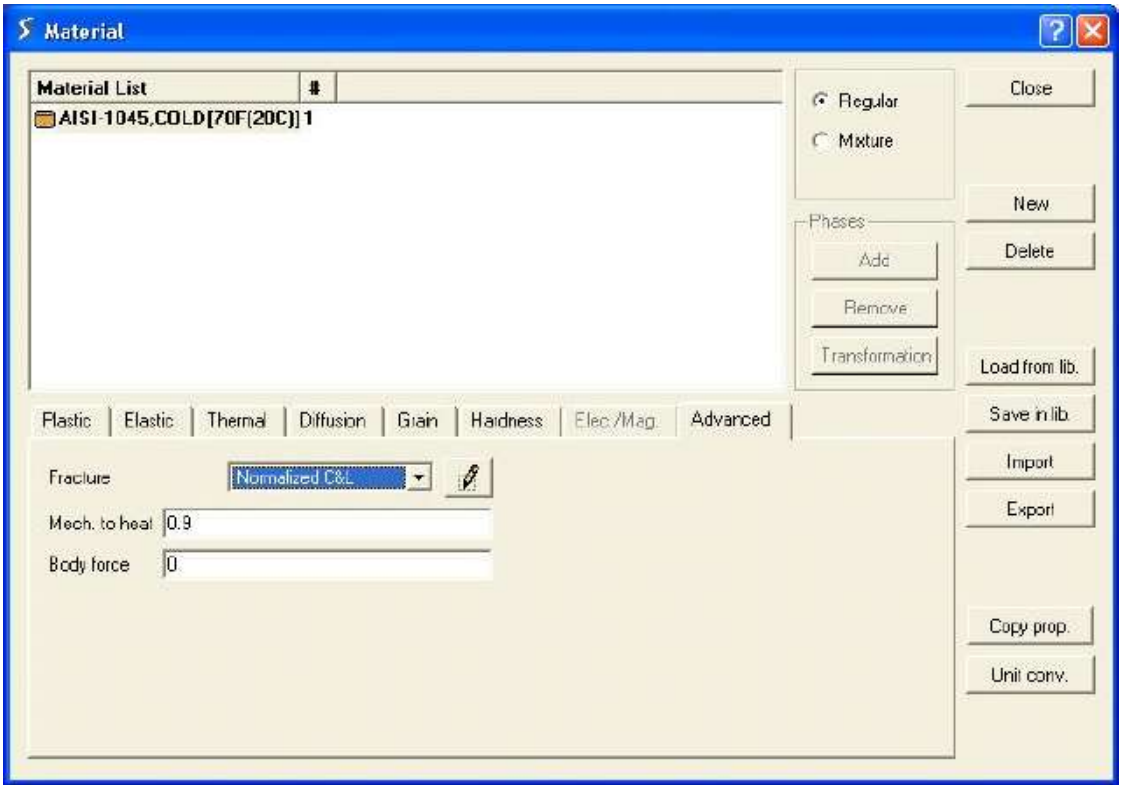


Figure F.1: The advanced material window.

To implement only a critical damage value will enable damage softening. Damage softening is a method by which the flow stress of an element above this critical value will by reduced to a very low value. The advantage of this approach above element deletion is that the topology of part is maintained and is simple

thus producing a good-looking result. To enable element deletion proceed to the object->properties and set the number fracture elements (See Figure).

Comparison of two methods:

The two methods for simulating fracture are compared in the following example. The example case is a gear piercing as seen in Figure F.4 , Figure F.5, Figure F.6 and Figure F.7 show the results of using both damage softening and element deletion.

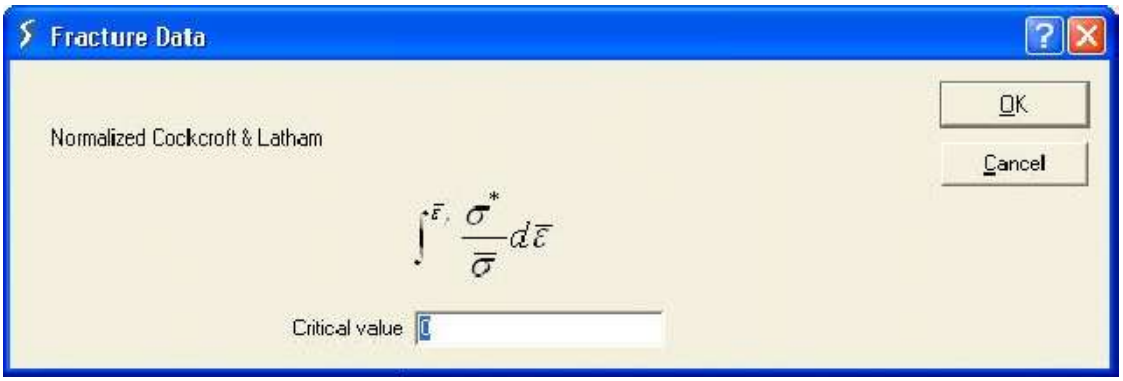


Figure F.2 : The critical value for fracture.

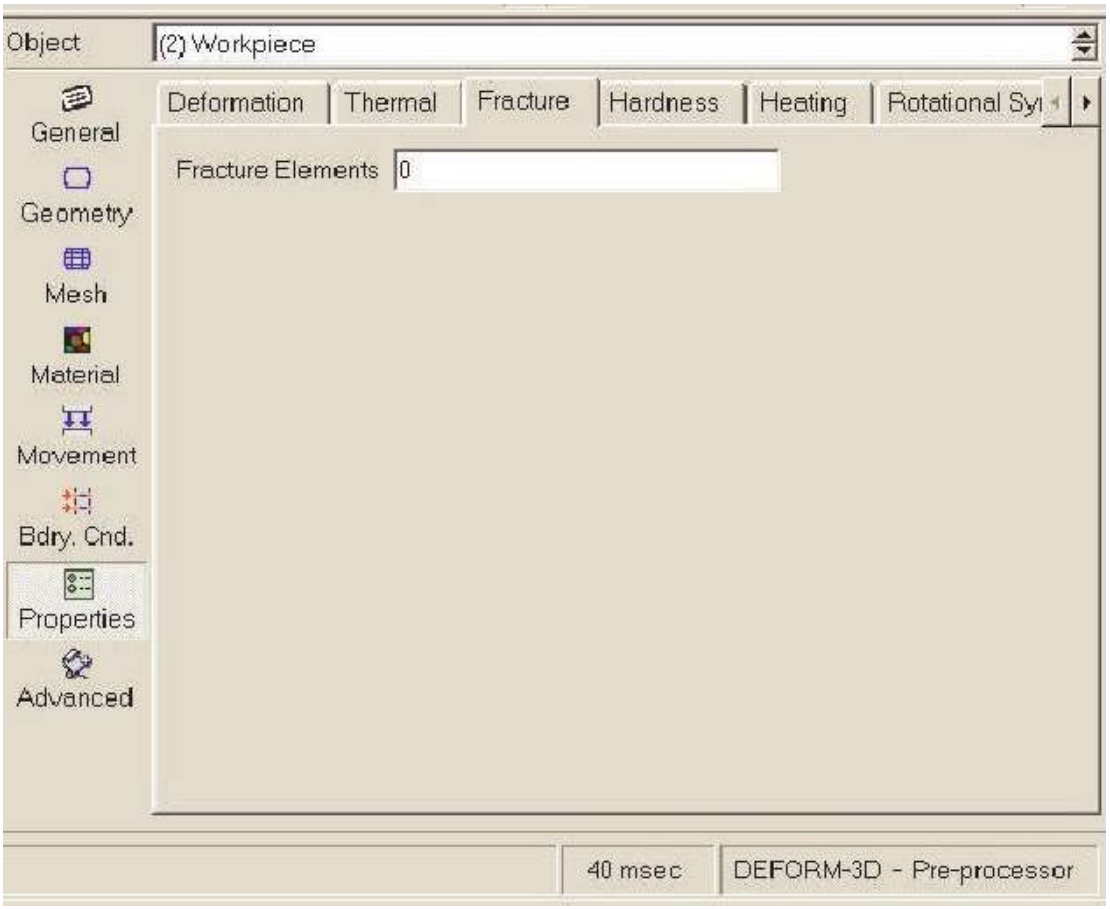


Figure F.3: Fracture settings window.

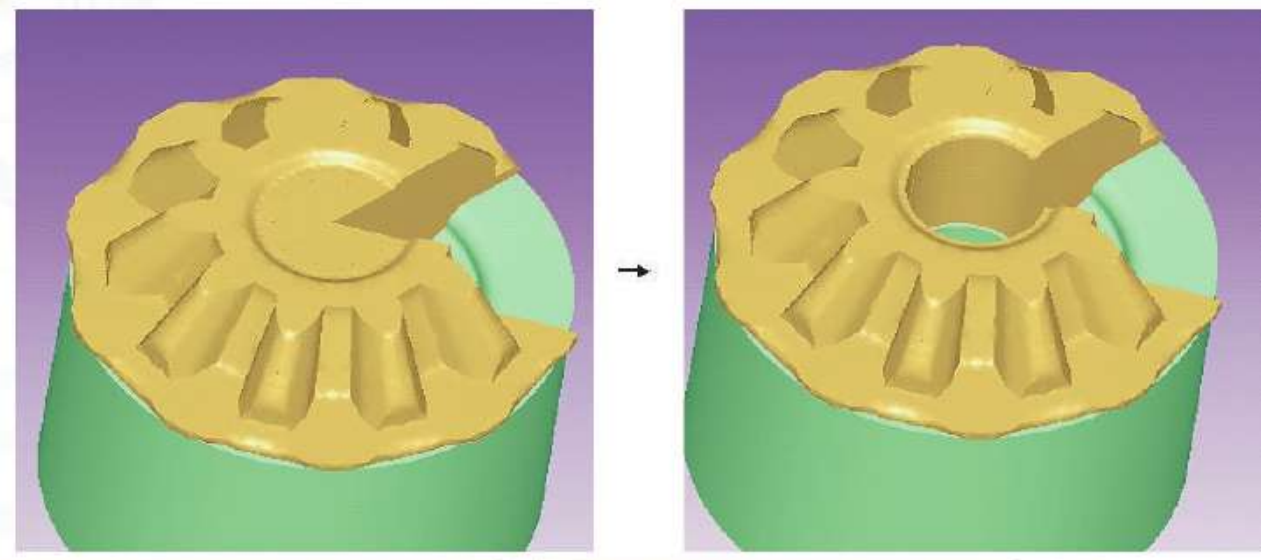


Figure F.4 : Gear piercing case that is a good candidate for fracture study.

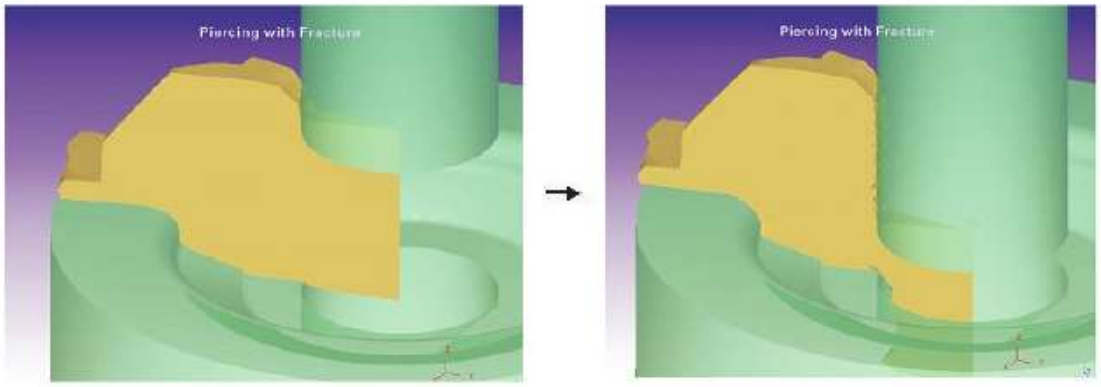


Figure F.5: Beginning and near ending step of gear piercing with element deletion.

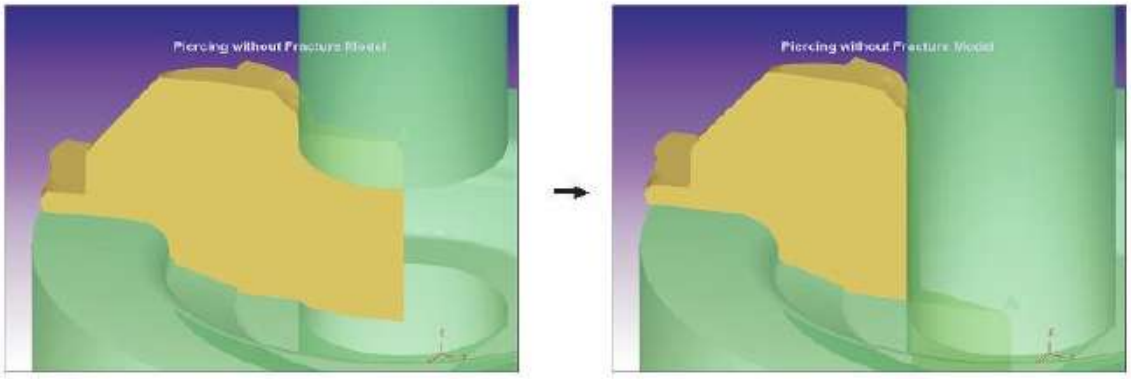


Figure F.6 : Beginning and near ending step of gear piercing with damage softening.

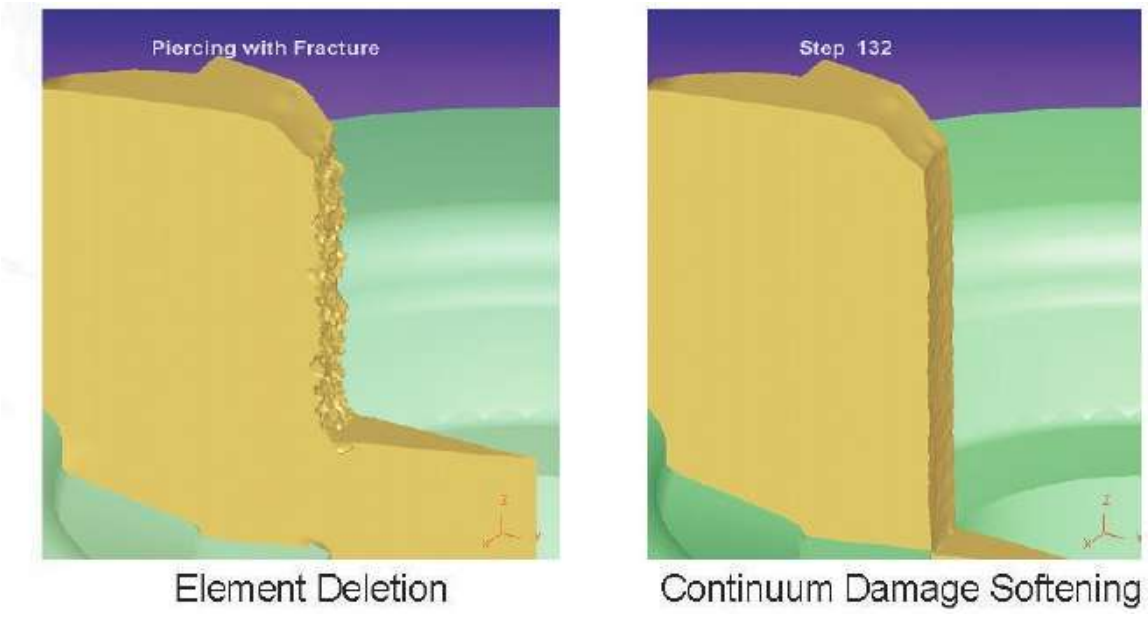


Figure F.7 : Side-by-side comparison of piercing operations with element deletion and damage softening.

Appendix G: Rotating Work piece Simulations

In this, special techniques for spinning work piece simulations are discussed. Among the applications that this would cover would be cross-rolling simulations (See Figure G.1).

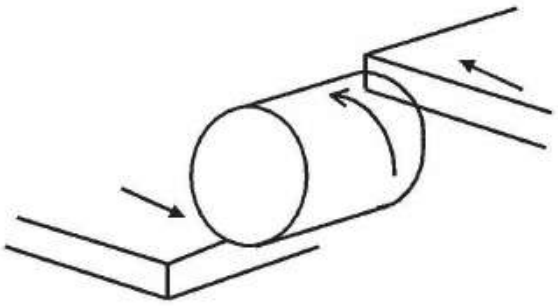


Figure G.1 : Cross-rolling diagram.

In the above case, there is a problem when the work piece rotates. The problem occurs due to the nature of updating nodal position based on integrating velocity over a time increment. The simple process of updating based on instantaneous velocity over a discrete time interval can cause an increase of the diameter of the work piece. As seen in Figure G.2 , all the nodal velocities are perpendicular to the radius where they are located. Thus, simply updating the coordinates based directly on their velocity will incur an increase in radius and in volume as well.

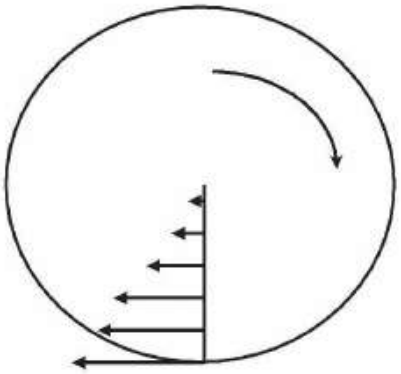


Figure G.2 : Velocity profile of a rotating body.

Another issue that may arise in simulations where the work piece is turning based on friction at very localized regions of the surface (particularly thread-rolling cases). The work piece may tend to slide rather than be rotated. This arises due to sparse contact that can occur between the tool and the work piece. The sparse contact arises when the work piece has a coarse mesh definition or when the tool geometry is coarsely defined.

Currently, there exists a number of different solutions for handling these types of problems. These solutions are listed below.

G1. Moving tools about a non-spinning work piece

Motivation: Although in the reality the work piece rotates during deformation, it may be advantageous to not allow the work piece to rotate and let the tools move about it. This will not change the nature of deformation, but has the following advantages: (1) Since the work piece does not spin, the increase in volume is avoided; and (2) flow-net can be used in post-processing.

Cartesian coordinates vs. cylindrical coordinates: For most of the cases, it is straightforward to specify the orbiting movement in the Cartesian coordinate system for a rotating tool, which orbits like a planetary gear. The user has to define the first and second rotations. However, when there is a tool translation that is not parallel to the second rotational axis, it is convenient to use a cylindrical coordinate system for the tool. We will only elaborate the latter case in this section.

Examples: (1) spinning with a roller, if the user wants to fix the work piece and to rotate the roller (Figure G.3) , and (2) thread rolling between two flat dies, if the user wants to fix the work piece and to rotate the two flat dies (Figure G.1).

How to Implement: If the following conditions are all met, DEFORM-3D will adopt the cylindrical coordinate for that object:

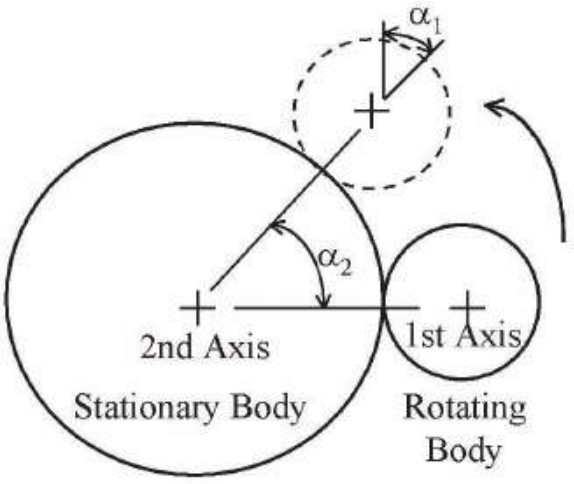


Figure G.3 : Description of rotational axis definitions and angle definitions (derived from angular velocity values) for this case.

1. The first rotational axis and the second rotational axis are defined and they are apart from each other.

2. The translational movement is non-zero in the direction non-parallel to the second rotational axis.

These values are defined in the Rotational Movement window. As seen in Figure G.3 , the first axis is the axis of the rotating tool and the second axis superimposes with the axis for the non-rotating work piece. The first rotational axis defines the rotational properties of the tool about it's own axis. If the tool does not spin about its own axis, as in a cross-rolling simulation, the axis center should be specified far from the work piece axis. The second rotational axis defines the rotational properties of tool about the axis of the work piece.

(For Example 2, the user needs to define the first rotational axis far away, say, 1.e6, but it is not used in calculation. In the DATA directory of DEFORM3D is an example file known as CROSS_ROLL.KEY.

This is a simple cross-rolling example showing an example of how the tools can move about the work piece in order to simulate cross-rolling without rotating the work piece).

As a note:

- The initial position of this object is always used as a reference. The "Current Angle" in the Rotational Movement window should be zero at Step –1 and will be updated by the system at the end of every step or sub-step. The user should not change its value in a later step in the pre-processor without changing the position of this object accordingly.
- The direction of the translational movement of this object is defined with respect to this reference position only. It will not be changed in a later step even the object rotates about the second rotational axis. The stroke will be updated in the same way.

In the case of two rotational axes, when the axes are parallel, the angular velocities are defined as follows (seen in Figure G.4):

$\omega_2 = (r_1/r_2)\ \omega_1$

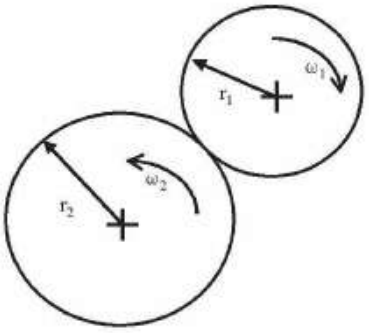


Figure G.4 : Two rotating bodies with parallel axes.

Otherwise, when the axes are at an angle to each other, such as in the case of orbital forming, the angular velocities are defined as (seen in Figure G.5):

$\omega_2 = -\ \omega_1 \cos \alpha$

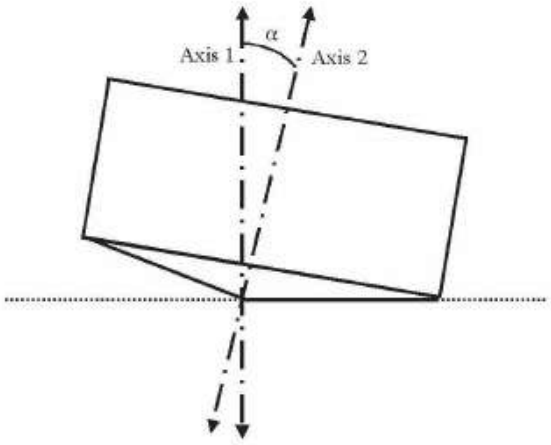


Figure G.5 : A rotating body with two non-parallel axes.

G2. Spinning Work piece

There are some features used to model the deformation of a rotating work piece with DEFORM-3D. They are under testing and have yet been officially added to DEFORM-3D. However, the user may activate these features when necessary by defining a data file "AXIS.DAT" in the working directory of a simulation. The options and contents of AXIS.DAT are explained as follows. This functionality works for a single rigid-plastic object and rigid tools only.

Here is the outline of AXIS.DAT file structure as described on a line-by-line basis. Each line is data that define how this feature will work for the current simulation. Once this file is created and placed in the current directory that is running a simulation, it will be read by DEFORM and applied to the simulation. **Caution: When finished using this file in a simulation, be careful to not run another simulation that does not require it as DEFORM will use this and may cause an errant simulation. Either rename or delete the file before running another simulation in the same directory.**

There are two functionalities that are available in this feature: Coordinate updating based on rotational motion and enforced rotational motion of the work piece. These two features (modes) can be enabled either separately or simultaneously depending on the mode set in the AXIS.DAT file. The rest of the data defines certain options on how these modes apply to the current case. Here is a line by-by-line description of the file.

Line 1: KOBJAX - Object number (an integer)

Line 2: Mode – An integer value that determines which function this feature should use.

Line 3: RAXIS(1),RAXIS(2),RAXIS(3) (3 real numbers)
Direction vector parallel to the axis of rotation. These components are unitless.

Line 4: ORGN(1),ORGN(2),ORGN(3) (3 real numbers)
A point that lies on the axis of rotation. This point can be any point on the central axis. The units for each component is mm for SI simulations and inches for the english unit system.

Line 5: RADCTR,OMECTR (2 real numbers)
RADCTR is the radius of a specified central core about the rotational axis for which rotational speed is fixed to the rotational speed of OMECTR. The units of RADCTR is mm for SI simulations and inches for the english unit system. The units of OMECTR is rad/s for SI and the english unit system.

Line 6: XMIN, XMAX (2 real numbers; optional)
RADCTR, XMIN and XMAX define the dimensions of a cylinder within which the nodes of object KOBJAX are forced to rotate about RAXIS at a rotational velocity OMECTR. (See Figure).

Line 7: XMIN2, XMAX2 (2 real numbers; optional)
XMIN2 and XMAX2, if available, define a second cylinder with an infinite radius within which the nodes of object KOBJAX are forced to rotate about RAXIS, but the magnitude of the nodal velocity is the result of simulation.

Notes on Line 2

Mode = 1 Enforces rotational update of nodal coordinates (This solves the problem described above with the volume increase).

Mode = 3 Part of object KOBJAX (defined below) is forced to spin about an axis (defined by RAXIS) in addition to enforcement of rotational updating.

Mode = 5 Part of object KOBJAX (defined below) is forced to spin about an axis (defined by RAXIS) in addition to enforcement of rotational updating, but the consolidation technique is applied.

Notes on Line 5

If OMECTR = 0, the nodal updating direction is specified as rotational, while the magnitude of each node velocity is the result of simulation (i.e. rigid tool(s) will control the speed of the nodes).

If OMECTR != 0, XMIN and XMAX (discussed below) are defined as the minimum and maximum bounds with respect to the axis and the origin defined on Lines 2 and 3. So are XMIN2 and XMAX2, if any. (Figure G.6)

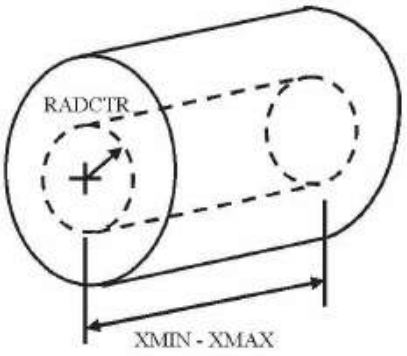


Figure G.6 : Outline of dimensions for central core.

Notes on Line 6

XMIN and XMAX, if available, are the axial bounds of the central core with the respect to ORGN.

If Line 6 is not defined, the cylinder has an unlimited length and Line 7 is not needed.

(Lines 5,6 and 7 are used only if Mode 3 is selected)

Option = 5 allows for the user to specify two independent cores that can drive spinning.

Only new inputs are explained.
Line 1: KOBJAX - Object number (an integer)

Line 2: Option (an integer)
Option = 5 Part of object KOBJAX (defined below) is forced to spin about an axis (defined by RAXIS) PLUS Option 1, but the consolidation techinique is applied.
Line 3: RAXIS(1),RAXIS(2),RAXIS(3) (3 real numbers)
Direction vector defining the axis of rotation.
Line 4: ORGN(1),ORGN(2),ORGN(3) (3 real numbers)
Origin of the above axis.
Line 5: NUMSEC,ISECPL (2 integers)
NUMSEC = 1 or 2 -- How many rigid zones to be specified
ISECPL = 0: if NUMSEC = 1, nothing is implied
NUMSEC = 2, two zones are not coupled
ISECPL = 1: if NUMSEC = 2, two zones are coupled
INPUT FOR SECTION 1:
Line 6: RADCTR, OMECTR (2 real numbers)
PLEASE NOTE: The meaning of OMECTR is different than that in Option 3:
If OMECTR is set to 1.e+12, the rotating direction is specified, while the magnitude of each nodal velocity is the result of simulation.
If OMECTR is set to 0, the part of work piece is fixed.

Line 7: XMIN, XMAX , VXMIN, VXMAX (4 real numbers; optional)
VXMIN is the speed of the left bounding point, XMIN
VXMAX is the speed of the right bounding point, XMAX
INPUT FOR SECTION 2, IF NUMSEC=2:
Line 6: RADCTR, OMECTR (2 real numbers)
PLEASE NOTE: The meaning of OMECTR is different than that in Option 3:
If OMECTR is set to 1.e+12, the rotating direction is specified, while the magnitude of each nodal velocity is the result of simulation.
If OMECTR is set to 0, the part of work piece is fixed.

Line 7: XMIN, XMAX , VXMIN, VXMAX (4 real numbers; optional)
VXMIN is the speed of the left bounding point, XMIN
VXMAX is the speed of the right bounding point, XMAX

Note: *When using a core region, the user should be cautious not to regard the stress or strain within the core region as significant. This core region should be far from the deformation area and in the case of simulations where there is interest in the material at the central region of the spinning object, this method cannot be used. Also, in order for the AXIS.DAT file to work properly in the latest version of DEFORM-3D, a file named DEF_RSE.DAT containing a single 0 should also exist in the working directory.*

Appendix H: Sheet Forming in DEFORM-3D

Due to advantages in modeling thin structures, the membrane or shell element formulations are very popular in the simulation of sheet forming processes. Although shell elements represent the stress variation through their thickness effectively, they generally require special treatments for the drilling degree of freedom and the transverse shear locking to preserve the Kirchhoff or Reissner-Mindlin hypotheses. Thus, the shell formulation requires more complicated and sophisticated procedures than solid element formulations. Moreover, shell elements do not have the continuity of the thickness over the neighborhood elements. A comprehensive comparison of solid and shell elements can be found in the reference (Wriggers et. al. [1]). In the reference, the authors showed the possibility of the application of solid elements for thin shell as well as thick shell problems.

A brief coverage of the theory of anisotropy and assumed strain formulation will be presented in the following sections. After this, specific information will be provided on how to simulate accurate sheet forming applications within DEFORM-3D.

Theory - Anisotropy

The associated flow rule with Hill'48 anisotropic yield criterion (Hill [2]) is used for consideration of initial texture property of sheet metal. The flow potential for orthotropy which conserves three symmetry planes are written in terms of the stress $\boldsymbol{\sigma}$ as,

$$f = \frac{1}{2}\boldsymbol{\sigma}^{T}\mathbf{P}\boldsymbol{\sigma} - \left(\sigma^{o}\right)^{2} = 0. \qquad (1)$$

with

$$\mathbf{P} = 2\begin{vmatrix} \beta_{11} & -\beta_{12} & -\beta_{13} & 0 & 0 & 0 \\ -\beta_{12} & \beta_{22} & -\beta_{23} & 0 & 0 & 0 \\ -\beta_{13} & -\beta_{23} & \beta_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & \beta_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & \beta_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & \beta_{66} \end{vmatrix} \qquad (2)$$

where $\boldsymbol{\sigma}^{T} = \{\sigma_{xx},\sigma_{yy},\sigma_{zz},\sigma_{xy},\sigma_{yz},\sigma_{xz}\}$ and $\beta_{11} = \beta_{12} + \beta_{13}$; $\beta_{22} = \beta_{12} + \beta_{23}$; $\beta_{33} = \beta_{13} + \beta_{23}$ (or $2\beta_{12} = \beta_{11} + \beta_{22} - \beta_{33}; 2\beta_{13} = \beta_{11} - \beta_{22} + \beta_{33}; 2\beta_{23} = -\beta_{11} + \beta_{22} + \beta_{33}$). Therefore, six independent parameters $\beta_{11},\beta_{22},\beta_{33},\beta_{44},\beta_{55},\beta_{66}$ need to be defined to characterize the anisotropic hardening state. σ^{o} is an equivalent stress

representing the current yield surface size. The coefficients in **P** can be related to the R-values (Valliappan *et al.* [3]). By setting $\beta_{11}=1$(this means the principal anisotropic axis coincides the reference axis),

$$\beta_{12}=\frac{R_0}{1+R_0};\beta_{13}=\frac{1}{1+R_0};\beta_{23}=\frac{R_0}{R_{90}(1+R_0)};\beta_{44}=\frac{(R_0+R_{90})(1+2R_{45})}{R_{90}(1+R_0)}. \quad (3)$$

The remaining parameters, β_{55},β_{66}, can not be determined by the uniaxial tensile test. Generally the corresponding stresses have little effect on sheet metal forming processes, the parameters are assumed to be equal to β_{44}. It should be noted that von-Mises isotropic yield criterion is recovered when three R-values, R_0, R_{45} R_{90} are set to be 1. Numerical implementation of Hill'48 yield criterion is outlined below.

The additive decomposition of strain-rate into elastic and plastic parts is employed together with the normality rule,

$$\dot{\boldsymbol{\varepsilon}}=\dot{\boldsymbol{\varepsilon}}^{e}+\dot{\boldsymbol{\varepsilon}}^{p},$$
$$\dot{\boldsymbol{\sigma}}=\mathbf{C}\dot{\boldsymbol{\varepsilon}}^{e},$$
$$\dot{\boldsymbol{\varepsilon}}^{p}=\dot{\lambda}\left(\frac{\partial f}{\partial \boldsymbol{\sigma}}\right)=\dot{\lambda}\mathbf{a}. \quad (4)$$

where the superscripts e and p represent the elastic and plastic parts, respectively. **C** is the elasticity tensor, $\dot{\lambda}$ is the plastic strain-rate multiplier and **a** is the flow vector defined by

$$\mathbf{a}=\mathbf{P}\cdot\boldsymbol{\sigma}. \quad (5)$$

From Equations (4) and (5) with the consistency condition (6), the plastic strain-rate multiplier can be expressed as below:

$$\dot{f}=\frac{\partial f}{\partial \boldsymbol{\sigma}}:\dot{\boldsymbol{\sigma}}+\frac{\partial f}{\partial \sigma^{o}}\dot{\sigma}^{o}=0, \quad (6)$$

$$\dot{\lambda}=\frac{\mathbf{a}^{T}\mathbf{C}\dot{\boldsymbol{\varepsilon}}}{\mathbf{a}^{T}\mathbf{C}\mathbf{a}+A_{iso}}. \quad (7)$$

where $A_{iso}=2\sigma^{o}H\dot{\bar{\varepsilon}}^{p}$.

Finally, the rate form of the constitutive equation can be written as,

$$\dot{\boldsymbol{\sigma}}=\left(\mathbf{C}-\frac{\mathbf{C}\mathbf{a}\mathbf{a}^{T}\mathbf{C}}{\mathbf{a}^{T}\mathbf{C}\mathbf{a}+A_{iso}}\right)\dot{\boldsymbol{\varepsilon}}=\mathbf{C}^{ep}\dot{\boldsymbol{\varepsilon}}. \quad (8)$$

It should be noted that the element stiffness matrix is directly related to the tangent modulus $\mathbf{C}^{ep}$ evaluated at each integration point, which governs the convergence rate of the global iterative scheme. Thus the consistent tangent modulus is essential to keep the quadratic rate of convergence in the Newton-Raphson scheme (Simo and Taylor [4], Crisfield [5]).

Theory – Assumed Strain Formulation

A locking-free element is essential for the robustness of the finite element method. Several versions of the reduced integration method using hourglass control techniques have made remarkable progress on this issue (Belytschko and Bachrach [6]; Hughes [7]; Belytschko *et al.* [8]). Li[9] and Jetteur[10] proposed a strain field modification to avoid numerical instability. This paper was based on the method proposed by Li [9]. The essential equations for the strain field description can be written as follows.

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^{o} + \boldsymbol{\varepsilon}^{h}. \tag{9}$$

where $\boldsymbol{\varepsilon}^{o}$ and $\boldsymbol{\varepsilon}^{h}$ are the constant and the non-constant terms of the displacement gradient respectively. The non-constant terms can cause undesirable locking, volumetric locking or hourglassing. To avoid these undesirable effects, the modified normal strain part, Equation (11), is assumed to be the same as Equation (12) in Equation (10).

$$\boldsymbol{\varepsilon}^{h} = \boldsymbol{\varepsilon}^{n} + \boldsymbol{\varepsilon}^{s} \cong \boldsymbol{\varepsilon}^{d} + \boldsymbol{\varepsilon}^{s}. \tag{10}$$

where

$$\boldsymbol{\varepsilon}^{n} = \{u_{1,1}, u_{2,2}, u_{3,3}, 0, 0, 0\}, \tag{11}$$

$$\boldsymbol{\varepsilon}^{d} = \{\overline{u}_{1,1}, \overline{u}_{2,2}, \overline{u}_{3,3}, 0, 0, 0\}, \tag{12}$$

$$\overline{u}_{i,j} = u_{i,j} - \delta_{ij} u_{k,k}/3. \tag{13}$$

Here, the repeated index is used to denote the summation

Possible shear locking in thin-structure analysis can be resolved on the element level by adopting the the assumed transverse shear strain field (Sze and Yao [11]; Kinkel *et al.* [12]). The transverse shear strains are interpolated from the values evaluated at the mid-points of the element edges as below.

$$\begin{aligned} \widehat{\gamma}_{\varsigma\xi} &= \frac{1-\eta}{2}\gamma_{\varsigma\xi}\Big|_{\xi=0,\eta=-1} + \frac{1+\eta}{2}\gamma_{\varsigma\xi}\Big|_{\xi=0,\eta=+1}, \\ \widehat{\gamma}_{\eta\xi} &= \frac{1-\xi}{2}\gamma_{\eta\varsigma}\Big|_{\xi=-1,\eta=0} + \frac{1+\xi}{2}\gamma_{\eta\varsigma}\Big|_{\xi=+1,\eta=0}. \end{aligned} \tag{14}$$

Simulation Principles

Some new features that allow for improved modeling of sheet forming processes within DEFORM are anisotropy modeling and a new assumed strain formulation

for elasto-plastic models. When modeling a sheet forming process, the following setup is recommended:

- Brick, elasto-plastic elements with an assumed strain formulation
- Consider if anisotropy should be added for better material modeling

Example: Square cup drawing process

As an example case, consider a square cup drawing process (See Figure H.1). The blank is made of an aluminum alloy with the following properties:

- Young's Modulus: 70 GPa
- Poisson's ratio: 0.3
- $\overline{\sigma} = 570.40\,(0.01502 + \overline{\varepsilon}^{p})^{0.3469}$

The thickness of the blank is 0.81 mm and the area of the blank is 150 mm ^{2} x 150 mm^2. The blank holder force is 19.6 kN, the punch stroke is 40 mm and the coefficient friction is 0.162. The deformation of the drawing process can be seen in Figure H.2 and Figure H.3 . This simulation correlates well to experimental results as can be seen in Table 1. The deformed shape at 40mm punch stroke is shown in Figure H.3 and the amount of draw-in along the rolling (DX), transverse (DY), and diagonal (DD) directions is compared with the average values of the measurements in Table 1. Since an isotropic yield criterion was used for this simulation, the predicted draw-in along two directions, DX and DY are almost identical in the simulation. Numerical results are well correlated with the measurement results.

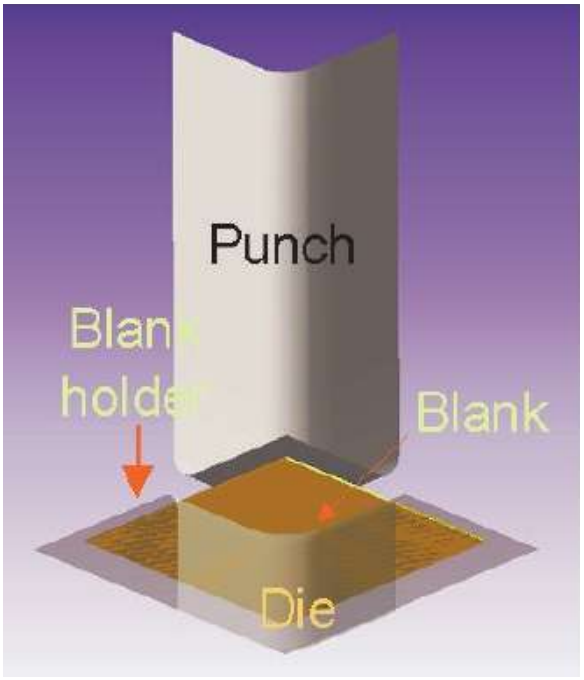


Figure H.1: Square cup drawing process.

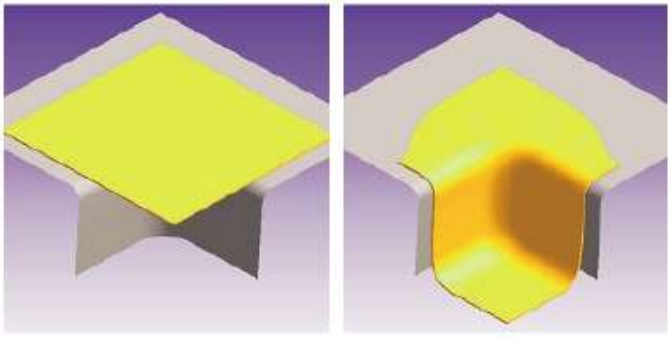


Figure H.2 : Square cup object (a) before deformation and (b) after deformation.

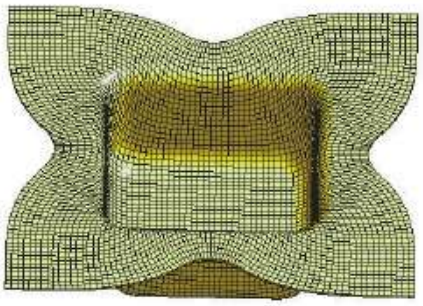


Figure H.3 : Deformed shape in square cup drawing (aluminum alloy sheet, at 40 mm punch stroke).

Measured point	Measured	Simulation
15* (DX,DY)	5.3 mm	5.5mm
15*(DD)	3.3 mm	3.3mm
40**(DX,DY)	28.5mm	26.9mm
40**(DD)	15.0mm	15.5 mm

Table 1: Draw-in distance. Note) * and ** denotes 15mm , 40mm punch stroke respectively

Example: Cylindrical cup drawing process

This benchmark problem was proposed for NUMISHEET'99 (Benchmark B1-part 2), designed to explore the anisotropic aspects of sheet metal forming processes, both from experiments and numerical simulations [13]. Part I, which is omitted in this paper, refers to a deep drawn cylindrical cup with a hemispherical punch free of any localized necking or split according to the actual individual forming-limit curve. Part 2 is simulated under given a constant blank holder condition.

The typical parameters for this simulation are summarized below:

- Blank thickness: 1.0 mm
- Drawing ratio: 2.0
- Constant BHF: 80 kN
- Drawing depth : 85 mm
- Material : DDQ (mild steel)
- R-values: $R_0 = 1.73; R_{45} = 1.23; R_{90} = 2.02$

The NUMISHEET'99 committee supplied tool geometries and material data for DDQ(mild steel). The FE model for this benchmark is shown in Figure H.4 . The work piece is an elasto-plastic material with the planar anisotropic yield criterion (Hill'48). The earing shapes can be obtained from planar anisotropic yield criterion and the corresponding punch travel and punch force are compared in Figure H.5 . The amount of draw-in along the rolling (DX), transverse (DY), and diagonal (DD) directions are compared with the measurements in Table 2. The measured data is average values of three participations (B1E-02, B1E-03 and B1E-04) in NUMESHEET'99 benchmark test.



Figure H.4 : FE model for cylindrical cup drawing.

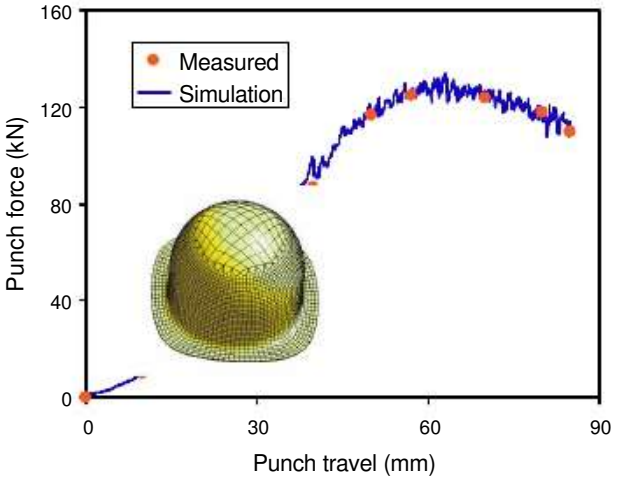


Figure H.5 : Punch force vs. stroke curve.

Measured point	Measured	Simulation
DX	29.0 mm	27.0 mm
DD	32.0 mm	35.0 mm
DY	27.5 mm	25.0 mm

Table 2: Draw-in distance.

References:

[1] Wriggers, P.,Eberlein, R., and REESE, S., 1996, A comparison of three-dimensional continuum and shell elements for finite plasticity, Int. J. Solids Str., 33(20-22), pp.3309-3326

[2] Hill, R., 1950, The Mathematical Theory of Plasticity, Oxford Univ. Press, London, Chapter 12

[3] Valliappan, S., Boonlaulohr, P., and Lee, I. K., 1976, Non-linear analysis for anisotropic materiala, Int J Num Meth Eng. 10, 597-606.

[4] Simo, J.C. and Taylor, R.L., 1985, Consistent tangent operators for rate independent elasto-plasticity, Comp. Meth. Appl. Mech. Eng., 48, pp. 101-118

[5] Crisfield,M.A., 1987, Consistent schemes for plasticity computation with the Newton-Raphson method, Computational plasticity, part I, pp. 133-159

[6] Belytschko, T. and Bachrach. W.E., 1986, Efficient implementation of quadrilaterals with high coarse-mesh accuracy, Comp. Meth. Appl. Mech. Eng., 54, pp. 279-301

[7] Hughes, T.J.R., 1980, Genralization of selective integration procedures to anisotropic and nonlinear media, Int. J. Num. Meth. Eng., 15, pp. 1413-1418

[8] Belytschko, T., Ong. J.S., Liu, W.K. and Kennedy, J.M., 1984, Hourglass control in linear and nonlinear problems, Comp. Meth. Appl. Mech. Eng., 43, pp. 251-276

[9] Kaiping Li, 1995, Contribution to the finite element simulation of three-dimensional sheet metal forming, Ph.D thesis, MSM, Universite de Liege, Belgique

[10] Jetteur, P., 1991, A mixed finite element for the analysis of large inelastic strains, Int. J. Num. Meth. Eng., 13, pp. 229-239

[11] Sze, KY, and Yao, LQ, 2000, A hybrid stress ANS solid-shell element and its generalization for smart structure modeling. Part I-solid-shell element formulation, International Journal for Numerical Methods in Engineering, 48, pp. 545-564

[12] Kinkel, S, Gruttmann, F, Wagner, W, 1999, A continuum based three-dimensional shell element for laminated structures, Computers and Structures, 71, pp. 43-62

[13] Gelin, J. C. and Picart, P., 1999, Proceedings of NUMISHEET'99 - The 4 th international conference and workshop on numerical simulation of 3D sheet forming processes, France, September 13-17.

Appendix I: Eulerian treatment of the 3D rolling process

Note: **From 3DV6.1, a dedicated shape rolling system is being released, to handle each of the above details in a very convenient way for the user. The new system, can generate the data required for both transient multi pass rolling, ALE type shape rolling models, and a ring rolling system with special ALE techniques developed for computational efficiency. System manuals and labs have been provided in the installation folders /MANUALS/pdf.**

Appendix J: Preventing leakage of nodes in sectioned simulations

In many cases, To prevent the leaking of nodes about symmetry planes, requires extra information so that the simulation engine knows the exact definition of the symmetric condition. This is done by two definitions:

1. The deforming body needs a symmetric plane definition on the cut surfaces;
2. The rigid geometry require a symmetric surface definition on their cut surfaces.

In the example used for this section, the spike simulation (Figure J.1) will be used to demonstrate this capability. Note that the die geometries and the work piece mesh are the same size. What follows is a step-by-step procedure that shows how this simulation is constructed in order to allow the geometry and mesh to coincide in size.
Note: If there is a difference in the size of the die versus the work piece in the symmetry surfaces, it is safer to error in making the dies larger.

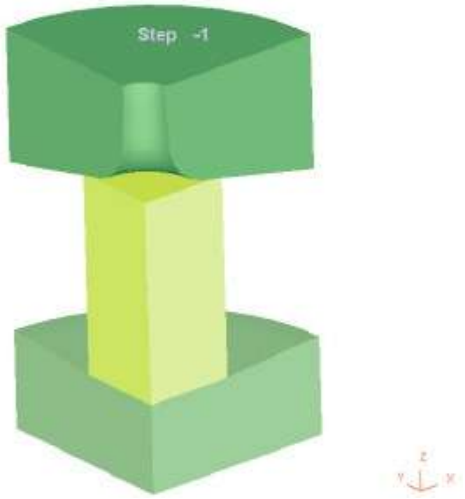


Figure J.1 : Spike problem being used as the example case.

Step 1: Define symmetric surface on work piece cut faces to allow for proper meshing.

On the work piece geometry, the cut faces should have symmetric surface defined prior to the meshing step (Figure J.2). This option is available from the geometry selection under the symmetric surface tab. This information allows the mesh generator to maintain a tight seam of nodes on the centerline.

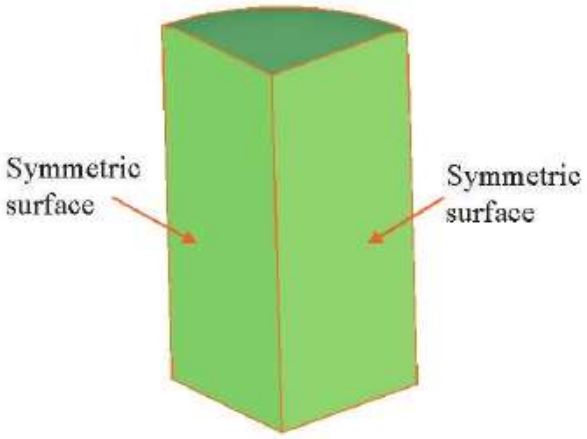


Figure J.2 : Adding symmetric surface to the part helps the part maintain the centerline during meshing.

Step 2: Define symmetry surface boundary condition on cut faces of work piece to maintain proper symmetry.

After the mesh has been generated for the work piece, the nodes on the symmetry surfaces must be given a boundary condition to hold them in the plane. This is done under the BCC window. Use the symmetry plane selection and define symmetry planes for both cut surfaces (Figure J.3). After this, the work piece boundary conditions are taken care of in terms of leakage.

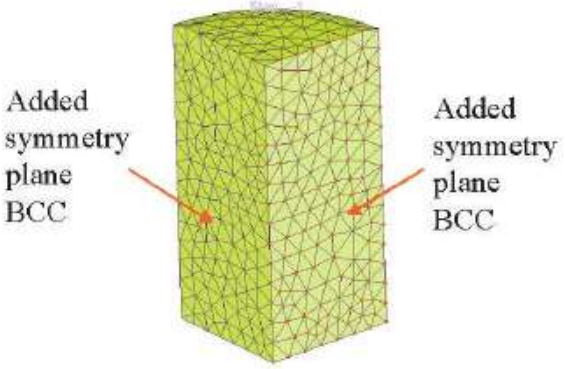


Figure J.3: Adding symmetry planes in the boundary condition window maintains the nodes in a planar condition.

Step 3: Add symmetric surface to die geometry cut surfaces.

Adding symmetric surface condition to the geometry of both the top die (Figure

J.4) and the bottom die completes the specification that allows the dies and work piece to be the same size.

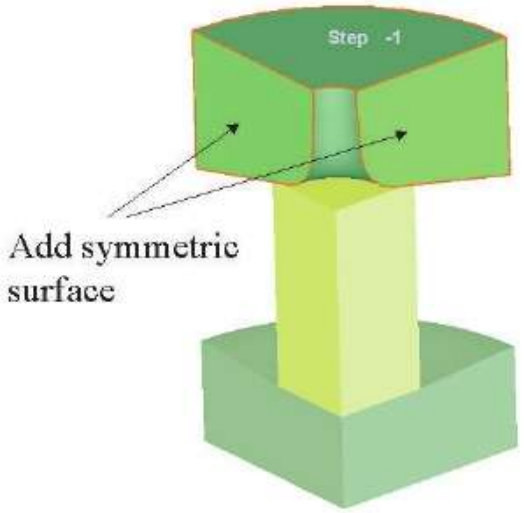


Figure J.4 : Adding a symmetric surface to the top die prevents any leakage from occurring.

Appendix K: The Double Concave Corner Constraint

This feature is available under the Simulation Controls -> Advanced menu in the preprocessor. Any given node in an FEM mesh has three degrees of freedom (DOF). In a cartesian coordinate system they can be the X, Y and Z directions. In a cylindrical system, they can be the radial, axial and hoop directions. In any case, no matter what coordinate system one selects, there are no more and no less than three degrees of freedom for any node. In the boundary condition dialog (as seen in Figure K.1), the DOF for the nodes are defined through contact, through velocity control and other conditions. The way in which a DOF is defined for a node in contact is to not allow the node to penetrate into the object as well as do not allow separation if the tensile separation criteria is not exceeded (usually a small nominal value). Three contact conditions, completely specify the motion of a given node.

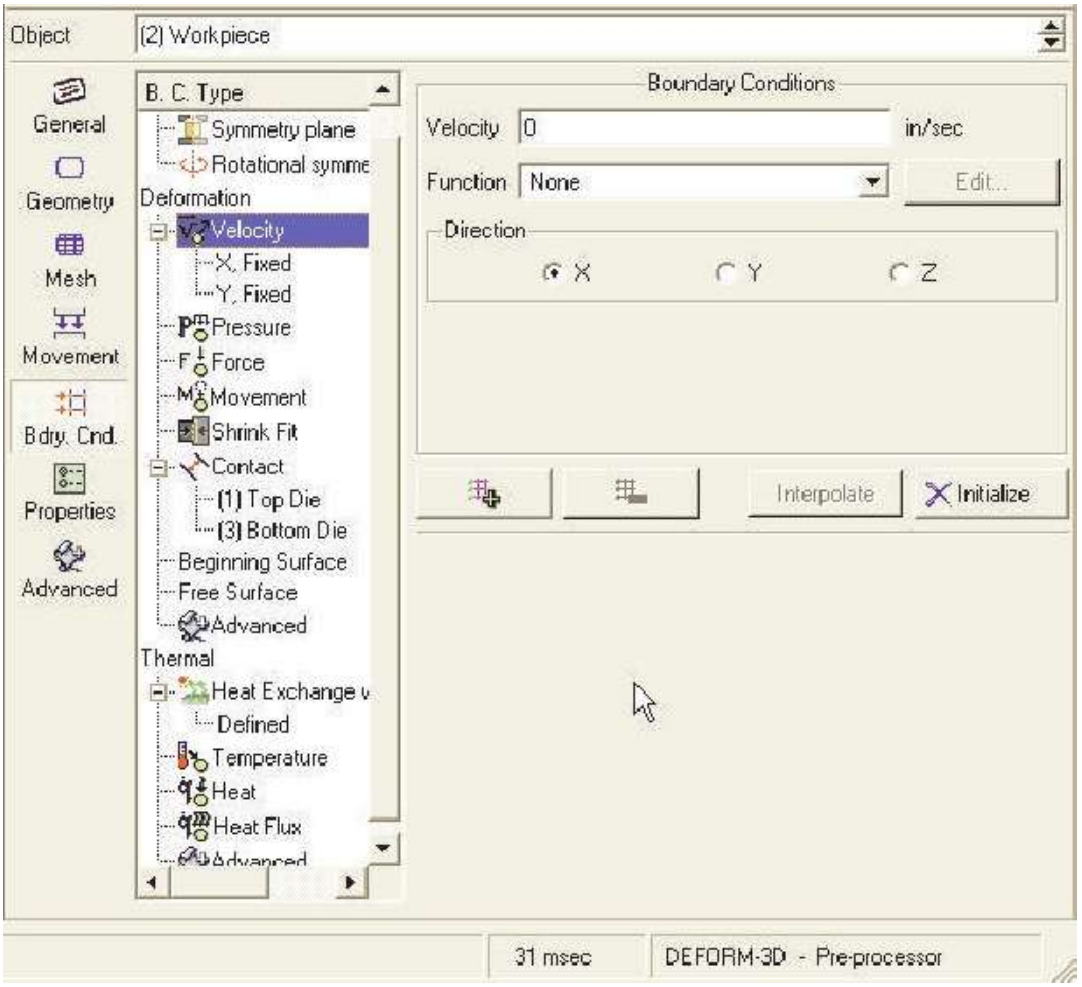


Figure K.1 : The boundary condition dialog.

There is a specific case where more than one degree of freedom is required for a given node. Consider the case where a node resides in the corner of a die cavity (as seen in Figure K.2). Note that nodes 1,2,3 are in contact with the die surface and their *vertica*l motion should follow the die surface. Note also that nodes 3,4,5 are in contact with the die surface and their *horizontal* motion should follow the die surface. The problem comes for node 3. It should have two boundary condition codes to restrict its motion in two DOF. These two degrees of freedom can be seen as the directions of the red arrows in Figure K.2 . However, if it only has one, it is only restricted in one direction and can thus penetrate the other direction.

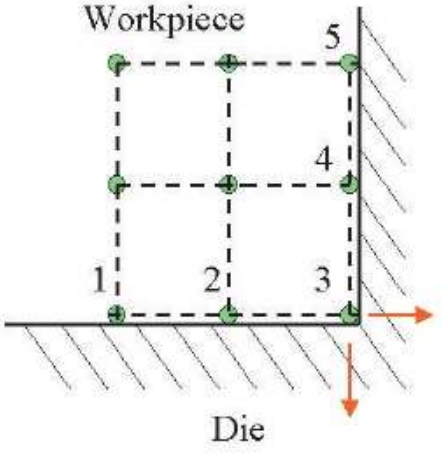


Figure K.2 : A set of nodes lying on a die surface. Note that nodes 1,2,3 should be constrained in the vertical direction and nodes 3,4,5 should be constrained in the horizontal direction.

For this reason, there is a new functionality to let nodes in convex corners be applied with 2 contact conditions. In order to specify which nodes should be given this constraint, two angles are to be given for this consideration. As seen in Figure K.3 , angle a is the minimum angle value and angle b is the maximum. Between these two angles, nodes will be specified with a double contact constraint. Figure K.4 indicates the corresponding settings in the simulation controls (Control Files : Category 1)

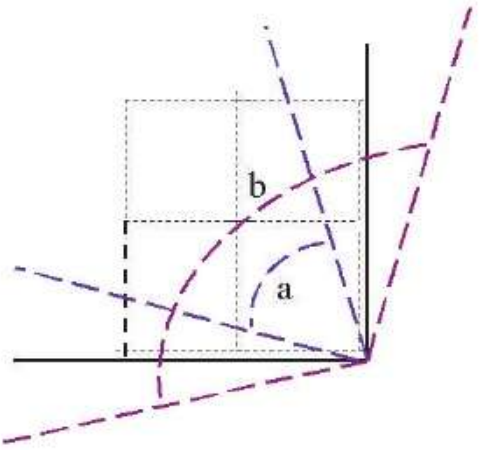


Figure K.3 : The two angles that are specified in the double concave corner constraint.

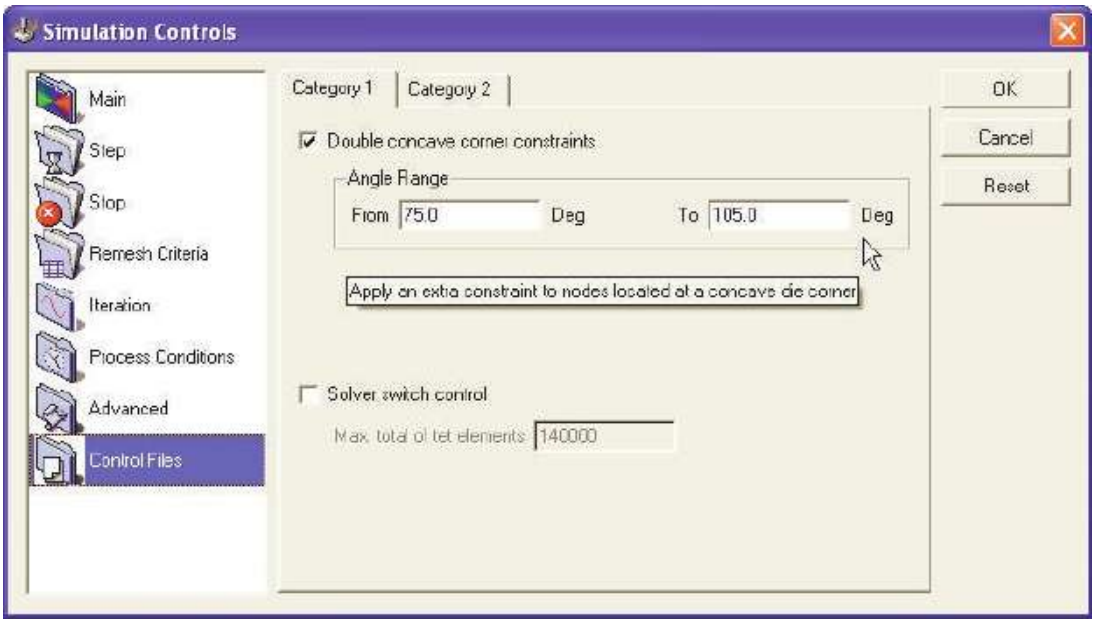


Figure K.4: The two angles that are specified in the double concave corner constraint.

Appendix L: Shape Rolling Simulation Overview

This appendix will cover a basic three-dimensional rolling case seen in Figure L.1

System features:

A listing of the system overview is as follows:

1. The work piece or rolling stock is object 1, rolls and other components are the following objects.
2. The rolling stock is of rigid-plastic type.
3. Model preview is available for interactive setup of multi pass conditions.
4. The rolls are rigid objects during for the rolling simulations.
5. Rigid rolls can handle non-isothermal conditions.
6. Meshing controls, and remesh procedures for brick elements.
7. The rolling direction is along global X-axis.
8. Side rolls can be defined with specific movement controls.
9. Support tables can be defined including thermal interaction with work piece.
10. Automatic stopping criteria for ALE and Lagrangian models
11. Inter pass thermal and strain variations can be modeled.

Characteristics:

Project based

The Shape Rolling Template is project based in which each simulation will be associated with a project directory. A project can consist of a single operation or contain multiple operations that occur on a single rolled stock. Each operation can be either a change in roll geometry, roll gap, roll speed, workpiece orientation or a heat transfer operation.

User interface

The interface is an innovative mixture of an open system and guided user interface. If the user desires, navigation can be sequential, via a list of menus to construct a simulation data; alternatively, the user can access menus in any sequence by selecting any item in a list. Running the simulation and Post-Processing the results is carried out via the standard DEFORM™-3D features.

Complete system manual and labs have been provided in the MANUALS/PDF folder of the installed system.

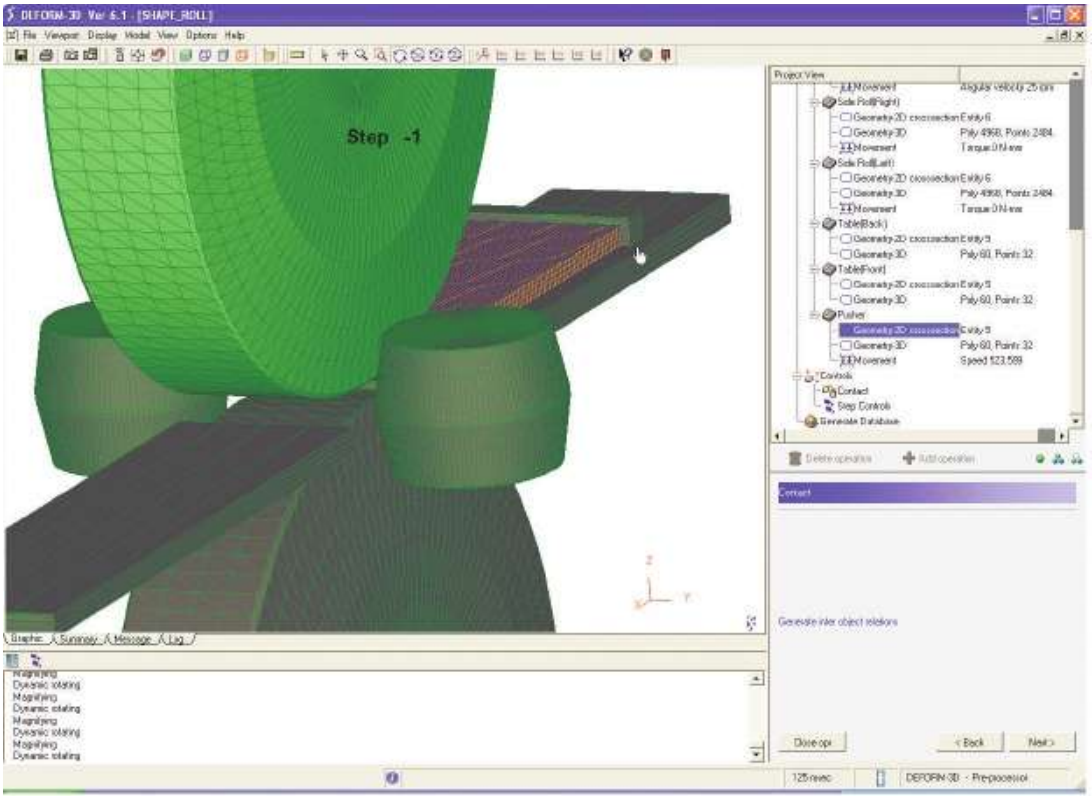


Figure L.1 : Simple 3D shape rolling model

Appendix M: Checking the forming loads results of a simulation

There are several factors that affect the forming loads and tool stresses of a simulation. This appendix will try to give the reader a cursory introduction into understanding what is required of a simulation in order to give accurate results. It is the presupposition of this document that DEFORM will yield an accurate result given that the inputs properly reflect the actual case being modeled. It has been verified many times that DEFORM is a leader in accurate results for the correct input.

The outline of this appendix is to first discuss some guidelines for obtaining proper load results. Since these loads are transmitted as forces onto the dies, it is imperative that these results be accurate in order for the stresses in the dies to be accurate.

Guideline 1: Check the flow stress data and make sure that it is representative of your actual stock.

This is a very obvious rule that sounds simple at first but tends to be overlooked very frequently. Some people perform testing on their material to make sure that the data they have matches the materials they are using. Often some data is meant for different processes or has had slightly different processing conditions or has a different chemistry. If testing is not an option, often one can try to correlate load results over several simulated processes and try to determine the suitability of material data.

Guideline 2: Make sure that the material data covers the process condition range.

The required material data for a simulation can be only flow stress data for a rigid-plastic material at isothermal conditions. In the case of a non-isothermal elasto-plastic simulation, elastic, plastic and thermal data should be specified. All the required data should be specified for the range of temperature, strain and strain rate that the process exists at. If any extrapolation occurs, the results can become inaccurate.

Guideline 3: Check that the mesh resolution of work piece is reasonable to capture the shape of the dies.

The number of required elements in a simulation can vary depending on the process and the desired results. In the case of a simple upset of a round bar, the deformation gradient is not large and the only region that can require a fine mesh is at the contact areas if a hot work piece is contacting a cold tool. However, in

the case of a forging of a complex shape such as a crankshaft, many elements are required to capture the many details of the final shape.

Guideline 4: Make sure that if the process is hot or warm that correct die speed is considered as well as time for the part to be transferred

In the case of hot forming and some warm forming cases, the materials tend to be sensitive to forming rate. In this case, the speed of the moving tool can impact the results greatly. The impact of the forming rate can be seen directly in the flow stress data. By checking how much the flow stress data changes at a given temperature based on the forming rate can show very large changes in the stress of the material (thus the forming load of the part) versus the forming rate. Also, in cases where a part is very hot, small periods of time between transfers can add up to a non-negligible amount of heat loss. This is important to consider since many materials can have their properties changes very quickly at hot temperatures.

Guideline 5: Check at the end of the simulation that the flash thickness is correct (or that the tool travel distance is correct)

This should be of no surprise to anyone who designs tools are works in the metal forming industry. As a part fills all the crevices of a die, the load will tend to increase rather quickly. If the simulation is overstroked or understroked, the results will behave just like real life. The results will tend to over or underestimate loads respectively.

Guideline 6: Make sure that the friction value is consistent with the actual process

In many processes such as a forward extrusion, the friction can contribute to the forming load of the process. DEFORM provides some recommended values within the interface but it is important that the user should take care to check whether these values are applicable to the process at hand.

Appendix N: Model set up for Steady state machining process from the DEFORM Pre-Processor.

(Keyword file available to the user: steady_state_machining.KEY,
Process type: Turning)

Objective:

- To predict steady state chip geometry
- To predict steady state thermal behavior

Procedure:

Here is a step-by-step instruction on how to perform this analysis

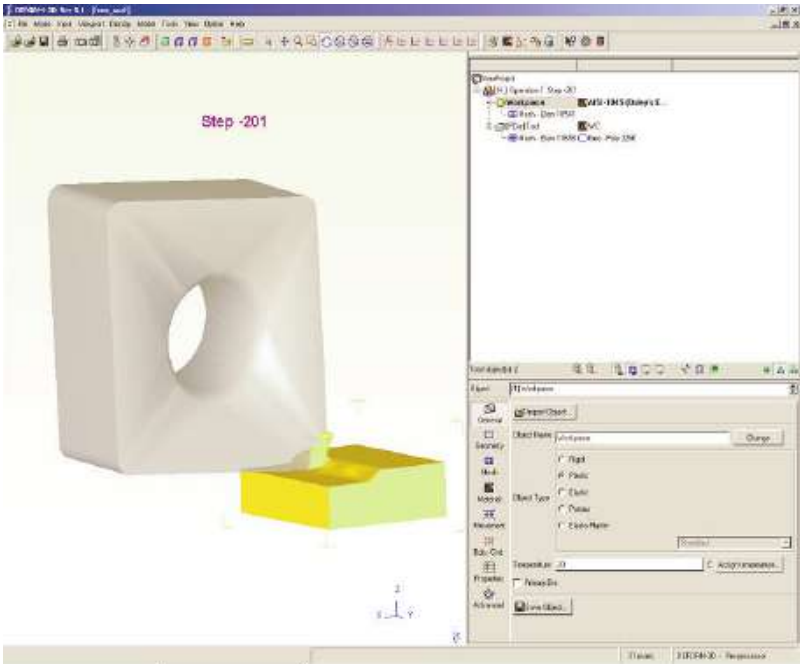


Figure 1: Result of lagrangian simulation of chip forming

Step 1

Load the machining database in Pre after sufficient chip has formed in the transient (Lagrangian) mode

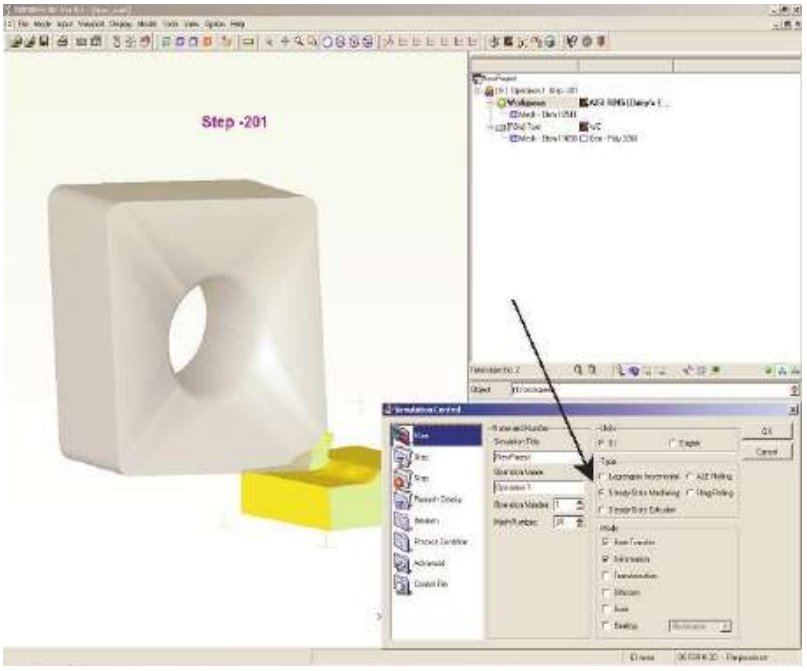


Figure 2: Setting the simulation type to steady state

Step 2

Set the analysis type to the Steady-State Machining mode in the simulation controls menu

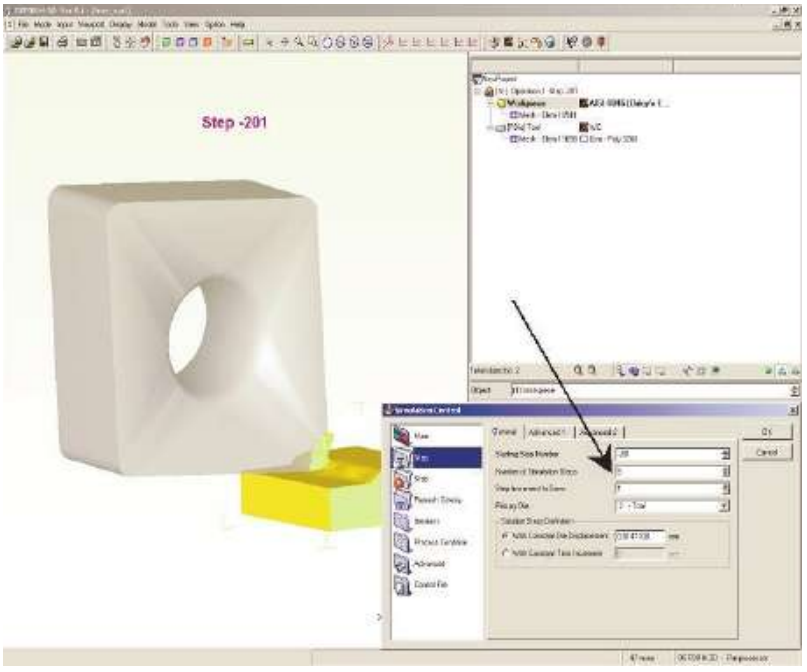


Figure 3: Set the number of steady state iterations

Step 3
Set the number of steady state iterations (Number of simulation steps)

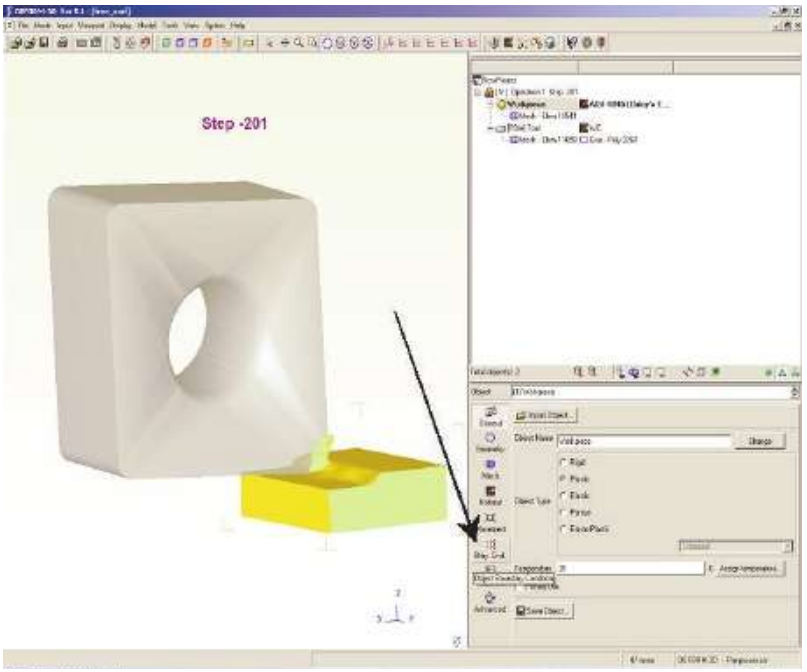


Figure 4: Entering the BCC menu

Step 4
Enter the BCC menu to define the free surface nodes on the chip

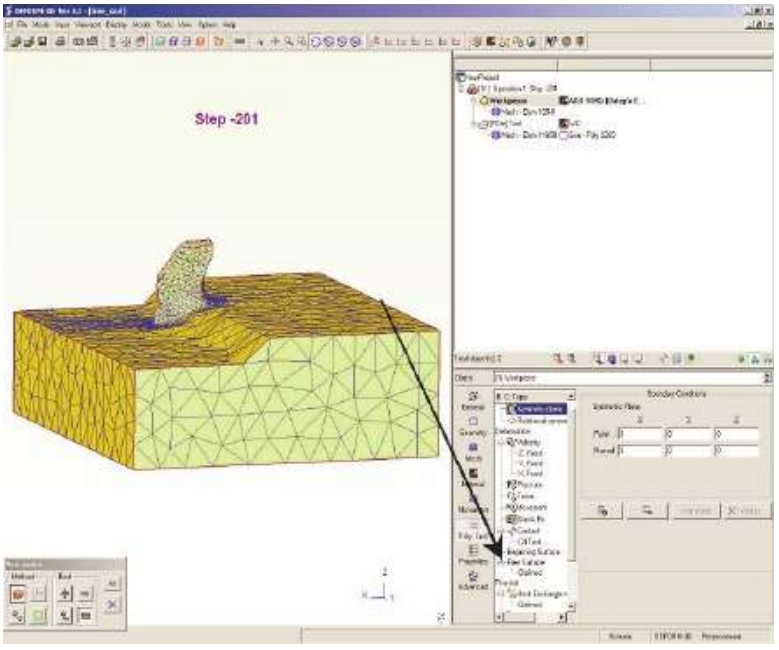


Figure 5: Entering the free surface BCC definition

Step 5
Enter the Free surface BCC definition menu

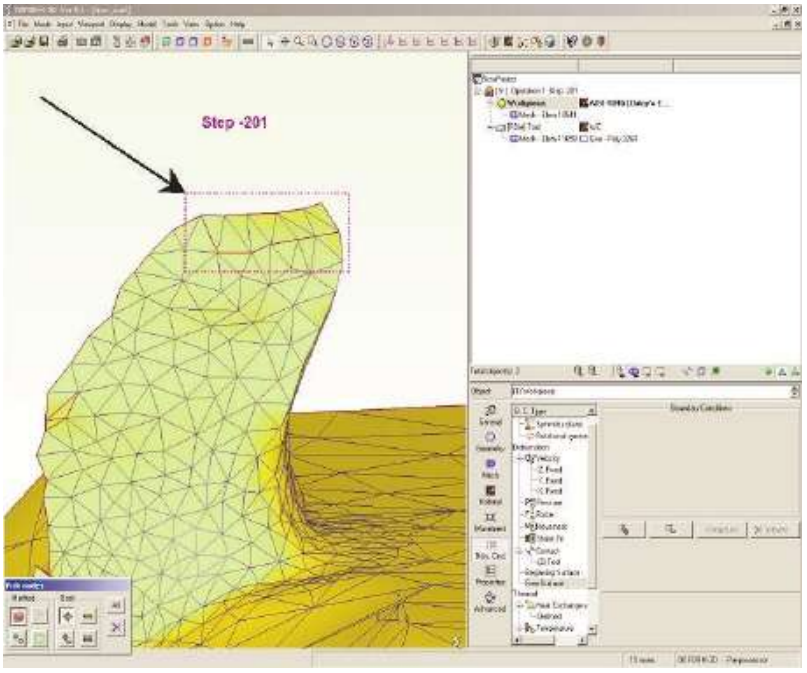


Figure 6: Zoomed in on end of chip

Step 6
- Zoom in to the chip end surface area
- Identify the end surface on the chip

Program treats this region as the material exit region and rest of the chip surface is corrected to follow the steady state velocity field

Please note that this region should be sufficiently away from the insert contact region, otherwise free surface correction predictions may not be accurate.

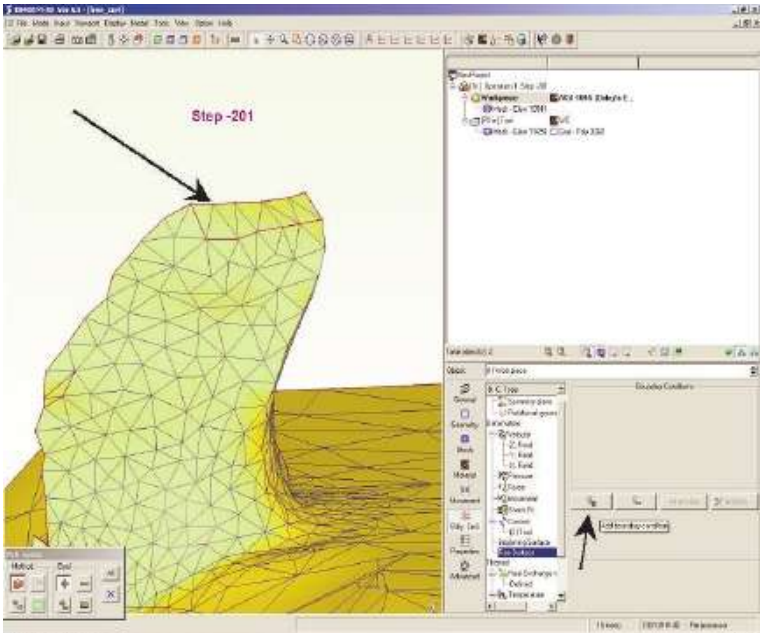


Figure 7: Selecting the end nodes

Step 7

Select the chip end surface nodes using the available options, and click on the '+' add icon to confirm the selection (see the red dots on the selected nodes)

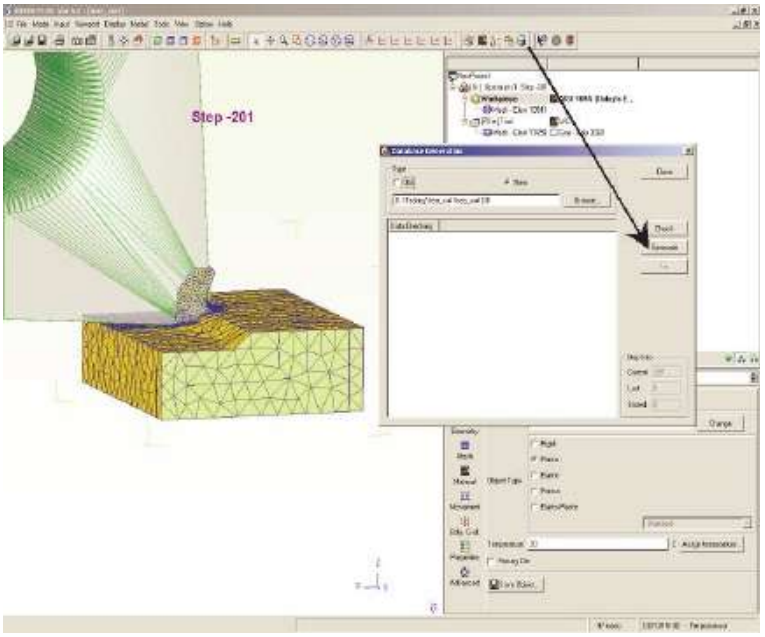


Figure 8: Write the database

Step 8

Write the database

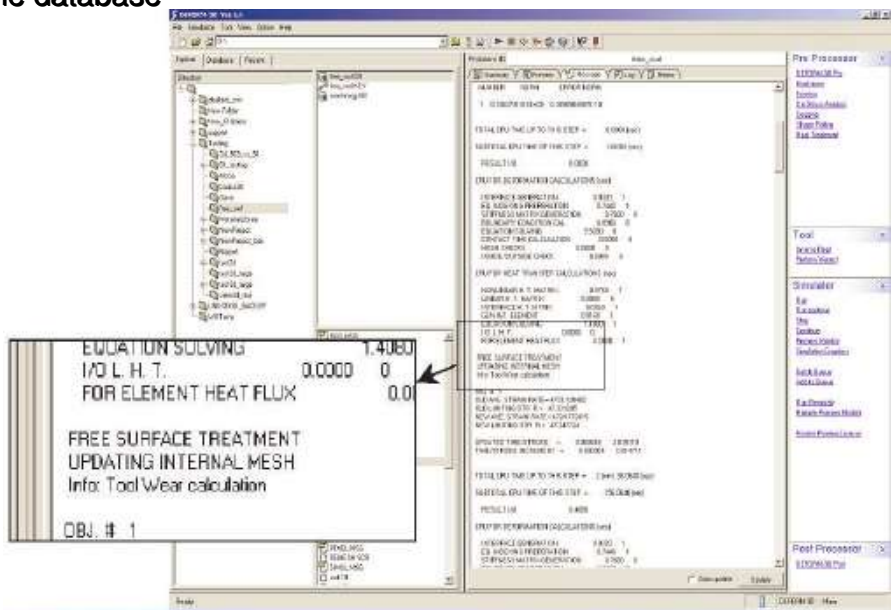


Figure 9: Running the simulation (Note the text in the message file)

Step 9

• Carry out the analysis

Watch for the messages in the message file as shown here,

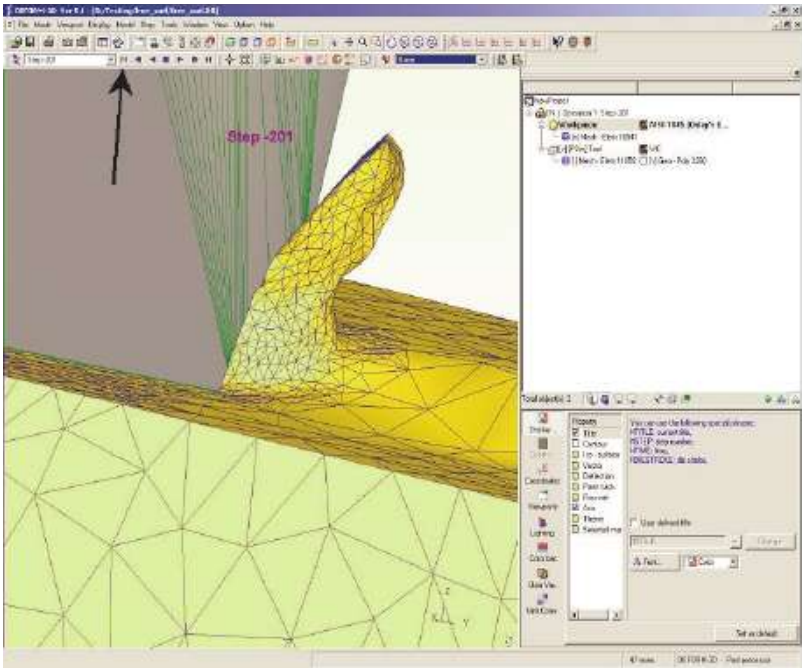


Figure 10: Loading the simulation in the post-processor

Step 10
Load the machining database in Post.

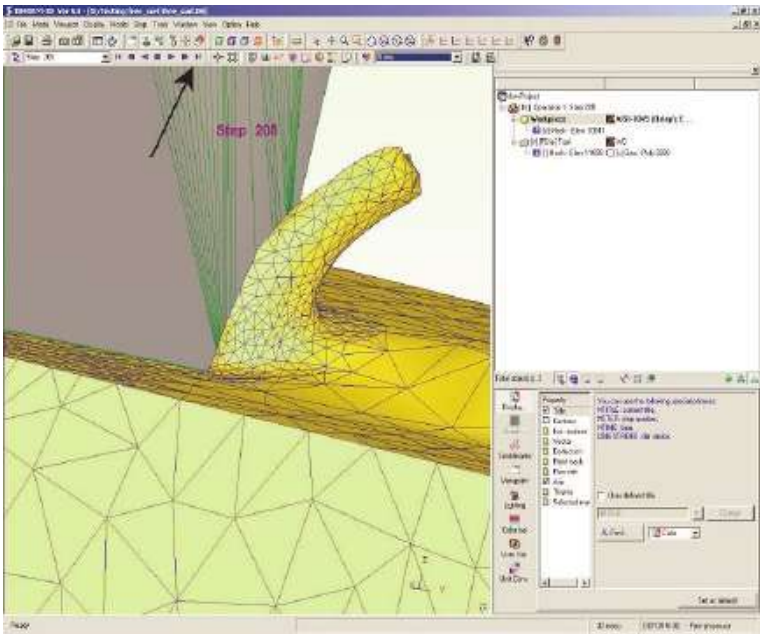


Figure 11: Check the corrected shape of the chip

Step 11
Check for the corrected chip shape

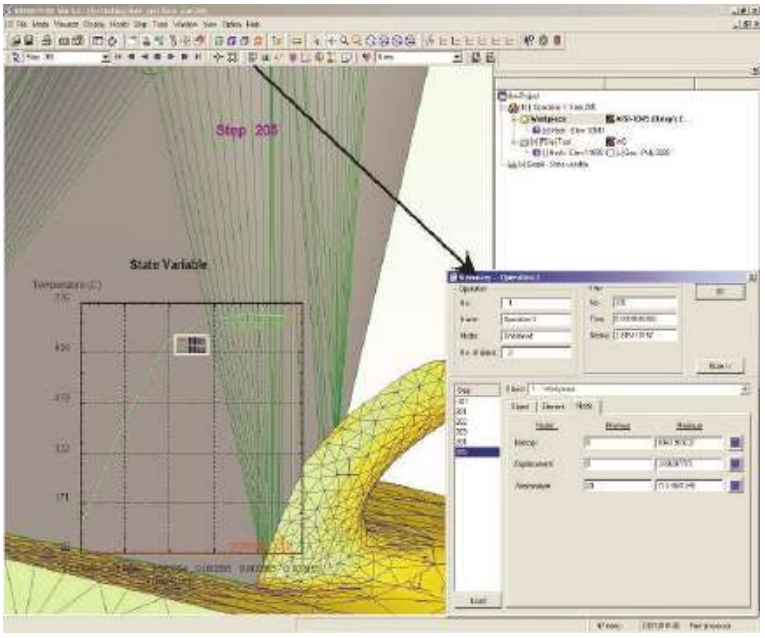


Figure 12: Check the updated temperature on the chip

Step 12
Check for the converged work piece temperature

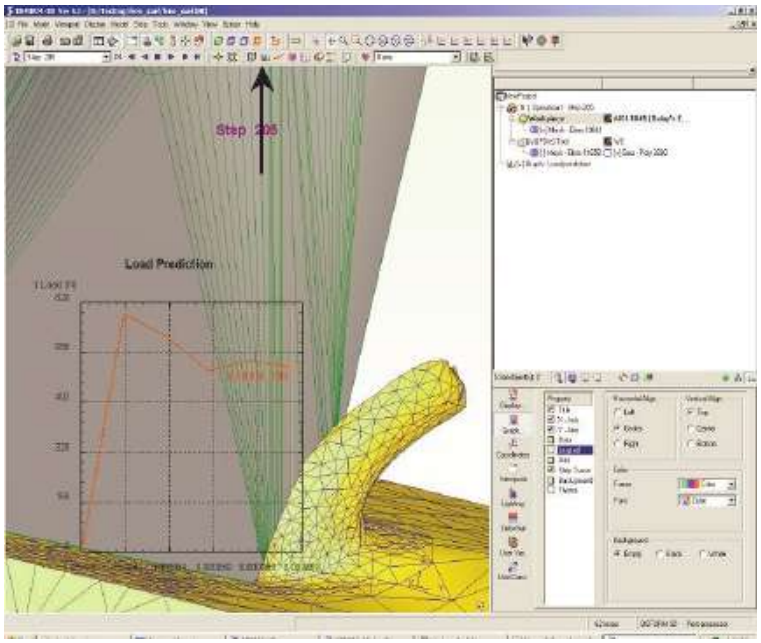


Figure 13: The converged cutting force

Step 13
Check for the converged cutting force on the insert

Appendix O: Document on constructing linear friction simulations

Qualification for the user:

This document is a suggestion on how linear friction welding simulations can be run. This material may be dated or may not represent your process precisely. Please exercise judgment on what is or is not applicable to your process.

Overview:

In this type of simulation, there are two distinct operations that occur: the heat-up stage and the deformation stage. The heat-up stage can be modeled as a pure heat transfer simulation as the oscillatory rubbing between two objects is modeled. Initially, the objects are room temperature and the frictional heating being generated as the simulation progresses characterize this stage. The simulation will stop once the interface temperature is at a set temperature where the objects would become bonded. At this time, the simulation should activate both deformation and heat transfer. This second stage, known as the deformation stage, will model the flow of the material during welding. A fine mesh should be present at the interface to give adequate temperature distribution through the thickness of the parts. As friction welding can be a rather fast process, the temperature gradient through the thickness of the part can be rather steep, thus it is highly recommended to make a fine mesh in the depth of the part as well.

There are several ways in which to run such an operation. A few are as follows:

- Single body with no modeling of the oscillatory motion

This is recommended when a fast simulation needs to be run and if both welding objects are the same size and same material composition at the interface. The heat contribution due to the relative motion of the two bodies is considered but the flash and the temperature distribution is considered to be symmetric.

- Two bodies are modeled with no modeling of the actual oscillatory motion

This is recommended when the two objects are of differing size at the interface or when they are of differing materials. The heating due to the relative motion is considered but no actual relative motion occurs.

- Two bodies are modeled with the actual oscillatory motion

This requires many steps since it is essential to represent the oscillation accurately.

Model Descriptions:

Non-welded linear friction heating

This phase describes how to model the first stage in a linear friction simulation: frictional heating. At this time, the weld hasn't started to begin and the material at the interface is heating and beginning to plasticise. This phase is a thermal calculation only and case use either one or two work pieces.

Single body case:

In this case, please use the same method as the two body case. When the heat up stage is finished, delete one of the bodies and perform the required actions for setting up the deformation stage.

Two body case:

In this case, there are only four objects that are essential: two work pieces, the pusher and the oscillator. The work pieces are the objects that will be bonded. The pusher is a rigid object that applies the upsetting load to the welding objects. The oscillator is an object that allows the welded objects to react against the upsetting load. Also, the oscillator controls the oscillating friction direction. The oscillator should be able to support the upset load. The pusher can be a flat plane or can be a tool shape that holds the work piece. The work piece shape is important and should be whatever shape is being welded. Note that there can be some economy taken in terms of work piece geometries such as an entire blisk is not required for modeling in order to run this type of simulation.

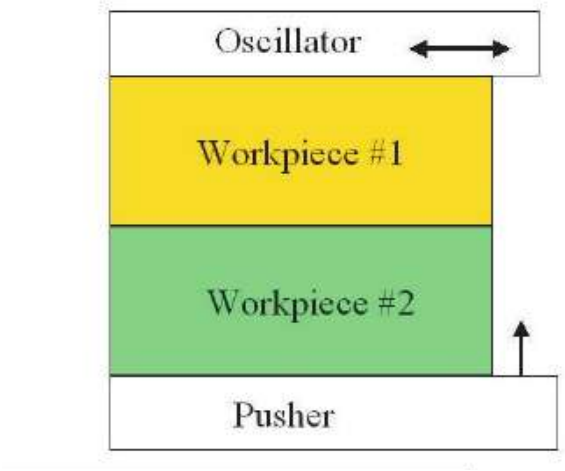


Figure 14: Diagram of two body case.

The movement conditions depend on the actual process, but in the case of a force driven process, the pusher can be specified with the force in the upset direction while the oscillator should be specified with speed control of 0 but with the direction of oscillating motion. The direction in the movement of the oscillator is used to let the simulation engine know the direction of oscillation. The oscillation parameters are taken from a external DATA file.

The inter-object relations are important to obtain a correct result. The relations between the pusher and the work piece should be high friction (constant shear = 20) and a high interface heat transfer coefficient. The relation between the work piece and the oscillator should be high friction (constant shear = 20) and a high interface heat transfer coefficient. Both relationships should have very high separation criteria (absolute pressure of high value, e.g. 1e+09). The contact condition at this point should be completely contacted in areas being pushed or welded. The contact condition between the two deforming objects should have coulomb friction defined.

A file named TFW.DAT should be located in the same directory where the simulation is being run. The content of the file is described as below:

Here is a checklist of all the required settings to run properly:

- There should be a minimum of four objects: two work pieces, an oscillator and a pusher.
- The work pieces should have contact defined at their interface.
- The pusher should have an upset load defined in the direction of the upset.
- The oscillator should have a zero-velocity condition defined with movement direction in the oscillating direction.
- The pusher and the contacting work piece should have high friction and a high interface heat transfer coefficient defined.
- The oscillator and the contacting work piece should have high friction and a high interface heat transfer coefficient defined.
- A file named TFW.DAT should be correctly assembled and placed in the directory where the database is located.

Note that there are a few extra options that can be run for different options. In particular are the options for END.DAT and TRW2.DAT. The first allows the user to consider the fact that the ends will not heat as much as the rest

of the part since it will be exposed to the air for part of the oscillation. The second option allows the pressure distribution to be non-uniform at the interface. Due to certain shapes in linear friction welding, it is possible that the interface pressure can be non-uniform leading to non-uniform heating.

TFW.DAT – Additional input of oscillating movement and pressure for friction heating

Line 1: I21, I22, A23, A24, A25,I26,A27
Line 2: (Optional) I3
If I3 exists:
Line 3: A41, A42
Line 4: A51, A52
…..

where
I21, I22 – boundary numbers of the friction pair
A23 – half amplitude
A24 – frequency
A25 – pressure
I26 – stopping criterion:
  1 – upset displacement (to be implemented)
  2 – upset rate (to be implemented)
  3 – percentage of the interface area where the yield shear stress has reduced to the frictional traction
A27 – value used in accordance with I26
I3 – total number of data pair defining half amplitude
A41, A42 – time, half amplitude
A51, A52 -- ditto
……

Example:

```
2 3 0.05 30 5 1. 0.
8
0       0.01
0.3    0.012
0.5    0.016
1.3    0.017
2.6    0.013
4      0.014
5      0.015
10     0.02
```

Notes:

1. The oscillator should have the direction defined ("Speed" type movement).
2. Coulomb friction should be used together with an appropriate coefficient.
3. Heat transfer mode or non-isothermal mode can be used. However, the stopping criteria 1 and 2 only work with the latter.

END.DAT – a flag to include the end effect in frictional heating

Line 1: I1

where
I1 = 4 for tetrahedral elements;
= 8 for brick elements

TFW2.DAT – a flag for using the stress as the normal pressure

Line 1: I1

where
I1 – component of stress to be used, such as 1 (x-component), 2 (y-component) and 3 (z-component).

Notes:
1. Non-isothermal mode is required to run the simulation.
2. This file should be used in conjunction with TFW.DAT
3. This file cannot be used together with TFW3.DAT (not in this manual yet).
4. Please note that the stress distribution could be spotty with the tetrahedral elements.

Description:

Simplified fully-welded flashing

After the heat-up stage, the deformation stage can be initiated. It can be performed in two general ways, a single deforming body or two deforming bodies. The single deforming body case is more efficient in terms of computational time and memory usage however should not be used in cases where this assumption is not valid. This assumption is not valid if the cross section of the contacting areas is not the same or if the material properties for the two bodies are significantly different.

In the case of two deforming bodies, the oscillating motion of the pieces can be modeled or not. It is much more efficient to not model oscillation as this requires many more very fine steps. Both methods will be described below.

Single body case:

- One of the two work pieces should be deleted.
- The deformation module should be turned on.
- There should be a minimum of three objects: one work piece, an oscillator and a pusher.
- The work piece should have contact defined at the interface with the pusher and the oscillator.
- The pusher should have an upset load defined in the direction of the upset.
- The oscillator should have a zero-velocity condition defined with movement direction in the oscillating direction.
- The pusher and the contacting work piece should have high friction, a high separation pressure and an interface heat transfer coefficient defined.
- The oscillator and the contacting work piece should have low friction and a low interface heat transfer coefficient defined.
- A fine named EPSR.INI should be correctly assembled and placed in the directory where the database is located.

Two body case (no oscillation):

- The deformation module should be turned on.
- There should be a minimum of four objects: two work pieces, an oscillator and a pusher.
- One of the work pieces should have contact defined at the interface with the pusher.

- One of the work pieces should have contact defined at the interface with the oscillator.
- The two work pieces should have contact defined at their interface.
- The pusher should have an upset load defined in the direction of the upset.
- The oscillator should have a zero-velocity condition defined with movement direction in the oscillating direction.
- The work piece interface should have appropriate friction and separation pressure.
- The pusher and the contacting work piece should have high friction, a high separation pressure and an interface heat transfer coefficient defined.
- The oscillator and the contacting work piece should have high friction and a high interface heat transfer coefficient defined.
- A fine named EPSR.INI should be correctly assembled and placed in the directory where the database is located.

EPSR.INI – data to initiate a simplified welding and flashing simulation

I11, I12, I13, I14, I15
A21, A22

where
I11 – type of strain rate to be added to the de-coupled model
= -1 – effective strain rate
= 1, 2, 3, 4, 5 or 6 -- x, y, z, xy, yz or zx component of strain rate
= 7 – all six strain rate components
I12 – boundary number of the oscillator
I13 – boundary number of the pusher
I14 – weld object number, if there is only one involved in simulation
= 0 – for two objects to be welded together
I15 – step increment where the strain rates are calculated using the full model and saved for the next I15 steps of de-coupled simulation
A21 – period of the oscillation
A22 – half amplitude of the oscillation

Example:
5 4 1 0 5
0.16 0.01

Description:

Fully modeled oscillation

Two body case (with oscillation):

- The deformation module should be turned on.
- Polygon edge length sub stepping should be turned off (set to zero).
- Add a stopping distance between the pusher and the oscillator where each point lies on the respective object. The purpose for this is that the oscillation distance at a given step is stored in the oscillation direction component of the stopping distance. For example, if the objects oscillate in the z-direction, leave the z-components of the stopping distance zero so that these values can update properly.
- There should be a minimum of four objects: two work pieces, an oscillator and a pusher.
- One of the work pieces should have contact defined at the interface with the pusher.
- One of the work pieces should have contact defined at the interface with the oscillator.
- The two work pieces should have contact defined at their interface.
- The pusher should have an upset load defined in the direction of the upset.
- The oscillator should have a zero-velocity condition defined with movement direction in the oscillating direction.
- The work piece interface should have appropriate friction and separation pressure.
- The pusher and the contacting work piece should have high friction, a high separation pressure and an interface heat transfer coefficient defined.

- The oscillator and the contacting work piece should have high friction and a high interface heat transfer coefficient defined.

- A fine named OSC.DAT should be correctly assembled and placed in the directory where the database is located.

OSC.DAT – data for more accurate control of oscillating movement

Line 1: A1, A2, A3, I1, I2

where
A1 – period of oscillation
A2 – starting time
A3 – amplitude
I1 – Boundary number of the oscillator
I2 – divisions in a half amplitude (= 1/4 oscillating cycle)

Example:

0.33333333 6 0.05 4 5

Appendix P: On Using Spring-Loaded Dies

The Spring-loaded die setting window

Below is a glossary of the settings for a spring-loaded die as seen in Figure 15.

On/Off – This activates/deactivates the facility for a given object.
Compression Direction – This is the direction in which the movement generates a corresponding spring force.
Stiffness – This is the rate at which the reaction force changes with spring displacement.
Preload – The load at which it is required to initiate movement of the spring.
Current displacement – The current amount of compression the spring is under.
Maximum displacement – This is the maximum amount of compression a spring can be subjected to before it "bottoms out". Setting this value to a very large number gives the spring much travel distance.
Other end of spring – This determines where the other end of the spring is attached. This end can be attached to another object or a fixed point in space.
Reversible – This functionality is meant only for cogging simulations.
Current Die Stroke – This feature a signed three-component vector that shows the current amount of displacement in the spring.

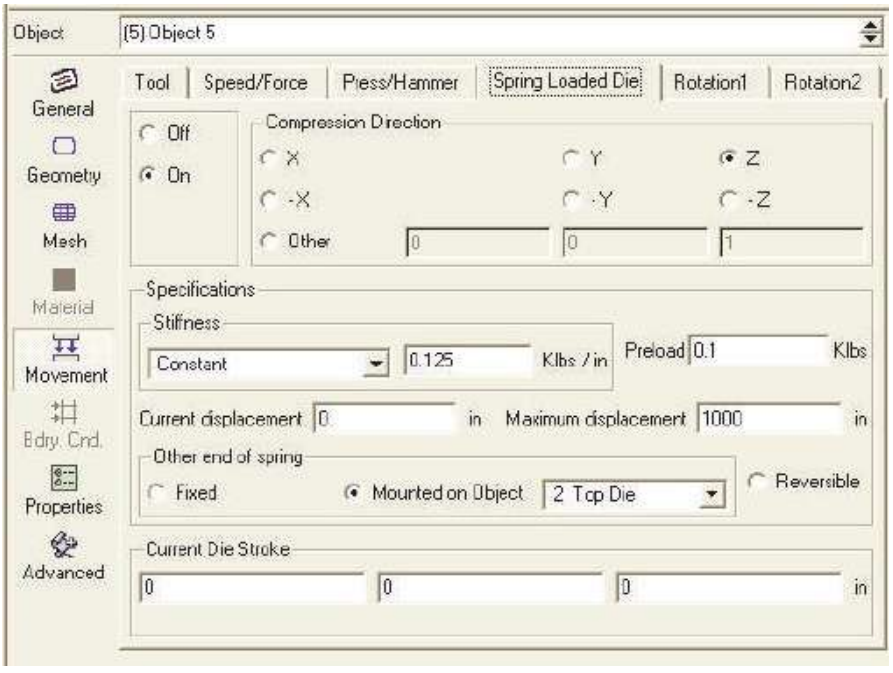


Figure 15: The spring-loaded die setting window

Required Settings to Model a Spring-Loaded Die

Consider the following axisymmetric case seen in Figure 16 where a punch shares a spring with another die. As the punch moves downward, the work piece is formed under the spring-loaded die and pushes the die upward.

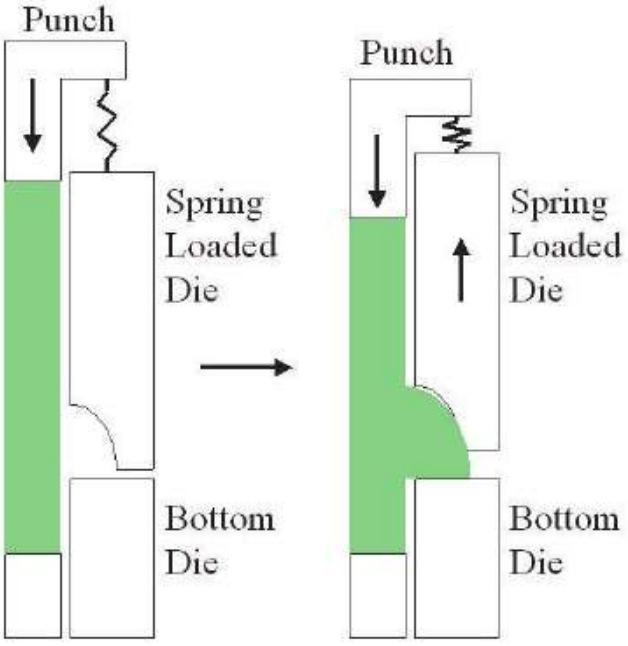


Figure 16: An example spring-loaded die case

To construct this simulation of a plastic work piece and rigid tools, set the following things in the simulation:

1. Set the spring-loaded die settings for the Spring Loaded Die. This includes turning on the spring-loaded die property, setting the compressive direction, the preload, the spring stiffness, the maximum displacement and the other end of the spring. The maximum displacement should be the distance from the top of the spring-loaded die and the bottom of the flange of the punch.

2. Set an inter-object relation between the bottom die and the spring-loaded die. The bottom die should be the master object and the spring-loaded die should be the slave object. Contact will not be generated between the two rigid objects.

Appendix Q: THE DEFORM ELASTO-PLASTIC MODEL

In general, the elasto-plastic material model should be run in a very similar manner as the rigid-plastic material model. Some differences are discussed in the following section.

Material Properties

In addition to the flow stress data, the material is also required to have Young's modulus (YOUNG) and Poisson's ratio (POISON). If thermal expansion and contraction is to be considered, the thermal expansion coefficient must also be present. Note: elastic and elasto-plastic materials in DEFORM deal with thermal expansion differently (see Thermal expansion (EXPAND) for more details).

In the elasto-plastic model, the flow stress at zero strain represents the yield stress for the material. As the accumulated effective plastic strain increases, the yield stress increases. The flow stress data must have a reasonable description for the initial yield stress particularly in the case of low deformation simulations such as heat treatment. This is where the elastic part of the stress-strain curve intersects with the plastic part of the curve. If the flow stress data is only defined for high strain values, DEFORM will extrapolate the yield stress and this value may not be close to the actual yield stress. Thus, valid results are unlikely and convergence difficulties are possible. Thus, having some flow stress data at low effective plastic strain values may aid convergence. (See Figure Q.1 for an example of extrapolation of the initial yield stress).

In order to provide guidance to users who are not familiar with modeling elasto-plastic materials, we offer the following suggestions.

a) When using the function form: $\bar{\sigma} = c\bar{\varepsilon}^{n}\dot{\bar{\varepsilon}}^{m} + y$, it is accurate for y0 to be equal to the initial yield stress. If y = 0, the initial yield stress will be poorly represented.

b) When using a table form, the program will extrapolate/interpolate the flow stress and use that as the initial yield stress. If extrapolation is expected, be sure that the slope of the flow stress in the small strain region is "reasonable", i.e.

extrapolated value should be the same as the initial yield stress. Some modifications may be needed in the small strain region, like adding one more point to correct the slope. These modifications should be made even if the flow stress is retrieved from the DEFORM material database.

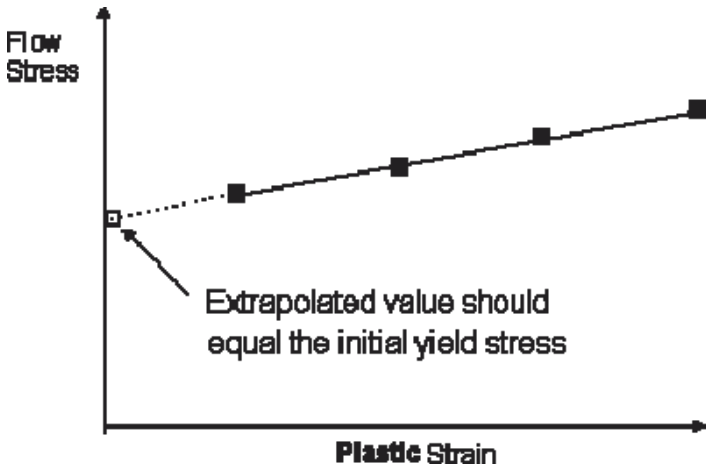


Figure Q.1 : Extrapolation of flow stress data to determine the initial yield stress.

Object data

- Set the EP initial guess under Objects/Properties/Deformation to Previous step solution.

- If there is a change in operation, e.g. moving the part from one station to another, initialize the velocity for the part under Objects/Nodes Data/Deformation. This will improve the initial guess of the velocity solution.

- If moving the part from one set-up to another, allow the part to relax its stresses by placing a few spring back steps between operations.

Solution procedure

- Always use Newton-Raphson iteration method or BFGS iteration method. The user can select the different methods under Simulation Controls/Iteration. It is useful to try switch iteration methods when either one fails to converge.

- The force error norm can be increased two orders of magnitude larger than the velocity error norm. This is suggested only when the solution fails to converge due to non-stationary force error norm behavior. The value can be changed under Simulation Controls/Iteration.

- It is recommended to reduce the value after convergence has been achieved.

- Elasto-plastic convergence is very sensitive on time-step size. By selecting a time step size too large (size depends on the simulation), convergence may be very difficult. By reducing time-step, convergence in many cases may be improved.

Strain Definition

- The calculated strain components are of the "plastic" strain and the "effective strain" is the effective plastic strain.

Advice on Predicting Spring back

- To calculate the residual stress or spring back during unloading, one of two methods may be used.

- If the amount of spring back or residual stress is the chief concern, remove the dies, put enough constraint on the work piece to prevent rigid body motion. run a one step simulation.

- If the user wishes to observe the material response during the unloading, then reverse the direction of the primary die. run multiple small steps.

Note on Convergence issues with Elasto-Plastic material models:

Background: It has been a common and convenient approach for the users to use the material data from an existing model or from library data. As long as material data covers the range experienced by material point for a given

simulation the model behavior is generally smooth. But once this material point crosses the defined data range, it is important to note that the flow stress data extrapolation plays a role in the accuracy of the model results. This becomes even more critical for elasto-plastic models when we need to depend on these extrapolated data to compute material yielding. This note summarizes couple of important points in this regard.
As the example shows in the Appendix, Log interpolation my lead numerical difficulties for some material data. In order to get more stable results, new convention rule is introduced in V8.2.

DEFORM convention rule for material interpolation

Convention: See Figure Q.2

1. If the current plastic strain is smaller than the 1 st data point user defined, then "Linear interpolation" will be used.
2. If the current plastic strain rate is smaller than the 1 st data point user defined, then "Linear interpolation" will be used.

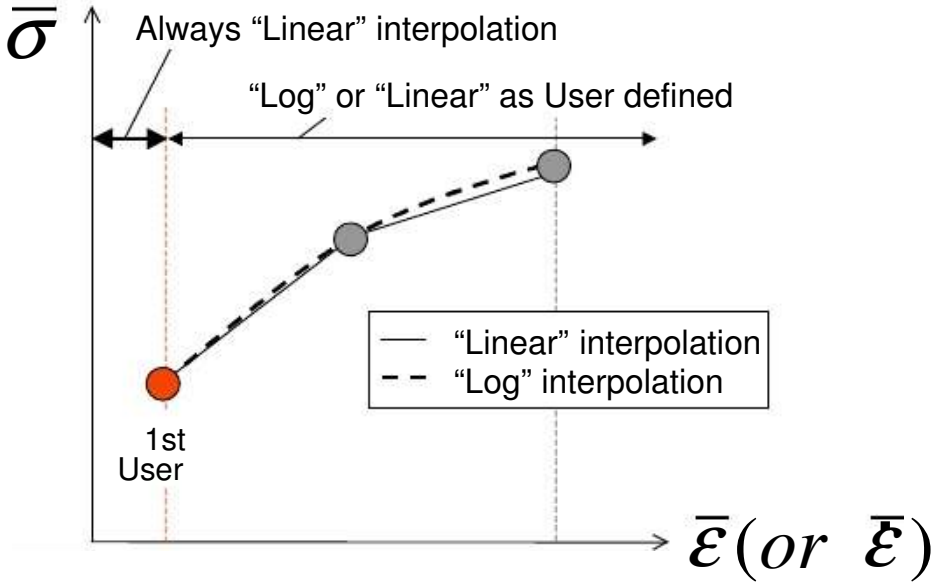


Figure Q.2 : Convention rule for flow stress definition

<u>Note:</u> If the user doesn't want to use the convention rule, the user should specify zero strain and zero strain rate curves.

Comments on "Flow curve definition for Rate-dependent material".

Typical convergence issues related elasto-plastic objects are related to material data and "material instability". It may lead to the failure in "material routine convergence" and can potentially give wrong stress predictions. Here the flow curve definition for Rate-dependent material will be discussed briefly.

1. Material AISI-1010, Cold (Log interpolation): The general shape of curve looks like "Hardening material". Two curves at different strain rates (= 1 , 100) defined.

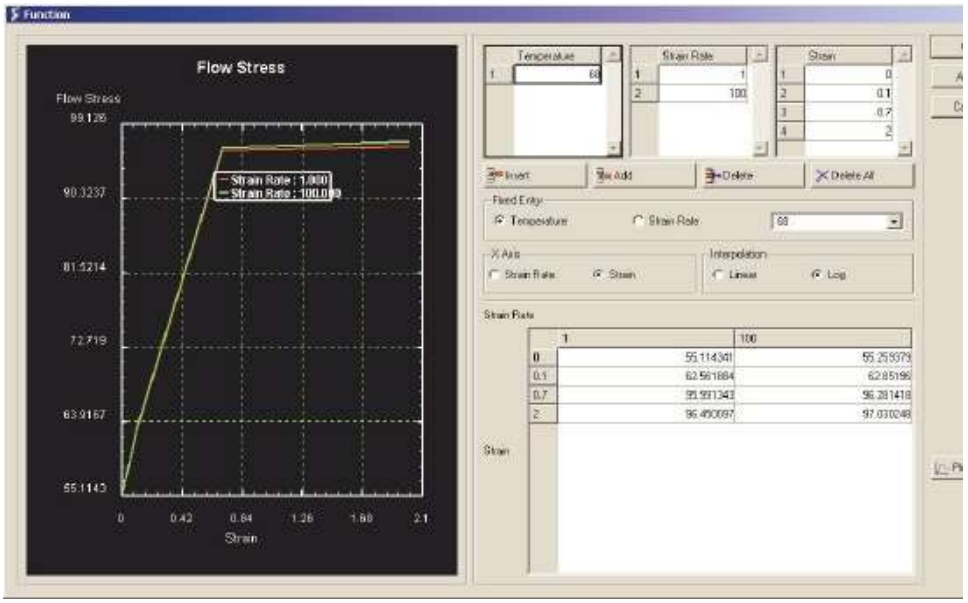


Figure Q.3 : Flow curves for "AISI-1010,Cold"

2. Special care is needed in using LOG interpolation

User can choose one of interpolation methods, either "Log" or "Linear". When user choose "Log (default)", the interpolated curve at the small strain rate (= 0 ~ 1) is shown in Figure Q.4 . The curve at the zero strain rate now has softening behavior. ("Linear interpolation results" is still hardening behavior as shown in Figure Q.5).

During the simulation with Log interpolation, the yield stress (96.0144 at strain rate = 0.032867) was smaller than the static yield stress (96.4215655 at strain rate = 0). This makes "Material instability during radial return mapping). **The**

specific reason of Log interpolation should be examined.

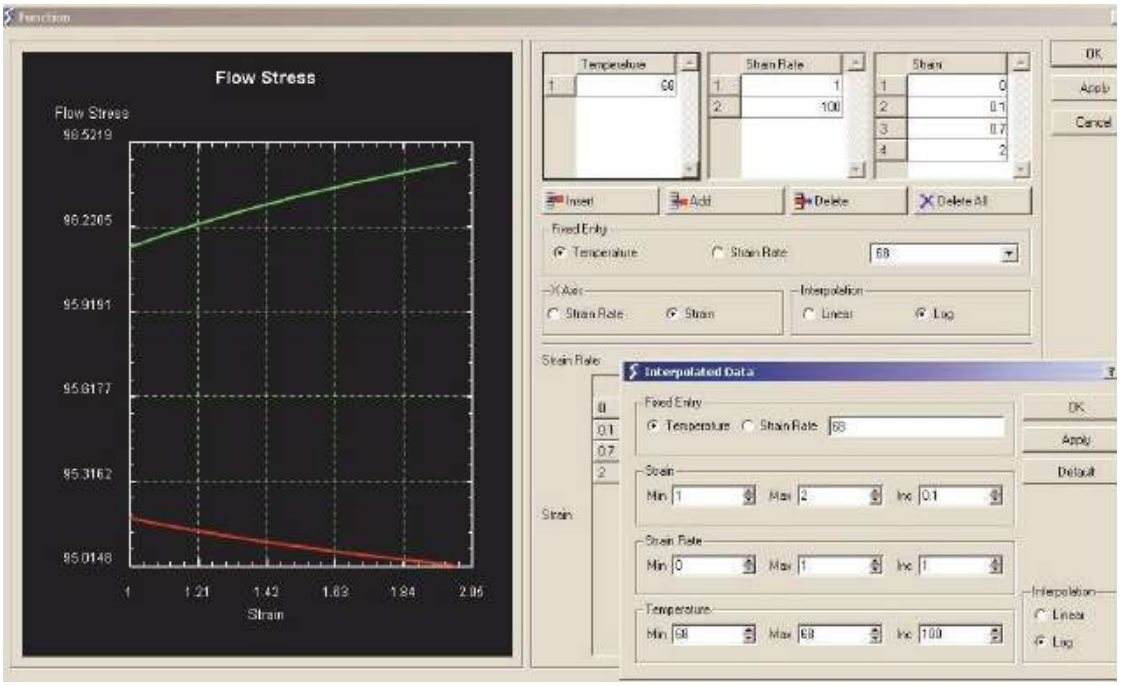


Figure Q.4 : Log interpolation at the small strain rate(= 0 ~ 1)

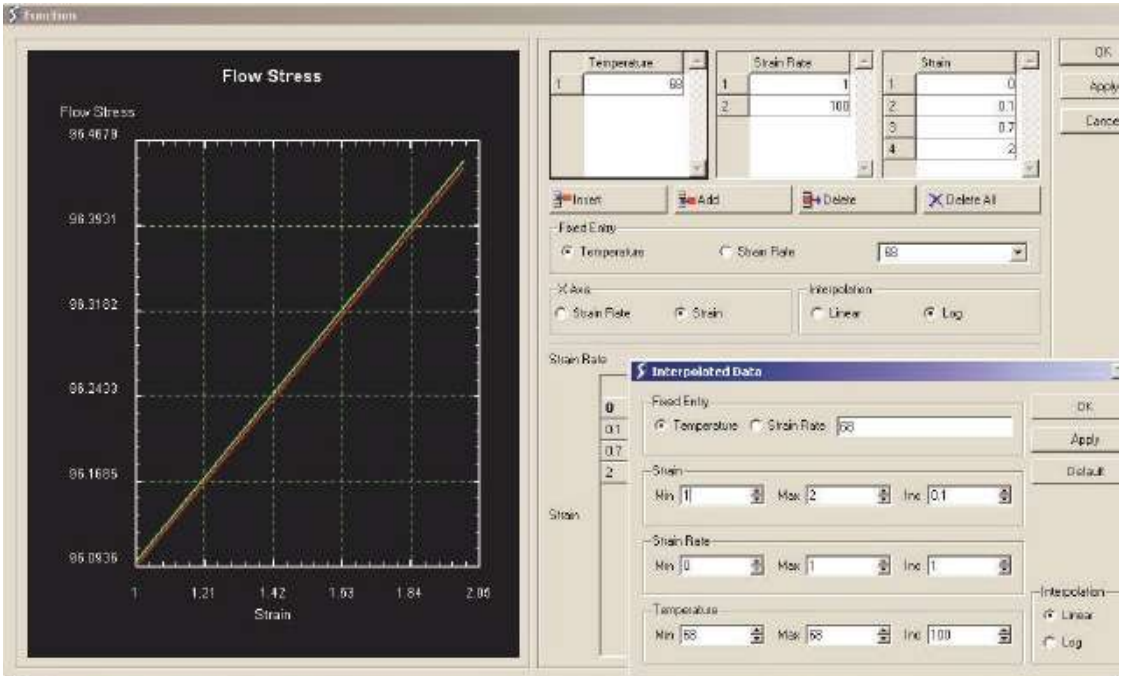


Figure Q.5 : Linear interpolation at the small strain rate(= 0 ~ 1)

Appendix R: Setting Up Multiple Processor Simulations

To turn on multiple processing, open the main window of DEFORM-3D and select the Option pulldown selection and click the Environment option. Scroll through the tabs and select the MPI tab as seen in Figure 17. At this point, click the Use MPI tab and input the name of the computer to run the DEFORM simulation and the number of processors to use.

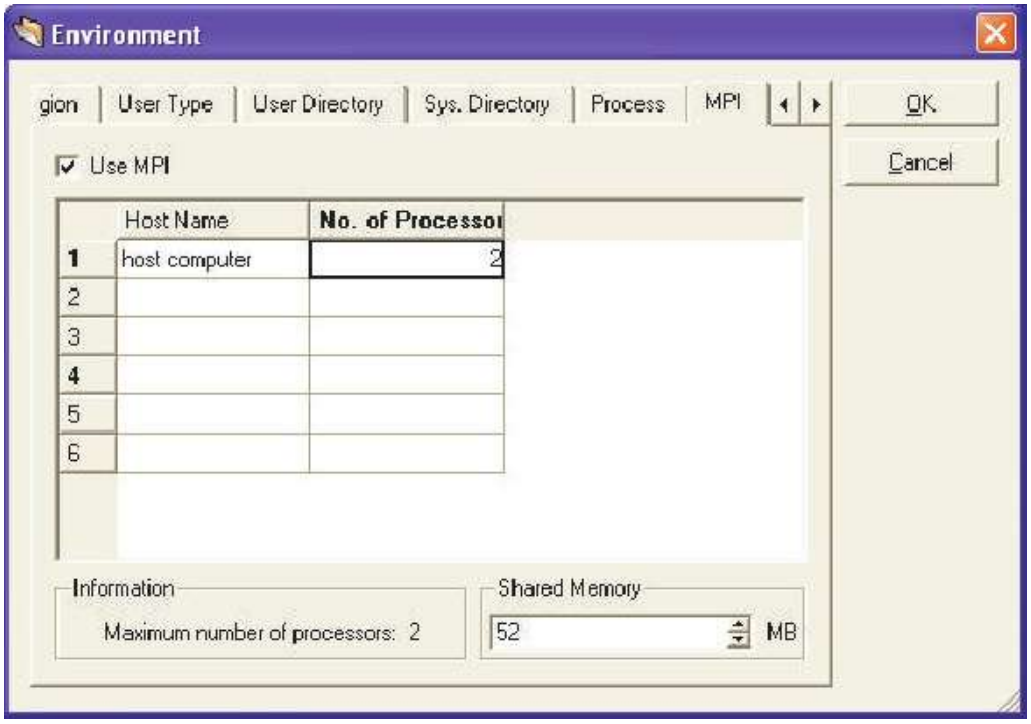


Figure 17: The MPI settings window under Option->Environment

If you are unsure the name of the computer, go to the Control Panel of your Windows machine and click on the System applet and search for your computer name. As seen in Figure 18, the computer name is given where Full Computer Name is. It is very important to ensure that the correct computer name is set under the MPI settings otherwise the simulation will not run.

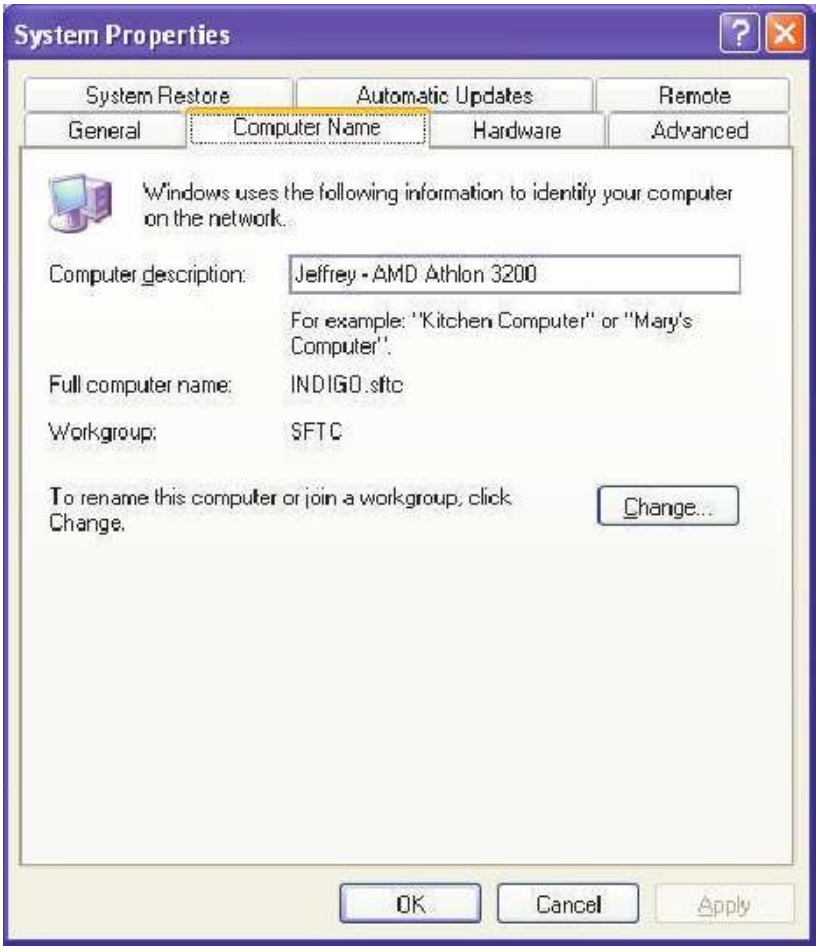


Figure 18: The computer name given by the Windows System applet

Enabling the MPI setting in the Option->Environment will turn on multi-processing for New Problems but will not enable this options for any problems that were created prior to setting this. To enable multi-processing for old problems, click the Run (option) selection in the Simulator section of the main window of DEFORM-3D. A window as seen in Figure 19 will appear. Check the Multiple Processor selection and click the Set Up… button. A window as seen in Figure 20 will appear. Again, the user should input the host name of the computer and the number of processors. After this, the simulation can be run in multiple processor mode.

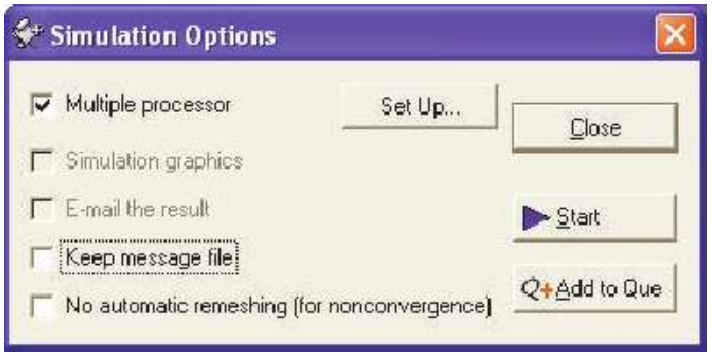


Figure 19: The Run (option) window

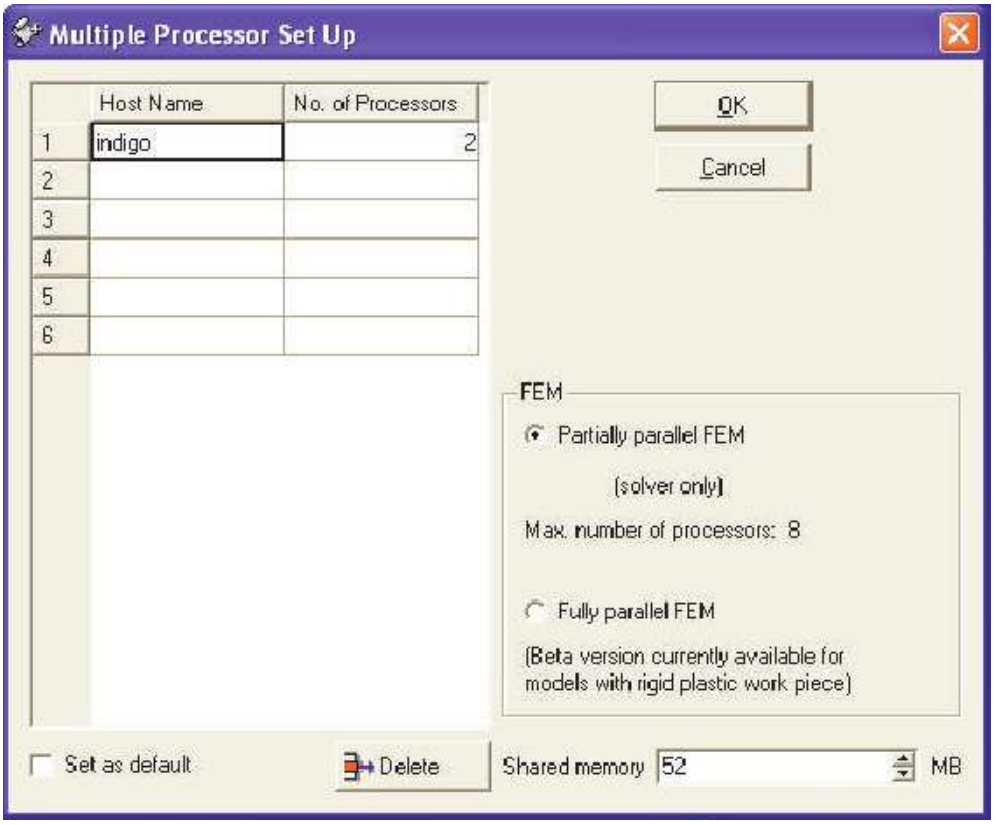


Figure 20: The run option setup window

Appendix S: Coupled Die Stress Analysis

Starting from 3D V6.1 the system supports a convenient way to carryout die stress analysis. Current structure depends on a local DAT file DEF_LCDSTS.DAT) to trigger these procedures. In a typical die stress analysis since fine mesh systems are needed on the die objects, models are memory intensive and need long computing time. Procedures developed in DEFORM allow user to specify different time steps for the deforming objects, elastic dies and for thermal computations. Also depending upon the model size, user can specify if a fully coupled die stress or in a staggered manner using interpolated forces. Either way the model stores the results in the same database. Currently the following requirements have to be met by the model to be able to use these analysis features.

The dies should use only the velocity boundary conditions (The symmetry plane definition will be supported from V6.2)
The dies should be elastic and use tetrahedral mesh
The dies should have movement conditions defined in the BCC dialog
The dies movement should be specified in the movement control dialogs. (The option of movement control with velocity is currently supported in V6.1)

Contents of the DAT file are as follows
Line 1: = 1 for Coupled deformation, = 2 for Staggered using interpolated forces
Line 2: = n ratio of the time steps (Dt_elastic_dies/Dt_plastic_workpiece)
Line 3: = m thermal time step ratio.

For example
n = 5 in Line 2 indicates that the coupled calculations are computed every 5 steps, and every 5th step elastic object sees 5 times the step size compared to plastic workpiece.
m = 5 in Line 3 indicates time step size used by thermal computations is 5 times that of deformations computations.

It was observed that the above procedures result in nearly identical deformation results and load-stroke curves. Comparing the two available options, it was observed the coupled models (Line 1 with entry '1') results in more accurate deformation results, while the other option (Line 1 with entry '2') is more efficient in handling large model size with savings in computing time up to 80% for large die stress analysis problems.

Appendix T: Setting up Steady State Extrusion

General description

The modeling is based on the Eulerian formulation of the finite element method with the assumption that the extrusion process has reached the *steady state* thermo-mechanically. Each of the simulation steps is solved in an uncoupled way, which consists of two stages:

1. the Lagrangian calculation, where the equilibrium is achieved; and
2. the remap of state variables to obtain the convective values.

A few steps of simulation are conducted iteratively until the solutions converge. In these steps, the nodes do not march forward as in the updated Lagrangian formulation, but the free surface of the extrudate will be corrected so the surface is tangential to the material flow and the strain and temperature will be calculated. Accordingly, the step size has nothing to do with the simulation.

To run the steady state extrusion, the regular DEFORM-3D is used. A data file called ALE.DAT is put in the working directory to branch out the regular DEFORM-3D to the steady state capability. Irrespective of the name "ALE" used in the datafile, this capability does not include the arbitrary meshing. The mesh is fixed in space as other steady state modeling; only the free surface of the extrudate is corrected.

Data preparation

The input data needed for the steady state extrusion is very similar to those for a regular forming process modeling. Either non-isothermal or isothermal mode can be run, and either the brick mesh or tetrahedral mesh is supported.

There are some special requirements for the geometry of the billet:

1. The workpiece should entirely fill the die cavities;
2. The initial surface of the extruded part should be parallel to the extrusion direction (-Z);
3. The top of the billet and the end of the extrudate should be perpendicular to the flow direction; and
4. The length of the un-deformed part should be at least three times as long as the maximum dimension of the billet cross-section. The length of the extrudate should not be too long so as to cause too much free surface correction. Four times of the maximum bearing length will generally be fine.
5. The workpiece is set to Obj. 1 and the die Obj. 2.
6. If there are symmetric planes, the old definition (fixed x or y, NBCD=1) are accepted. Don't use the symmetric plane BCCs from the GUI preprocessor (NBCD=6).

There is an extrusion template coming with DEFORM-3D that helps prepare the brick meshes for the steady state extrusion and creates ALE.DAT. However, the user can bypass it as long as the database can be generated in the regular preprocessor and ALE.DAT can be prepared with a text editor.

ALE.DAT preparation

This data file provides the additional data needed to control the simulation.

FORMAT (I – integer; A – real)

Line 1: I1, I2, I3
Line 2: A1

(If running the heat transfer mode only, stop here.)

Line 3: A2, A3, A4, A5, I4
Line 4: I5
Line 5: (I6(n), n = 1, I5)
Line 6: I7
Line 7: A6, A7

CONTENTS

I1 – Object number for the workpiece.
I2 – In flow direction. At present only use "-3," indicating the -Z direction.
I3 – Out flow direction. At present use "-3" only.
A1 – Z coordinate of the top of the billet inflow plane, where it is pushed by the punch.
A2 – Z coordinate of the die orifice.
A3 – Z coordinate of the point where the bearing length is minimum.
A4 – Z coordinate of the point where the bearing length is maximum.
A5 – Z coordinate of the end of extrudate.
I4 – Total boundary number of the out flowing extrudates.
I5 – Total number of the extrusion dies that form the extrusion orifice.
I6 (1 ~ I5) – List of the above die object numbers.
I7 – Total number of the "coating" layers; if no coating, leave 0 there;
If I7 < 0, the strain correction will be applied (under testing). For the die design
without a pocket, set I7 = -1. If there is a pocket, set I7 = -2.
A6 – Optional deacceleration coefficient for free surface calculation (0. ~ 1.) used for
Step 1 to (Nmax – 1) to avoid negative Jacobian if the free surface distortion is

too big, where Nmax is the maximum step number specified.
A7 – ditto, for the last step, i.e. at step Nmax, A7 will be applied for free surface calculation. If not specified, A7 = 1 will be used.

Some techniques for running the steady state extrusion

1. Brick mesh or tet mesh

Brick mesh generated with the template is good for small, simple geometry. When real industrial problem is to be modeled, the tetrahedral mesh currently is more realistic, both for the workpiece and the dies. However, the surfaces of the brick meshes generated using the extrusion template can be used to generate the tet meshed in the DEFORM-3D preprocessor.

2. How to generate the workpiece mesh

a. The mesh density should be fine enough in the area of the extrusion die orifice and the bearing length. At least three to four elements should be deployed across the thinnest section. The density can be slightly smaller where the extrudate is out of the bearing zone.
b. To use resources effectively, the mesh density windows can be utilized to set different mesh densities for different zones.
c. There should be no empty object numbers, i.e., the object numbers should be consecutive.
d. When the geometry is provided by the STL format, consisting of many slim triangles, it is a good idea to generate a uniform surface mesh by using the DEFORM pre-processor and setting the relative mesh density definition. Save the new surface mesh and use it to generate the surface and solid meshes for FEM simulation.

3. Test run with the Lagrangian formulation.

Since the steady state simulation starts from a Lagrangian stage, it is advisable to run through one step of the regular simulation using the updated Lagrangian formulation and check the results of the velocity field. A small step size, such as 1.e-5 sec. is suggested.

As an indicator of a reasonable solution, the outflow nodal velocities on the extrudate should be checked against the in-flow velocity of the billet to make sure that their ratio is about right:

Outflow speed / Inflow speed = Billet cross-section area / Extrudate cross-section are.

In case the above ratio cannot be roughly maintained, there are two possible reasons:

a. "Leaking" may exist. This is easily seen when the nodal velocity vectors of the workpiece are examined in the post-processor. If large velocities piercing out of the die surface, they cause the leaking and should be fixed before starting the steady state simulation (See Section 4 below).
b. The elements across the die orifice are not fine enough so the material cannot be squeezed out (See Section 2-a).

After a reasonable velocity solution is obtained, check the velocity components of the end of extrudate. If the lateral component is more than twice of the outflow velocity, the free surface correction algorithm may not work properly. If the free surface distorted too much in the steady state run even a small value is used for A7 (See above), the bearing length can be adjusted in the die design to reduce the distortion.

4. How to prevent the extruded material from leaking

The following should be done in the Lagrangian version (without ALE.DAT) before moving to the steady state simulation.

a. Based on volume constancy, the outflow speed and the inflow speed of the extruded material should be equal or close to the area reduction mentioned above. If this is not even roughly held, some nodes must be leaking and have to be fixed before going on to the steady state calculation.

b. If a contact node stays at a concave die corner consisting of two orthogonal die surface polygons, ideally this node should move along the common side of them. However, in the current DEFORM-3D, each contact node only derives one constraint from the die surface. While this is not a big problem for the conventional forming process, it will create a leak in the extrusion simulation. As a temporary fix, the user can put a data file NBC.DAT in the working directory with the following one line content:

   75 105

This means if the concave angle is between 75 and 105 degrees, an additional constraint perpendicular to the first constraint will be added to the node. However, if a node is close to but not close enough to a die corner, even NBC.DAT does not work. The coordinates of this node should be manually changed so that it is located right at the corner.

c. If the die STL geometry is not smooth enough so there is a small kink or fold on the die surface and a node happens to fall into the tiny irregularity, it may get a wrongly oriented constraint and end up with a high velocity shooting outwards like a leak. If this happens, the die geometry should be fixed.

d. Another measure for the node at the concave die corner is to simply define a sticking boundary condition for it. To define the sticking contact condition, you can manually assign the node with "-20n" for the BCCDEF in the pre-processor, where "n" is the object number of the die which the node contacts. The program will take this negative number as the sticking boundary condition and fix this node at the die surface. However, it cannot slide along the common edge of the die surface as it should.

e. If a node contacts the convex die corner like the edge of the die orifice, the constraint direction in the current DEFORM-3D depends on which surface polygon that forms the die corner has a smaller number. For aluminum extrusion, the sticking contact condition as in "d" can be applied to this node to solve the singularity problem.

5. How to treat the nodes at the sharp, convex die corner

In the extrusion of aluminum, there are sharp (90 degrees) convex die corners at which the material flow changes its direction drastically. To handle this problem of singularity, Another way of handling this is to specify the die edges by picking a starting and ending points in EXTDIE.DAT (See below) so that the nodes at a die corner edge will obtain an additional constraint or a normal constraint which is the average of the two normal vectors to the surfaces that form the sharp corner (Or you can imagine it as the sharp die corner being flattened out to become a small slope where the node is located).

An alternative to EXTDIE.DAT is to specify a special boundary condition to the nodes right at the sharp convex die corner to make them stick to the die corner (velocity equal to zero for this node) by using (1, 1, 1) or (0, 0, -20n), where n is the die object number (See above). This is not as good as EXTDIE.DAT, because the possible movement of the nodes along the die corner edge is stopped, which becomes an over-constraint.

6. How to define the BCCTMP

In addition to the regular BCCTMPs such as "heat exchange with environment," the following should be done:

a. On the top surface of the billet where it contacts the punch, the fixed temperature BCCTMP should be defined (BCCTMP=1) to the contact nodes with the initial billet temperature specified.

b. Similar BCCTMPs may be prescribed on an appropriate die surface with the initial die temperature specified.

7. Miscellaneous

a. The punch is not supposed to be meshed, but the initial billet temperature should be specified to the punch as its "Reference Temperature" (Object>Properites…)

b. The current DEFORM-3D does not require that the die should be slightly bigger than the billet and the punch OD should be slightly bigger than the container ID. If the billet and dies are of the same dimension on the symmetric plane, the surface polygons of the die on the symmetric planes better be defined as the "Symmetry Surface" in the pre-processor (Object>Geometry), so they will be excluded in the calculation to avoid mistake. Similarly, the lateral surface of the punch better also be defined as "Symmetry Surface."

c. If the setup has two dies stacked one above the other, their interfaces should also be defined as the "Symmetry Surface" in the pre-processor. This is to prevent the nearby contact nodes from moving into the die interface.

d. Set DPLEN = 0 in the pre-processor (Simulation Controls>Stopping Step>Advanced Step Controls).

EXTDIE.DAT preparation

This data file provides the additional data needed to control the boundary condition for the nodes at the die edge for either Lagrangian simulation, ALE formulation or steady state simulation of extrusion problems. Please note that the nodes should be close enough to the die corner and each should have a contact BCC to benefit from EXTDIE.DAT.

FORMAT (I – integer; A – real)

Line 1: I1, I2, I3, I4, I5

Lines 2~I1+1: I6, I7, I7, I9, A1, I10

CONTENTS

I1 – Total die edge number to be picked for the special treatment of contact nodes.
I2 – Out flow extrusion direction. At present only use "-3," indicating the -Z direction.

I3 – Object number for the workpiece.
I4 = 1 for a fresh case or an old case;
= 2 for an old case where EDGEINFO.DAT is available in the directory.
I5 – Treatment for the contact nodes at the die edge:
= 1, an additional constraint is added so the node can slide along the die edge, but will not leave the die edge
= 2, a skew normal, which is the average normal to the two die surface, is applied
I6 – Starting point number of the die edge curve
I7 = 0 for a closed curve; or the next point number on the edge, if this is an open curve
I8 = 0 for a closed curve; or the ending point number of the edge for an open curve
I9 = 0
A1 – Angle of the cross-section of the die corner in degrees, e.g., 90.
I10 – Boundary number of the die corner edge. If there are no empty objects, the boundary numbers are the same as the object number. The boundary numbers are always continuous. For instance, if the object numbers are 1, 3 and 4, there boundary numbers are 1, 2 and 3.

NOTE

After EXTDIE.DAT is used, a file named EDGEINFO.DAT is generated. It consists of the point numbers of the edges specified in EXTDIE.DAT to save time for the future use. Once the simulation is finished, EDGEINFO.DAT should be removed or renamed so it will not get messed up with the other die geometry.

EDGEINFO.DAT

This data file is output by the program when EXTDIE.DAT is first used.

FORMAT (I – integer; A – real)

Line 1: I11
Line 2: I21, I22, I23, I24, I25, I26
Line 3 and later: (I31, I32, I33)*I23

CONTENTS

I11: Total number of curves
I21: Boundary number the curve belongs to
I22: Open ends or closed ends
I23: Address of the starting point
I24: ditto, the ending point
I25: Address of the starting sharp corner
I26: ditto, the ending sharp corner
I31: Point number
I32: 1st polygon number between this and the next points
I33: ditto, 2nd polygon number

PKVEL.DAT

This data file provides the additional data needed to control the boundary condition for the nodes at the die edge for either Lagrangian simulation, ALE formulation or steady state simulation of extrusion problems. The purpose is to prevent billet nodes from "leaking." The idea is to compare the surface node velocity magnitude with the theoretical flow speed. The latter is calculated based on the ram speed, cross section areas of the un-deformed billet and the section where the node is located.

FORMAT (I – integer; A – real)

Line 1: I11
Line 2: I21, I22, I23, A21
Line 3 and later: A31, A32
…..

CONTENTS

I11: Option number—fixed as 6.
I21: Total number of the cross-section areas to be input
I22: Extruding direction
I23: Billet object number
A21: Ram speed
A31: Coordinate of the cross-section in the extruding direction
A32: Area of the cross-section

NOTE

To choose the cross-section, the initial billet has to be included and put as either the first or the last one. Other cross-sections are chosen in the place where the cross-section of the object is about to have a change. The cross-section should NOT include any in the bearing channel. The order of the sections should be in order of the increasing or decreasing coordinates. Jumping around will cause mistakes.

(First prepared on Sept. 8, 2003; revised on June 6, 2007, good for DEF_SIM v61 or after)

Appendix U: Setting up 3D machining models

DEFORM-3D version 6.1 contains several enhancements which substantially improve the performance of metal cutting simulation. Mesh definition is the most critical factor in simulation performance. The goal is to adaptively refine the mesh to maintain small elements in areas where they are necessary to maintain geometry or state variables. At the same time, we wish to keep the total number of elements in the simulation to a minimum. The most significant enhancement in 3dv61 is local remeshing – rather than completely regenerating the mesh, as was done in earlier versions of DEFORM, elements are simply split or merged to improve quality and match local element size requirements. Elements which meet local size and quality requirements are not changed.

Major advantages of this new feature are that the problem of element deletion due to the chip touching the edge of a workpiece has been nearly (if not completely) eliminated, and the changing shape of sharp curves such as the nose radius feature has been substantially reduced.

A new interpolation scheme has been implemented which uses a least squares fit of surrounding elements, and reduces state variable smoothing during repeated remeshings. Because mesh element size definition is based on these state variables (particularly strain), mesh generation behavior is also changed. Different approaches to mesh generation are appropriate depending on whether the user's primary interest is the chip or the workpiece. Both approaches are described below.

A) For capturing chip geometry

- these settings will give good resolution in the chip, but will tend to loose or smooth out temperature, residual stress, and microstructure information in the workpiece.

- There is no need for mesh windows. In nearly every case, properly defined adaptive meshing will consistently provide a good quality mesh with substantially fewer elements than can be achieved with mesh windows.

- Use Absolute Element Size. Set the minimum element size to about 1/3 or ¼ of the uncut chip thickness. ¼ will give better results, but run time will be significantly longer.

- Set the size ratio between 10 and 15.

- Use Local Remeshing: under Mesh->Remesh Criteria->Remeshing

Method select “Local Remeshing”

- Set the mesh weighting factor slider bars to 50% strain, 50% strain rate, all other values = 0.

- Edit the Local Remeshing file. Because Local Remeshing is a new feature, and the settings are still being tested, it is controlled through a text file. Using a text
- editor such as Notepad, change the value of FURTHER_IMPROVE_CONTACT_ELEMENTS” from 0 to 1

- The mesh generator normally determines mesh refinement based on the gradient of state variables. In other words, a region where state variables (strain or strain rate) are changing quickly will get relatively fine elements, but regions which have high constant values will get a relatively coarse mesh. For machining we want fine elements in the chip (high constant strain), and in the primary shear zone (high strain rate). To trigger the mesh generator to do this, create two files in the same directory as the database:

    - STRAIN_DST.DAT

    - STRAIN_RATE_DST.DAT

- The files can be empty. The mesh generator only checks for their presence.

- With the new interpolation scheme, the workpiece tends to keep a high strain, and therefore a fine mesh, on the cut surface. While this is technically a more accurate result, if the interest is in the chip and not the workpiece, it creates a lot of extra elements in the workpiece that don’t contribute to the results of interest.
- To avoid this, create a text file named OPT.DAT with contents ‘1’ (no quotes). This signals the system to use the old interpolation scheme.

Summary (these settings will maintain chip geometry, but will tend to loose state variables in the workpiece):

- absolute element size – minimum size 1/3 to ¼ of uncut chip thickness. Size ratio 10 to 15.
- Local remeshing
- Slider bar weighting 50% strain, 50% strain rate
- Edit LOCAL_REMESH.DAT – further improve contact elements = 1
- STRAIN_DST.DAT and STRAIN_RATE_DST.DAT files in the same directory as the database (contents irrelavent)
- OPT.DAT in the same directory as the database (contents ‘1’).

B) For capturing workpiece properties

For simulations where workpiece surface properties (residual stress, microstructure, temperature) are of interest, but chip geometry is not. For these simulations, a mesh window may be helpful to maintain mesh size on the cut surface. Strain based meshing may also be adequate. The user may wish to experiment with both approaches to find which gives better results.

With mesh windows.

- Use absolute element size. Use a size ratio of 1, and set the global element size
- to be about 5 x larger than the expected surface layer thickness. Windows will be used to refine the mesh in the surface layer.

- Define a mesh window from slightly in front of the tool edge, and extending backward. Define window movement to follow the cutting tool. The window can extend substantially behind the workpiece, such that it covers more and more of the workpiece surface as the tool advances.

- Assign element size in window to be roughly 30-70% of the size of the expected surface layer effect. In other words, if the residual stress variations are over a depth of 0.01mm, the minimum element size should be around 0.005mm. Note that there will always be a difficult balance between adequate resolution and acceptable run times.

- The size ratio between elements in the window and elements outside the window should not exceed about 5:1. If necessary, nested windows should be used with 5:1 ratio maintained between adjacent windows. The smallest elements (innermost window) should be first in the list of windows.

- Slider bars should be weighted either fully to mesh windows, or roughly 80% mesh windows, 20% divided between strain & strain rate.

- Local remeshing will do a better job of maintaining state variables. Use Local Remeshing: under Mesh->Remesh Criteria->Remeshing Method select “Local Remeshing”

- Edit the Local Remeshing file. Because Local Remeshing is a new feature, and the settings are still being tested, it is controlled through a text file. Using a text
- editor such as Notepad, change the value of FURTHER_IMPROVE_CONTACT_ELEMENTS” from 0 to 1

Without mesh windows

- Use absolute element size, with the minimum element size 30-70% of the expected surface layer effect thickness.

- Use a size ratio of 10

- Use Local Remeshing

- The new interpolation scheme will do a good job of maintaining strain in the cut
- surface of the workpiece, so this can be used as a key meshing parameter. Set slider bars to 80% strain, 20% strain rate.

- Edit the Local Remeshing file. Because Local Remeshing is a new feature, and the settings are still being tested, it is controlled through a text file. Using a text
- editor such as Notepad, change the value of FURTHER_IMPROVE_CONTACT_ELEMENTS" from 0 to 1

- Create a STRAIN_DST.DAT file in the same directory as the database. This will cause the mesh generator to maintain a fine mesh in regions where the strain is high (ie the cut surface). The contents of the file are arbitrary.