

Design Environment for FORMing

DEFORM™ 2D Version 8.1 User's Manual

Jeffrey Fluhrer

Scientific
Forming
Technologies
Corporation



5038 Reed Road
Columbus, Ohio, 43220
Tel (614) 451-8330
Fax (614) 451-8325
Email support@deform.com

Table of Contents

PREFACE TO THIS MANUAL	5
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	13
1.8. Creating input data	14
1.9. File system	15
1.10. Running the simulation	16
1.11. Post-processor	17
1.12. Units.....	17
Chapter 2. Pre-Processor	19
2.1. Simulation Controls	19
2.1.1. Main controls.....	20
2.1.2. Step controls.....	22
2.1.3. Advanced Step Controls	25
2.1.4. Stopping controls.....	27
2.1.5. Iteration controls.....	29
2.1.6. Processing conditions.....	31
2.1.7. Advanced controls.....	33
2.2. Material Data.....	37
2.2.1. Phases and mixtures (MSTMTR) [MIC]	37
2.2.2. Elastic data	38
2.2.3. Thermal data.....	39
2.2.4. Plastic Data.....	41
2.2.5. Diffusion data	47
2.2.6. Grain growth/recrystallization model.....	50
2.2.7. Hardness data [MIC].....	54
2.2.8. Advanced Features	56
2.2.9. Material data requirements	58
2.3. Inter Material Data	61
2.3.1. Transformation relation (PHASTF) [MIC]	62
2.3.2. Kinetics model (TTTD) [MIC].....	63
2.3.3. Latent heat (PHASLH) [MIC].....	67
2.3.4. Transformation induced volume change (PHASVL) [MIC]	68
2.3.5. Transformation plasticity (TRANSFP) [MIC]	69
2.4. Object Definition	69
2.4.1. Adding, deleting objects.....	71
2.4.2. Object name (OBJNAM)	71

2.4.3. Primary Die (PDIE).....	72
2.4.4. Object type (OBJTYP).....	73
2.4.5. Object geometry.....	75
2.4.6. Object meshing.....	80
2.5. Inter Object Definition.....	117
2.5.1. Inter object Interface.....	118
2.5.2. Positioning.....	123
2.5.3. Inter object boundary conditions.....	124
2.6. Database Generation.....	125
Chapter 3. Running Simulations.....	127
3.1. Interactive and batch modes.....	127
3.2. Relevant files.....	127
3.3. Email the Result.....	128
3.4. Starting the simulation.....	128
3.5. Add to Queue (Batch Queue).....	128
3.6. Process monitor.....	129
3.7. Stopping a simulation.....	129
3.8. Troubleshooting problems.....	130
3.8.1. Simulation aborted by user.....	131
3.8.2. Cannot remesh at a negative step.....	131
3.8.3. Remeshing is highly recommended.....	131
3.8.4. Negative Jacobian.....	132
3.8.5. Solution does not converge.....	133
3.8.6. Stiffness matrix is non-positive definite.....	135
3.8.7. Zero pivot.....	136
3.9. Database Management.....	137
Chapter 4: Post-Processor.....	139
4.1. Post-Processor Overview.....	139
4.2. Graphical display.....	140
4.2.1. Window layout.....	140
Object Display Modes.....	141
Tree levels and functions.....	141
Additional Post Processing Functions.....	143
4.3. Display Window.....	144
4.4. Graphic Utilities.....	145
4.5. Post-Processing Summary.....	147
4.5.1. Simulation Summary.....	147
4.5.2. Object Mirroring.....	148
4.5.3. State Variables.....	150
4.5.4. Flow Net.....	158
4.5.5. Point Tracking.....	164
4.5.6. Load Stroke.....	165
4.5.7. Steps.....	166
4.5.8. Database Management.....	168
4.5.9. Nodes window.....	168
4.5.10. Elements window.....	170

4.5.11. Object Edges	170
4.5.12. Viewport.....	171
4.5.13. Data Extraction.....	172
4.5.14. State variable distribution between 2 points.....	174
Chapter 5: Elementary Concepts in Metalforming and Finite Element Analysis	177
Chapter 6. User Routines	189
Overview:	189
User-Defined FEM Routines.....	189
Summary of subroutines and calling structure of user-defined FEM routines.....	190
User-Defined Post-Processing Routines.....	192
6.1. User defined FEM routines	192
6.2. User defined post-processing routines.....	214
Release Notes.....	218
System Installation: UNIX and Linux.....	224
Installing Java for the 2D cutting template (HP-UX 11.0 only).....	229
NT/2000/XP Installation: Installation Notes.....	236
Chapter 7: Quick start tutorial	244
7.1. Cold Forming.....	244
7.2. Hot Forming	249
Appendix A: Running an inertia weld simulation in DEFORM	256
Appendix B: Running DEFORM in text mode.	258
Appendix C: Inserting DEFORM™ Animations in Powerpoint Presentations	261
Appendix D: Adding Gas Trap Calculation to a Simulation	263
Appendix E: Modeling of a Spring-Loaded Sliding Die	266
Appendix F: Using the Inverse Heat Transfer Template.....	271
Appendix G: THE DEFORM ELASTO-PLASTIC MODEL	272
Appendix H: Die Stress Analysis - Theory	275
Appendix I: Running creep simulations in DEFORM-2D	276

Preface to this manual

This manual describes the features and capabilities of the DEFORM-2D 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-2D, 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-2D are documented in the keyword reference manuals which is 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-2D system and describes the components that make up the system.

Simulation Controls

describes the options to control the numerical behavior of the system

Material Data

describes the different material parameters that are required for simulations

Inter Material Data

describes inter material data which can be used to describe phase transformation related effects in the heat treatment system.

Object Definition

describes the definition of objects, loading geometry, generating meshes, setting boundary conditions, etc.

Inter Object Definition

describes how relations between objects are to be set, the interface friction and interface heat transfer coefficients.

Database Generation

describes database generation and the errors and warnings during database generation.

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

Installation Notes for (Unix) and (NT/2000)

contains installation 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 all popular UNIX platforms (HP,SGI,SUN,DEC,IBM) as well as personal computers running Windows-NT/2000/XP. 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 all popular UNIX (HP,SGI,SUN,DEC,IBM) platforms, as well as personal computers running Windows-NT/2000/XP. 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-PC (PC)

Available on personal computers running Windows 95, 98, or NT/2000/XP. Capable of modeling two dimensional axisymmetric or plane strain problems. Suitable for small to mid-sized shops starting in Finite Element Modeling.

DEFORM-PC Pro (Pro)

Available on personal computers running Windows 95, 98, or NT/2000/XP. A powerful two dimensional modeling package containing most features available on DEFORM-2D.

DEFORM-HT

Available as an ad-on to DEFORM-2D and -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 springback problems. (Pro, 2D, 3D).
- Porous material model for modeling forming of powder metallurgy products (Pro, 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, Pro) 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, Pro).
- Multiple deforming body capability allows for analysis of multiple deforming workpieces or coupled die stress analysis. (2D, Pro, 3D).
- Fracture initiation and crack propagation models based on well known damage factors allow modeling of shearing, blanking, piercing, and machining (2D).

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 of the 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, temperature controlled transformation, latent heat of transformation, transformation plasticity, transformation strains, stress effects on transformation, and carbon content effects on all material properties.

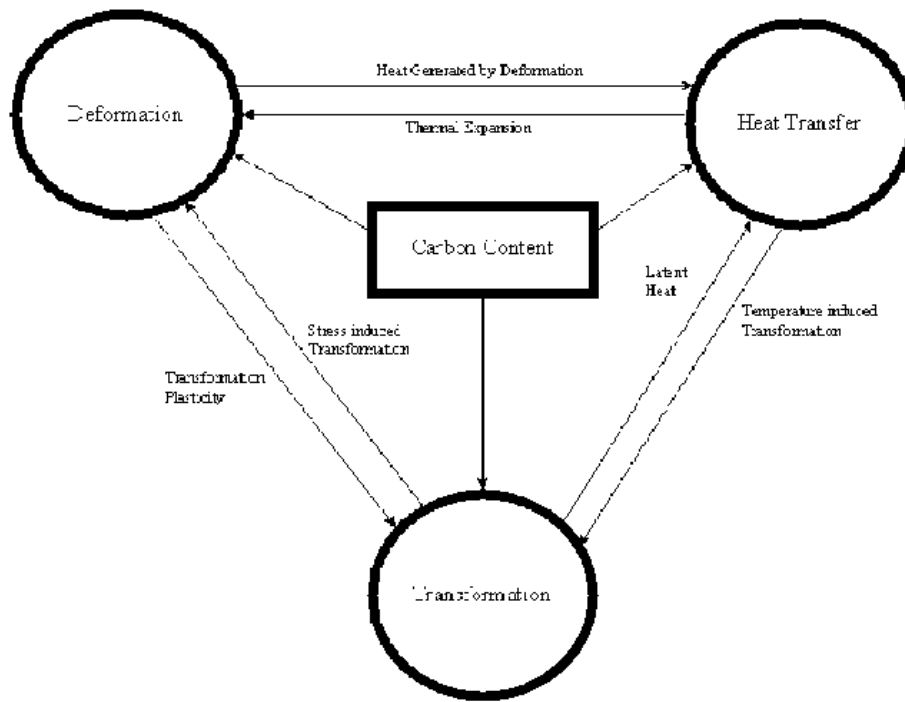


Figure 1: Relationship between various DEFORM modules.

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 workpiece in DEFORM, then sequentially simulate each process that is to be applied to the workpiece.

The recommended sequence for designing a manufacturing process using DEFORM

- Define your proposed process
 - Final forged part geometry
 - Material
 - Tool progressions
 - Starting workpiece/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

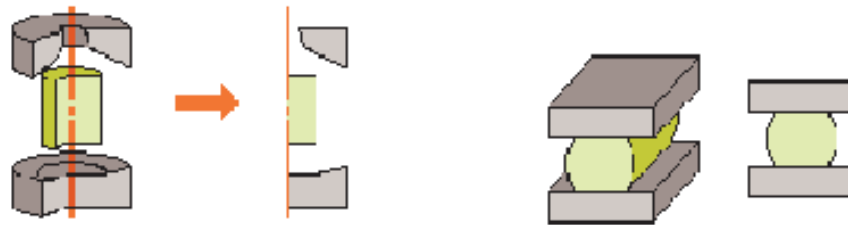


Figure 2: Axisymmetric and plane strain examples.

DEFORM simulations can be run either as two dimensional (2D, PC, Pro) 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 2 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 = $0.525 \cdot (\text{distance across flats})$). A gradually tapering part such as a turbine blade can be modeled by modeling several plane strain sections.

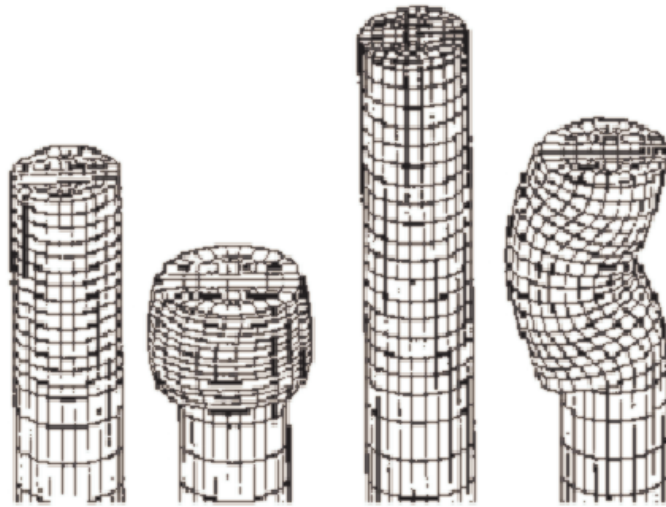


Figure 3: Buckling.

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 3). 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:

- A *pre-processor* for creating, assembling, or modifying the data required to analyze the simulation, and for generating the required database file.
- 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 workpiece whenever necessary. While the simulation engine is running, it writes status information, including any error messages, to the message (.MSG) and log (.LOG) files.
- 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, bainite, 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 interobject 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 are 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 4. The pre-processor uses an ASCII format file called the keyword file to create inputs.

-1	1	2	3	4	5	6	7	8	-9	9	10
I N P U T					S O L U T I O N			S O L U T I O N	I N P U T		S O L U T I O N

Figure 4: 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 (ie, 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	mm ²	in ²	6.4516e2
Volume	mm ³	in ³	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/in ² /sec	1.6353e3
Heat Capacity	N/mm ² /C	BTU/in ³ /F	1.1589e2
Convection Coefficient	N/sec/mm/C	BTU/sec/in ² /F	2.943e3
Lubricant Heat Transfer Coefficient	N/sec/mm/C	BTU/sec/in ² /F	2.943e3

Figure 5: 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.

In Version 3.1, 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

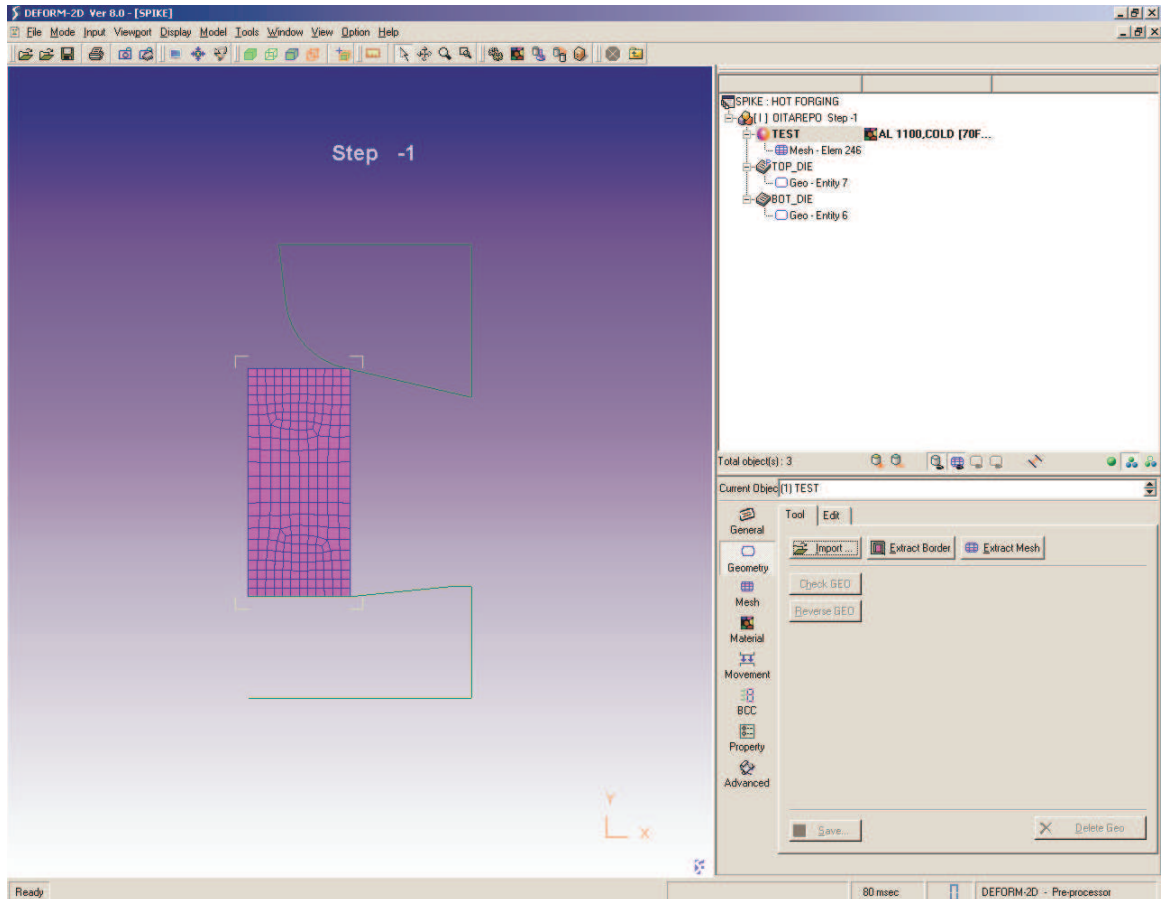


Figure 6: The DEFORM-2D Preprocessor.

2.1. Simulation Controls

The *Simulation Controls* window can be found by clicking a button in the Preprocessor (See Figure 6). Options defined under *Simulation Controls* (See Figure 7) 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.

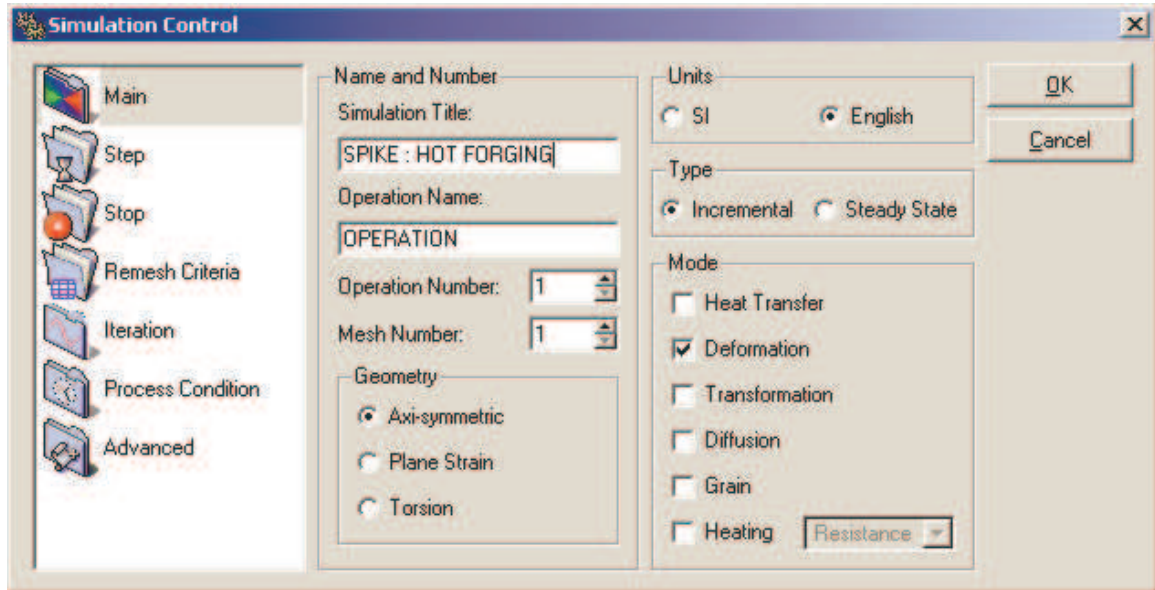


Figure 7: Simulation Control window.

2.1.1. Main controls

Simulation title (TITLE)

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

Simulation name (SIMNAM)

The simulation 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.

Entity	SI unit	English unit	SI unit = English unit * factor
Time	second	second	1.0
Length	mm	in	25.4
Area	mm ²	in ²	6.4516e2
Volume	mm ³	in ³	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/in ² /sec	1.6353e3
Heat Capacity	N/mm ² /C	BTU/in ³ /F	1.1589e2
Convection Coefficient	N/sec/mm/C	BTU/sec/in ² /F	2.943e3
Lubricant Heat Transfer Coefficient	N/sec/mm/C	BTU/sec/in ² /F	2.943e3

Figure 8: The DEFORM units system.

Geometry type (GEOTYP)

Two geometry models are currently available :

1. Axisymmetric models the object as a cross-section with respect to the central axis. Therefore, the model requires the deforming geometry to be axially symmetric and in the first quadrant and fourth quadrant (ie. $X > 0$). In addition, the system assumes that the flow in every radial plane is identical. (Figure 9)
2. The plane-strain model assumes that the geometry to have an unit depth with both front and back faces constrained. The simulation assumes that the objects will behave identically in any given cross-section across the width and height of the object. (Figure 9)

Simulation modes (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 recrystallization and grain growth.

Heating

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



Figure 9: Axisymmetric and plane strain cases.

For compatibility with old keywords and databases, before version 6.0, the keyword SMODE (old style isothermal, non-isothermal, heat transfer) is read and the corresponding TRANS mode switches are set in the pre-processor.

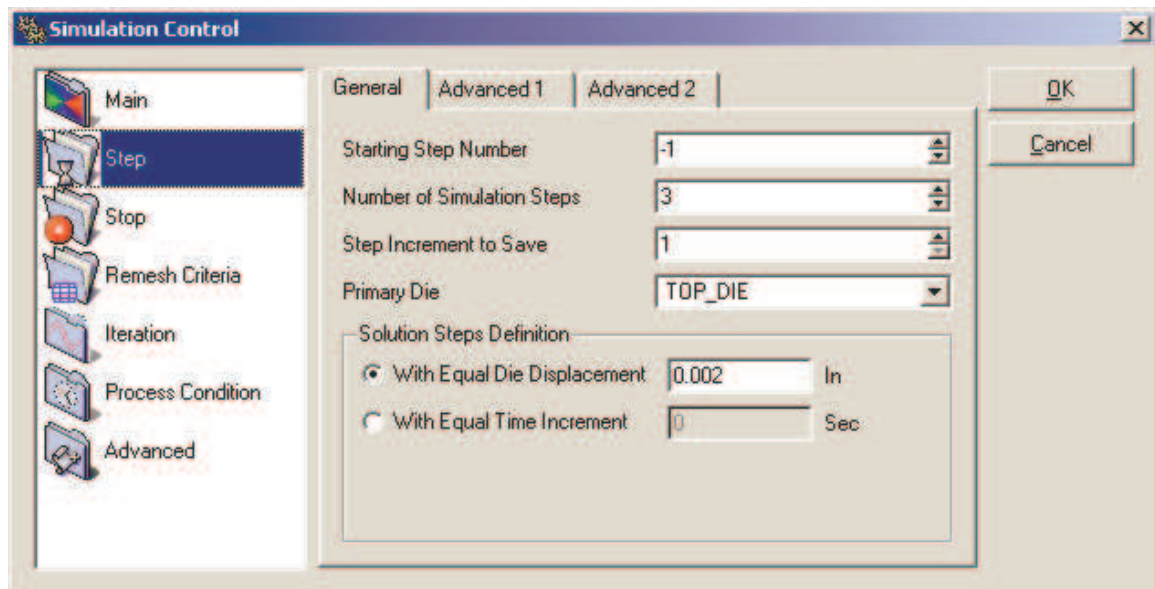


Figure 10: Simulation step controls window.

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 and thermomechanical properties of the workpiece materials. 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.

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, not by the simulation engine.

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

Number of simulation steps (NSTEP)

The number of simulation steps to be run 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, or if another stopping control is triggered to stop the simulation. Hence 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 the stroke is measured for any values which refer to the die stroke. For example, stopping criteria based on primary die stroke, or die velocity which is specified as a function of stroke are both keyed to the primary die movement.

The primary die is usually assigned to the object most closely associated 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 times 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.

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 or rapid mesh distortion. Too small a time step can lead to unnecessarily long solution times. The following section provides some guidelines for selecting time steps.

For typical two dimensional simulations of forming operations, 100 steps is generally adequate. For simple simulations, such as squaring up a cutoff billet, as few as 25-50 steps may be appropriate.

For complex or semi-continuous processes such as extrusion or drawing, more steps may be required. For these simulations, 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 workpiece velocity (for most problems, this will be the die velocity. For extrusion problems it will be the die velocity times the extrusion ratio)
3. Divide the result of 1 by the result of 2, and take about 1/2 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 $n = \frac{x}{V\Delta t}$ where n is the number of steps, x is the total movement of the primary die, V is the primary die velocity, and Δt is the time increment per step.

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, then specify about 120% of this value. Allow the simulation to overshoot the target, then use a step near, but not at the end as a final solution.

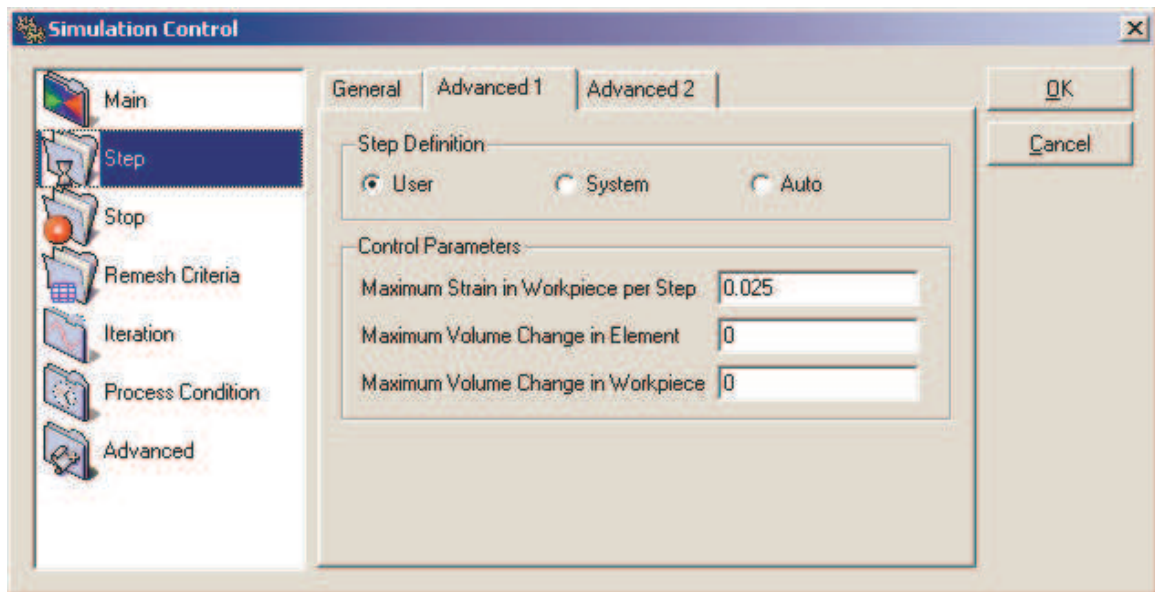


Figure 11: Advanced stepping menu 1.

2.1.3. Advanced Step Controls

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.

Volume change per step (DVMAX)

During a deformation time step, elements typically experience a change in volume. Over time, this volume change generally results in volume loss. The volume loss generally increases with increasing step sizes and increasing total

height reduction ($\Delta H/H$) where ΔH refers to a height reduction per step, and H refers to the height of an object.

For problems where volume loss is significant, the volume loss also can be controlled by specifying the maximum amount of volume change that an

individual element or an object can experience during a time step. If a non-zero value is assigned, a new sub step will be initiated when the ratio of the volume change to the original volume of any element exceeds the specified value.

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.

Sliding error (SLDERR)

The sliding error is used to control the movement of nodes of a slave object which are in contact with a master object. Sliding error limits the distance a contacting node of a slave object can move from the adjacent master object in a single time step. The value of sliding error should be between 5% and 10% of the smallest side length of the smallest element.

Slave nodes which are in contact with a master surface slide along the master surface as the object is deformed. However, when the slave node approaches a corner on the master surface, the direction of the nodal velocity may cause the node to shoot past the corner during a time step. Once the node has separated from the master surface, it will continue to move in the direction of its separation velocity until the time step is completed. When a new time step is generated, the node is forced back onto the master surface along the shortest normal connecting the node and the master surface. The length of this normal is referred to as the normal distance error. The sliding error limit causes a new time step to be initiated whenever a slave node's normal distance error exceeds the specified value for SLDERR.

An alternate approach to this problem is to specify the contact release method (OSCTRL) under *Simulation Controls, Advanced Controls*.

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.
- **Auto** In temperature based sub stepping the DTPMAX value controls the time stepping.

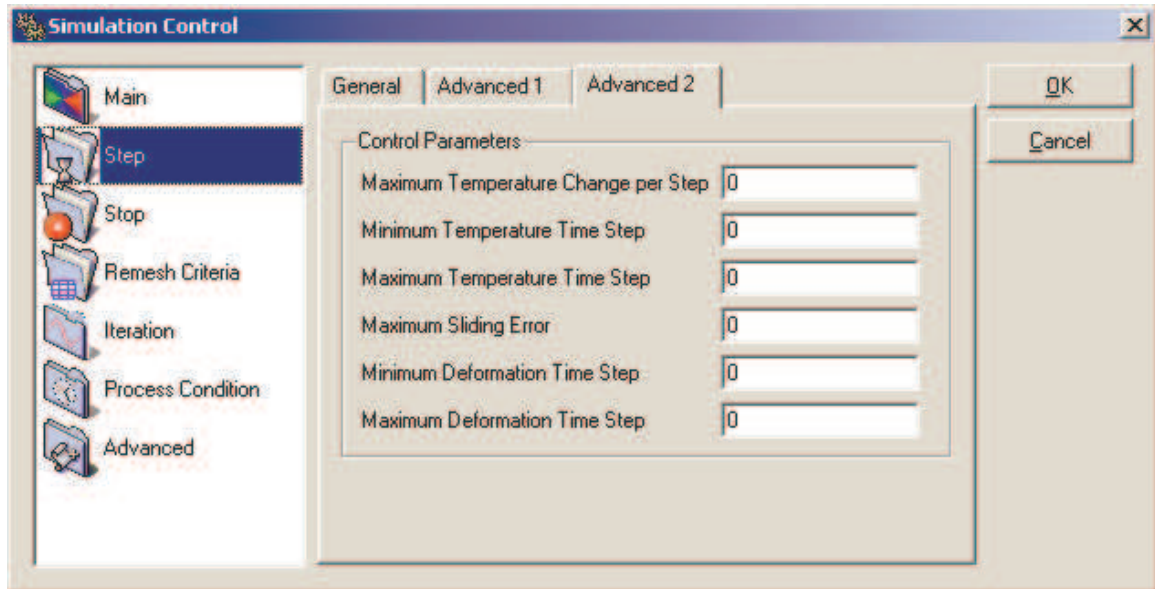


Figure 12: Advanced stepping menu 2.

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.

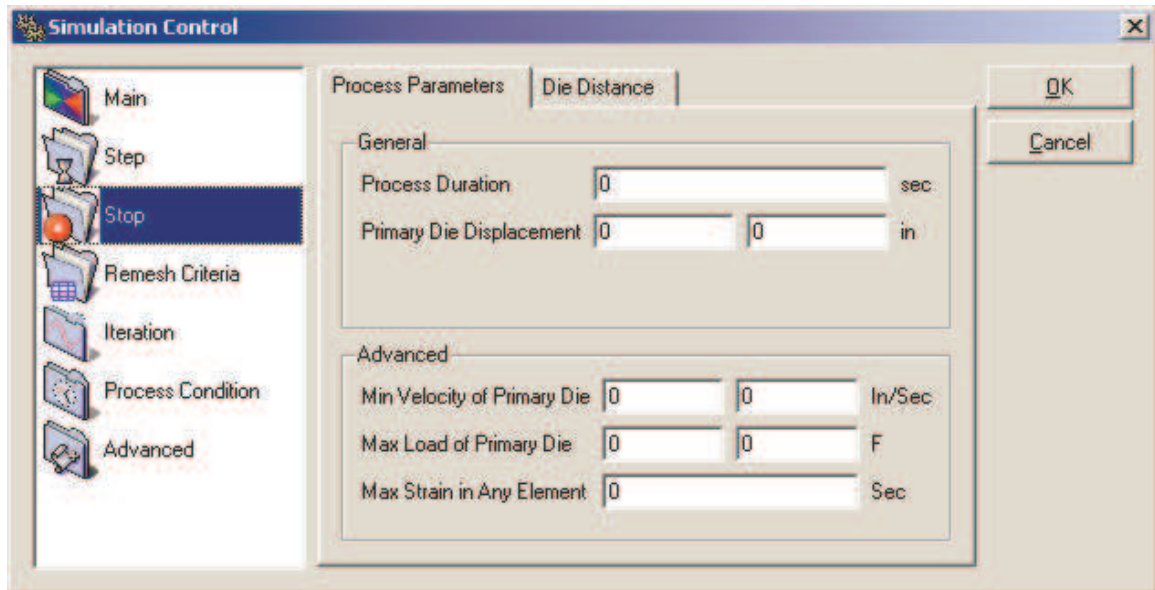


Figure 13: Stopping controls window.

Maximum strain (EMAX)

Terminates a simulation when the accumulated strain of any element reaches the specified value

Maximum process time (TMAX)

Terminates a simulation when the global process time reaches the value specified.

Maximum stroke (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 (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 (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.

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 on the *Objects, Properties* menu.

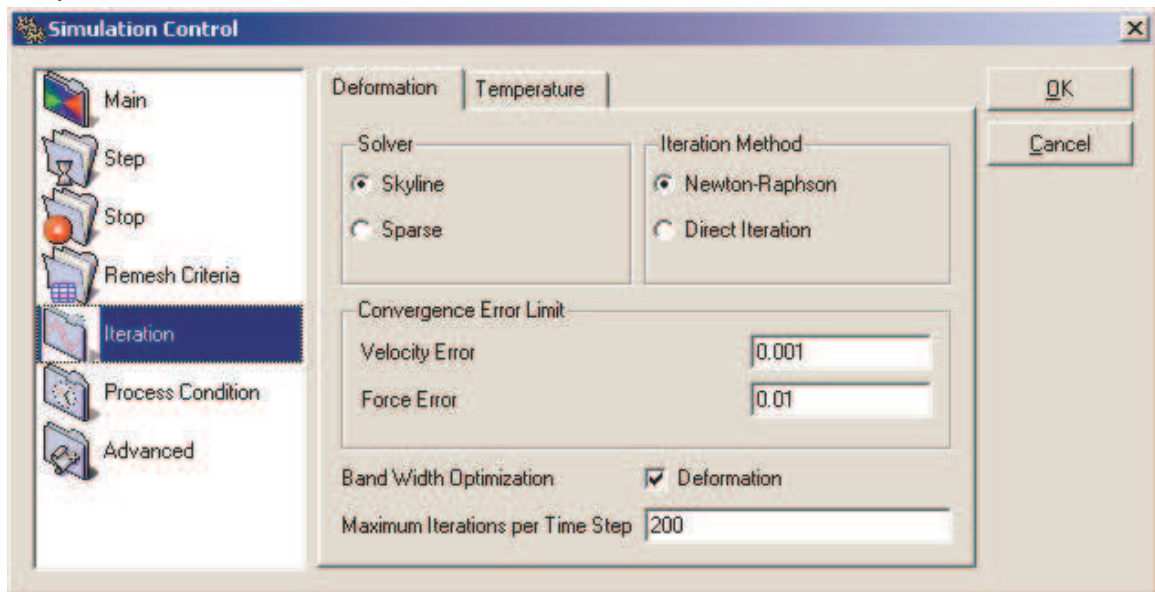


Figure 14: Iteration controls window.

2.1.5. 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.

Iteration methods (ITRMTH)

- **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 iterations to do so.

Temperature solver (SOLMTT)

The skyline solver uses the skyline storage method in conjunction with Gaussian elimination to store temperature matrix data. This method is recommended for most 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 multiple deforming body problems.

Convergence error limits (CVGERR)

A deformation iteration is assumed to have converged when the velocity and force error limits have been satisfied. The error norm values for each iteration step are displayed in the message file.

If the message file shows that the force and velocity error norms are getting small, but not dropping below the error limits, the simulation may be continued by increasing the error limits 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.

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 remembered that the accuracy of the flow stress data will have more impact on the accuracy of die stress and press load predictions.

Maximum number of iterations (ITRMXD,ITRMXT)

When Newton-Raphson iteration is being used, the specified number of iterations will be performed for each iteration segment until the solution has converged. At most, 30 iterations will be performed during a Newton-Raphson segment. If the solution does not converge in the specified number of iterations, the simulation will terminate and a message will be written to the DEFORM message file.

If direct iteration is specified as the iteration method, the specified number of iterations will be performed. If the solution has not converged, another series of iterations will be performed. If the solution has still not converged, the simulation will terminate, and a message will be written to the DEFORM message file.

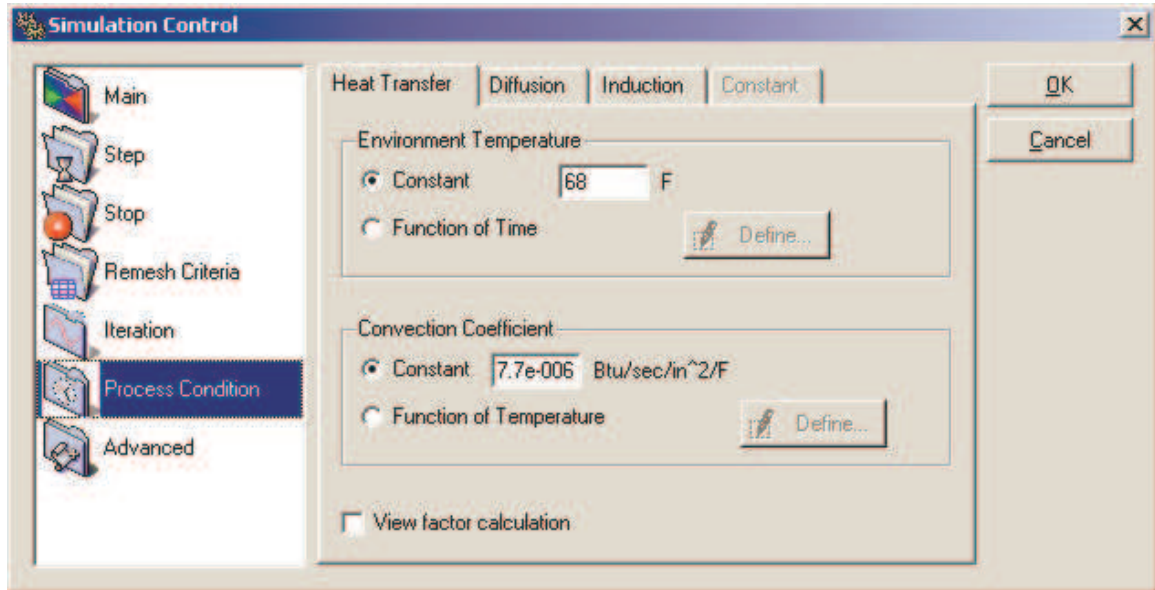


Figure 15: Process condition window (heat transfer data).

2.1.6. Processing conditions

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

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 shape factors are accounted for.

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.

View Factor calculation

This is a checkbox that will perform self-heating and radiation heating from other bodies when the view factor box is checked.

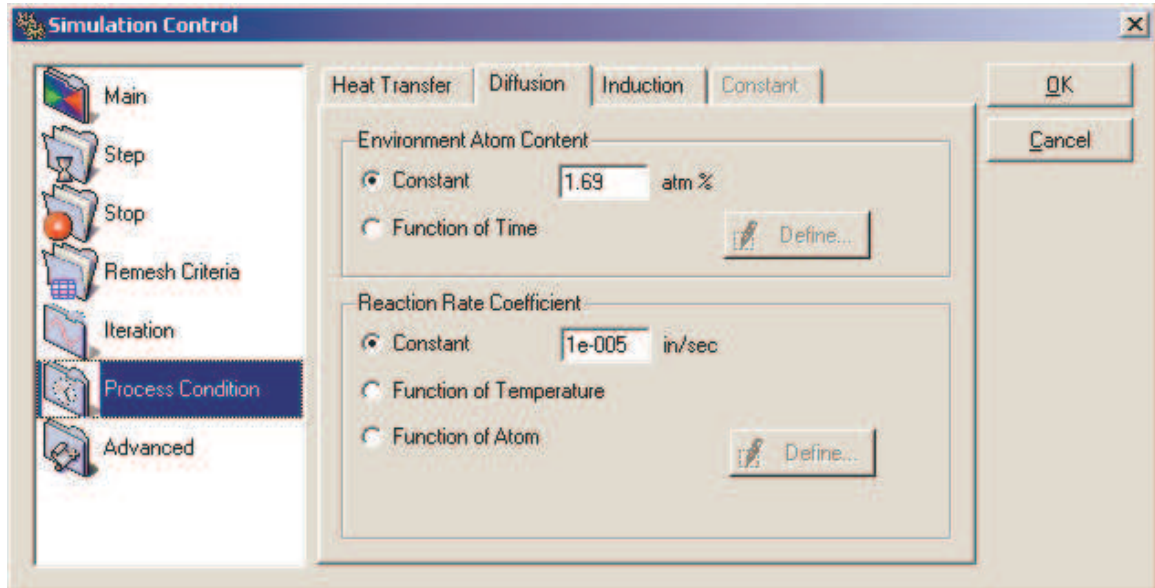


Figure 16: Process condition window (diffusion data).

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.

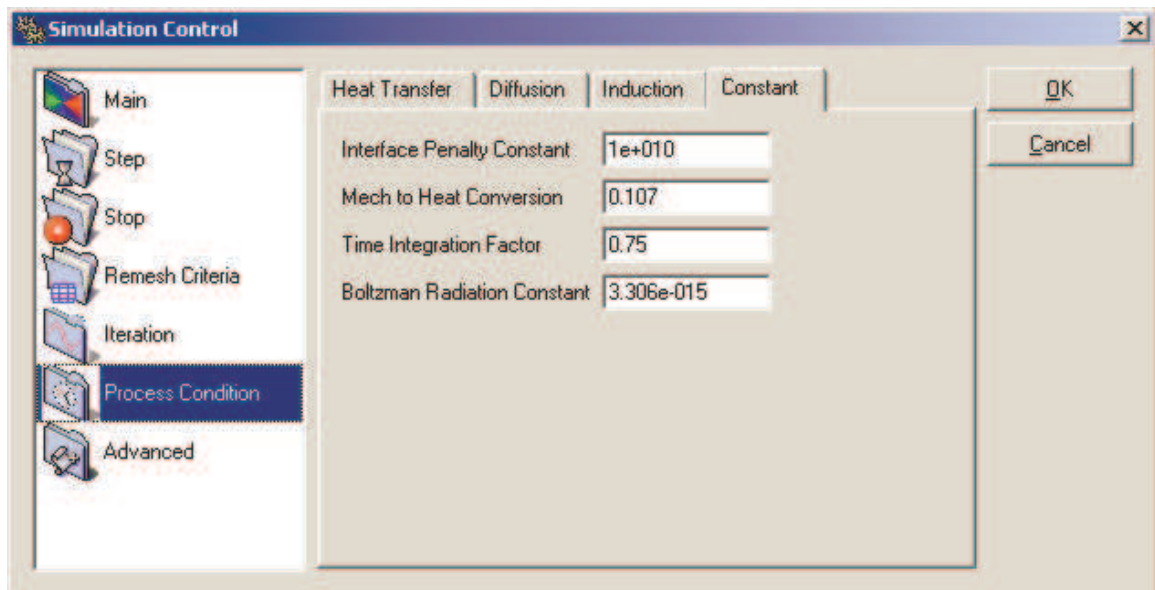


Figure 17: Process condition window (constants).

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.

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).

Mechanical to heat conversion (UNTE2H)

A constant coefficient to relate units of heat energy (e.g. BTU) to mechanical energy (e.g. 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.

2.1.7. Advanced controls

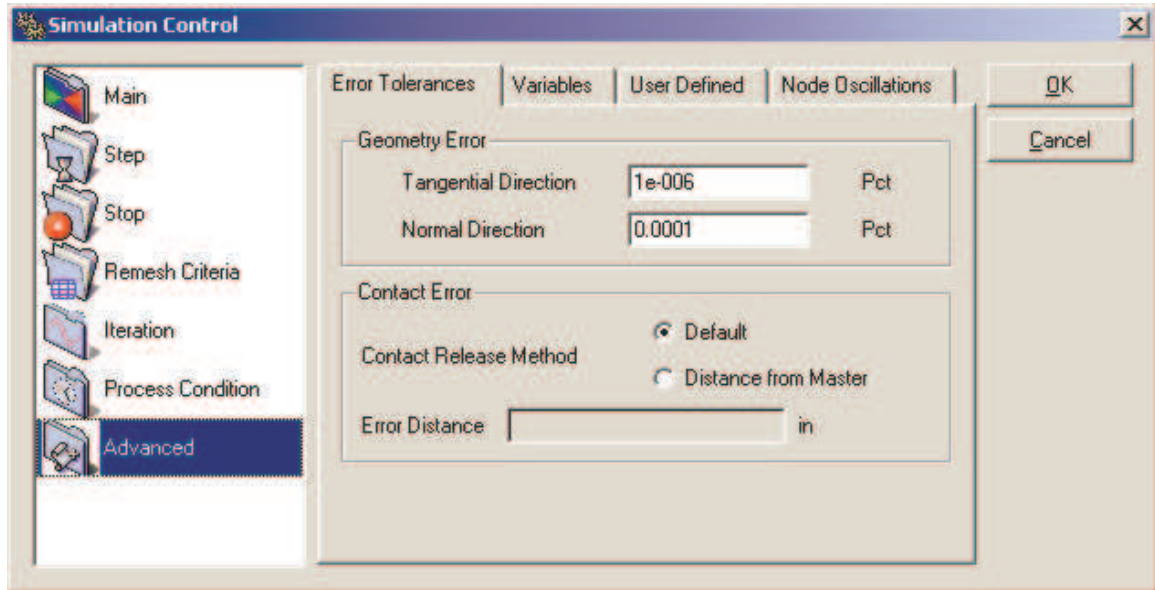


Figure 18: Simulation controls, advanced window (error tolerances).

Error tolerances

Geometry error (GEOERR)

This value is an estimate of the error between discretized objects. The default value for this is sufficient.

Contact release method (CNTERR)

In certain cases the present contact algorithm does not release nodes that are touching a master surface within the time step. This option allows the contact condition for a slave node to be released if it moves away from the master boundary by a prescribed distance. This value can be used as an alternative to the sliding error (SLDERR).

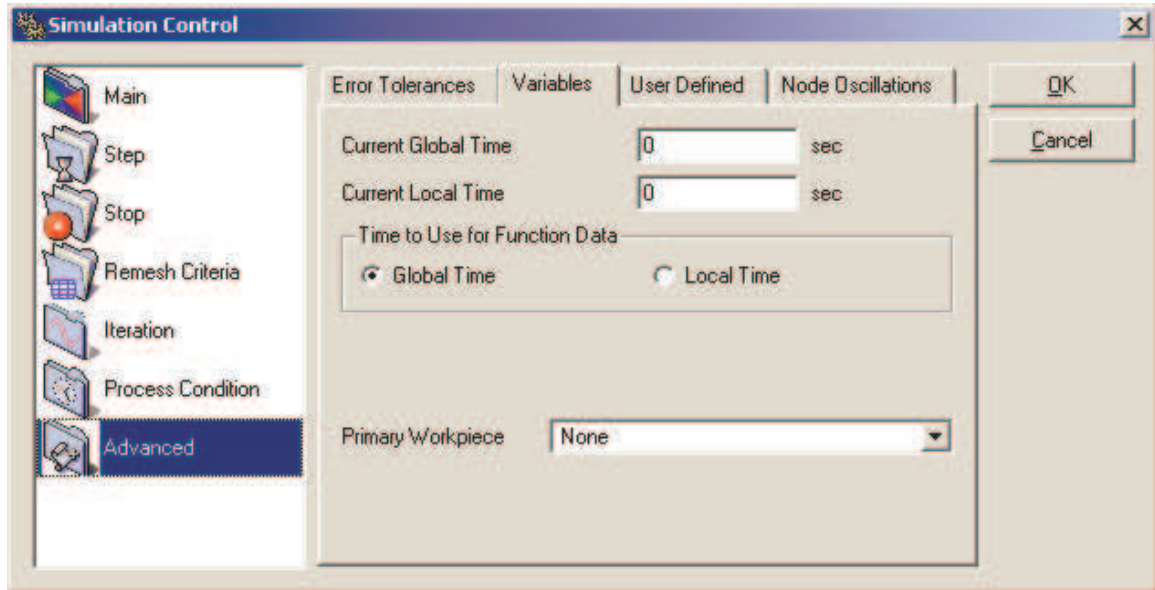


Figure 19: Simulation controls, advanced window (variables).

Current time (TNOW)

This value specifies the current process time and the local process time. The global time should not be reset during a simulation as the post-processor uses this time for many post-processing operations.

Primary workpiece (PDIE)

The deforming object that is used to pass average strain rate (AVGSTR) data to the user routines for control of die speed based on the strain rate in the workpiece.

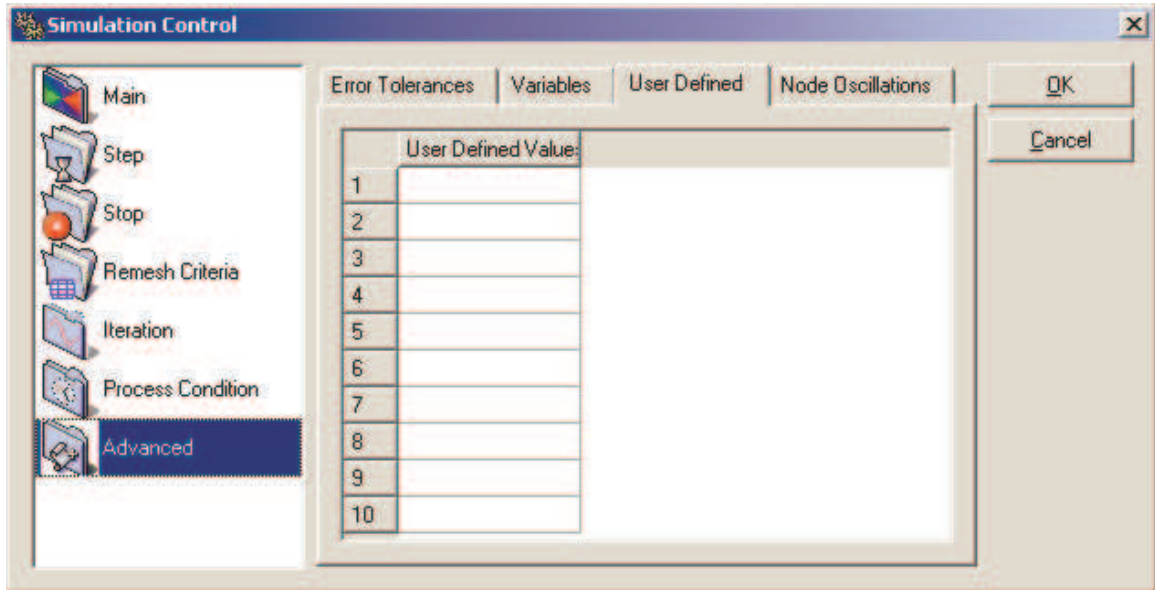


Figure 20: Simulation controls, advanced window (user defined).

User defined variables (USRDEF)

User defined variables are 80 character string variables which are passed to user defined subroutines. Refer to User Routines for more information on how to use these variables.

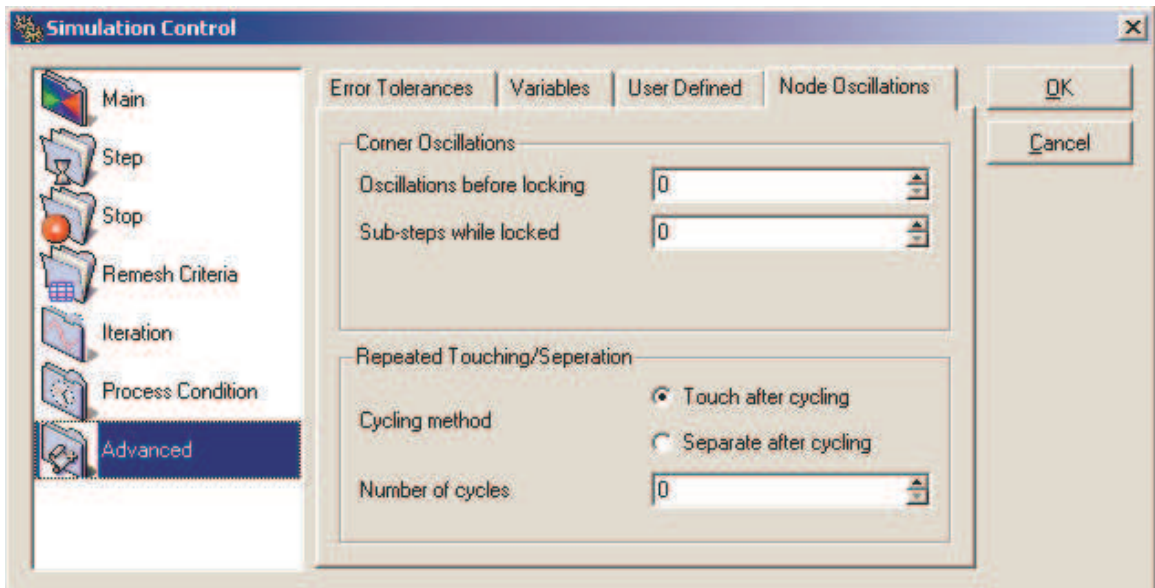


Figure 21: Simulation controls, advanced window (node oscillations).

Nodal oscillations (OSCTRL)

Corner oscillations

Corner oscillation controls are used to control oscillation of a node between adjacent line segments of a surface. If the number of oscillations exceeds the limit, the node will be locked for a fixed number of sub steps.

Repeated touching/separating

When slave nodes touch and separate from a master surface, after two oscillations the nodes are made to touch for the sub step. The touching/separating control can be used to make the node separate from the surface for the sub step after a specified number of oscillations.

2.2. Material Data

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 model
- Hardness data
- Fracture Data (FRCMOD)

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.

2.2.1. Phases and mixtures (MSTMTR) [MIC]

Material groups can be classified into two categories, phase materials and mixture materials. 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.

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.

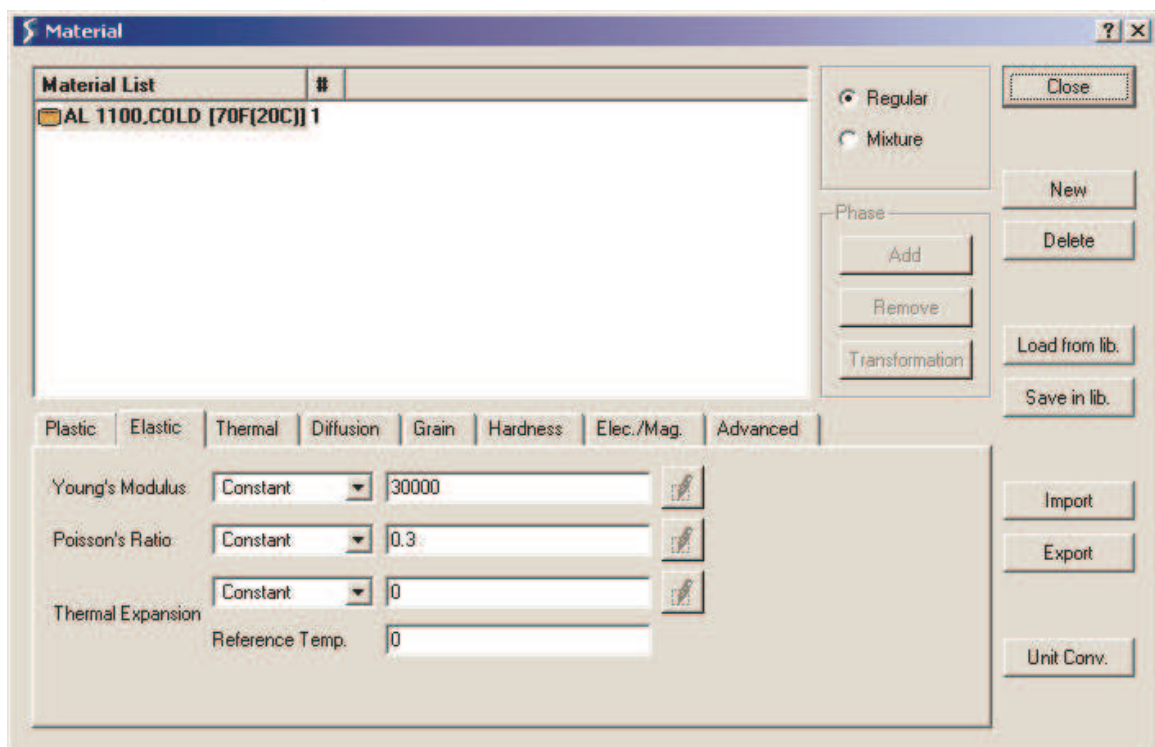


Figure 22: Material data window (elastic).

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) :

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

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

For elasto-plastic 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.

$$\epsilon^{th} = \alpha^* \Delta T + \sum \Delta \epsilon^{th_{prev}}$$

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

2.2.3. Thermal data

Thermal data is required for any object in the heat transfer mode.

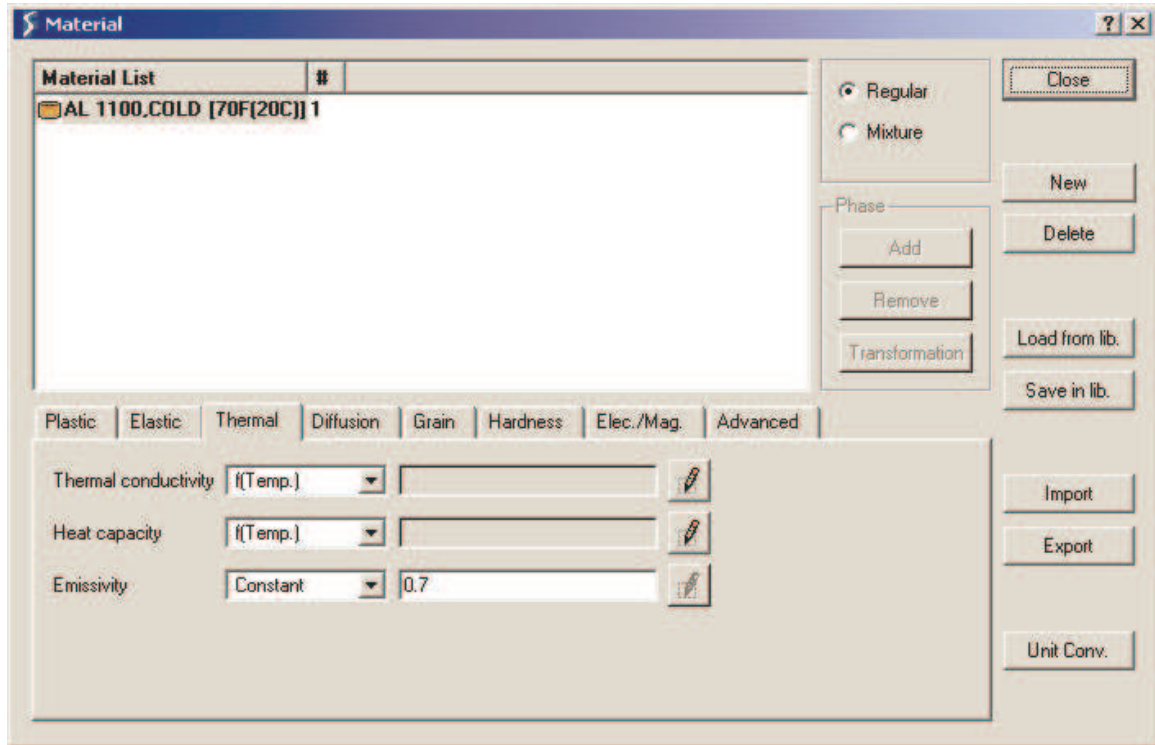


Figure 23: Material data window (thermal).

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.

Emmissivity (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, ϵ , 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.

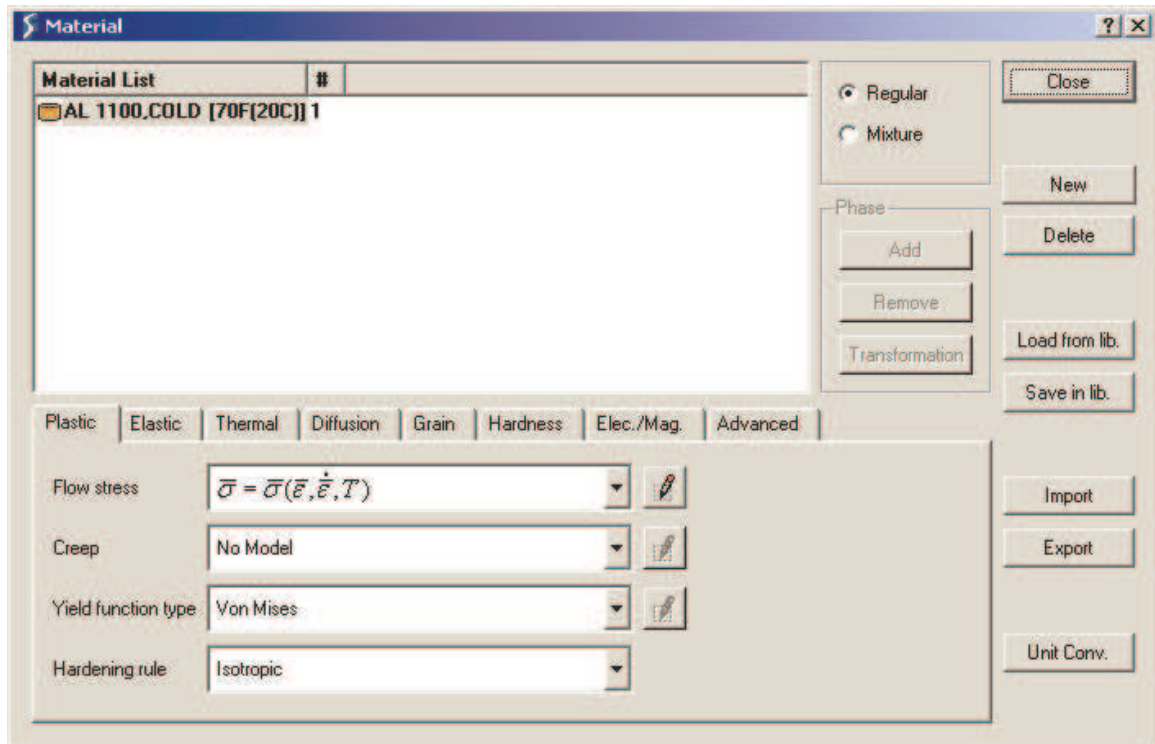


Figure 24: Material data window (plastic).

2.2.4. Plastic Data

For studying the plastic deformation behaviour 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. 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.

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 aluminium alloys (Type 1)

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

where

A = Constant

α = Flow stress

n = Strain rate exponent

ΔH = Activity Energy

R = Gas constant

T_{abs} = Absolute temperature

$\bar{\sigma}$ = Flow stress

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

Flow stress for aluminium alloys (Type 2)

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

where

A = Constant

n = Strain rate exponent

ΔH = 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 User Routines 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 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.

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 a 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.

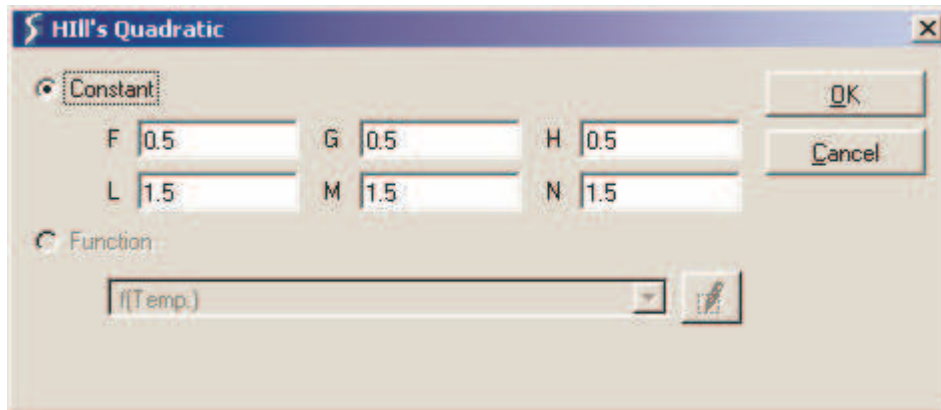


Figure 25: Hill's quadratic (FGHLMN) input screen.

Hill's quadratic (R value)

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

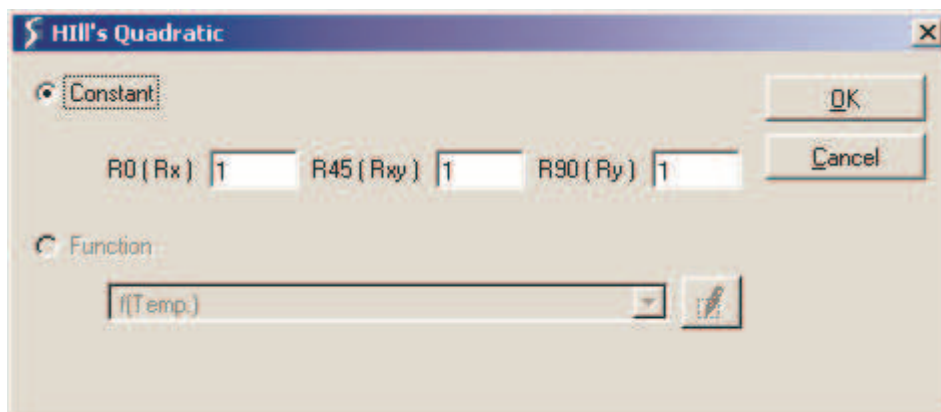


Figure 26: Hill's quadratic (R value) input screen.

Hardening model (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

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

where

γ = fluidity

$\bar{\sigma}$ = effective stress

S = Flow stress

m = Material parameter

$\dot{\epsilon}$ = 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

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

where

γ = fluidity

$\bar{\sigma}$ = effective stress

S = Flow stress

m = Material parameter

$\dot{\epsilon}$ = 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

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

where

$\bar{\sigma}$ = Effective stress

T_{abs} = Absolute temperature

K,n,m,Q,r = Constants

γ = fluidity

S = Flow stress

$\dot{\epsilon}$ = 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

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

where

$\bar{\sigma}$ = Effective stress

T_{abs} = Absolute temperature

K,n,C = Constants

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

Tabular data

This method is not currently available for this release.

User Routines

Please refer to User Routines for a description of how to implement user defined creep routines.

2.2.5. Diffusion data

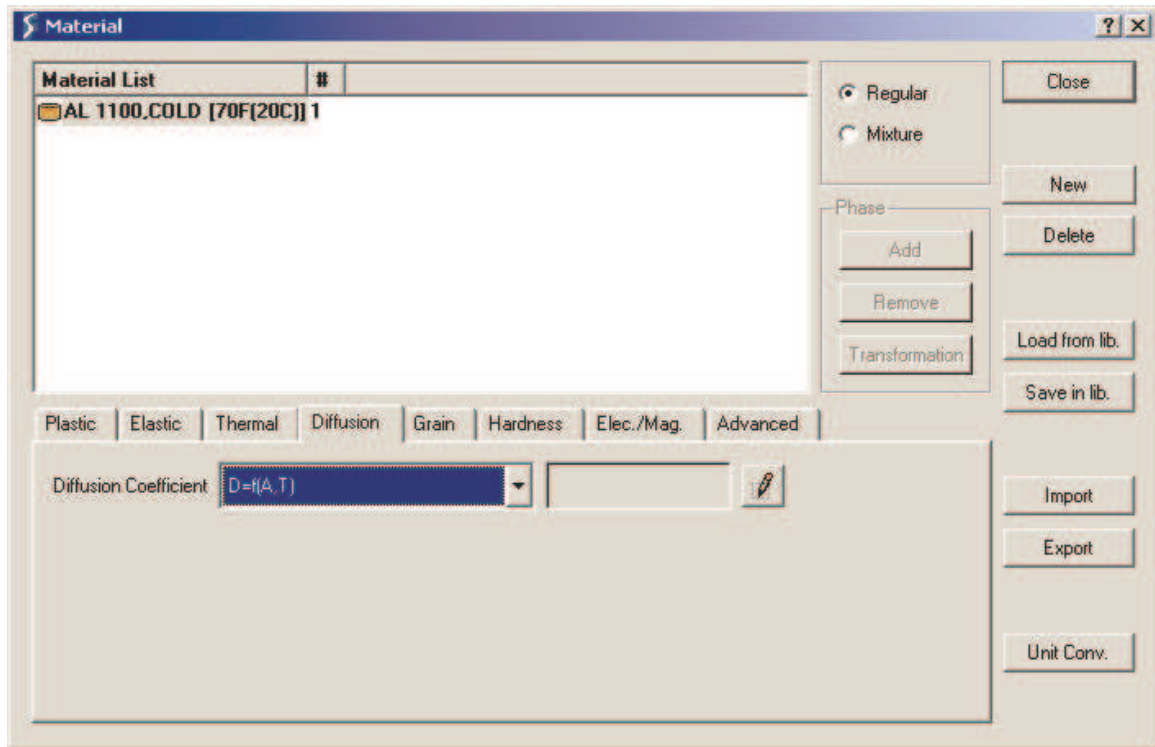


Figure 27: Material properties window (diffusion).

DEFORM allows the user to model the diffusion of the dominant atom (at this point carbon) in an object. 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.

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).

$$D=f(A,t) \quad (6.13)$$

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).

$$D=C_1(T)\exp(C_2(T)A) \quad (6.14)$$

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).

$$D=C_1(A)\exp(C_2(A)/T) \quad (6.15)$$

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

2.2.6. Grain growth/recrystallization model

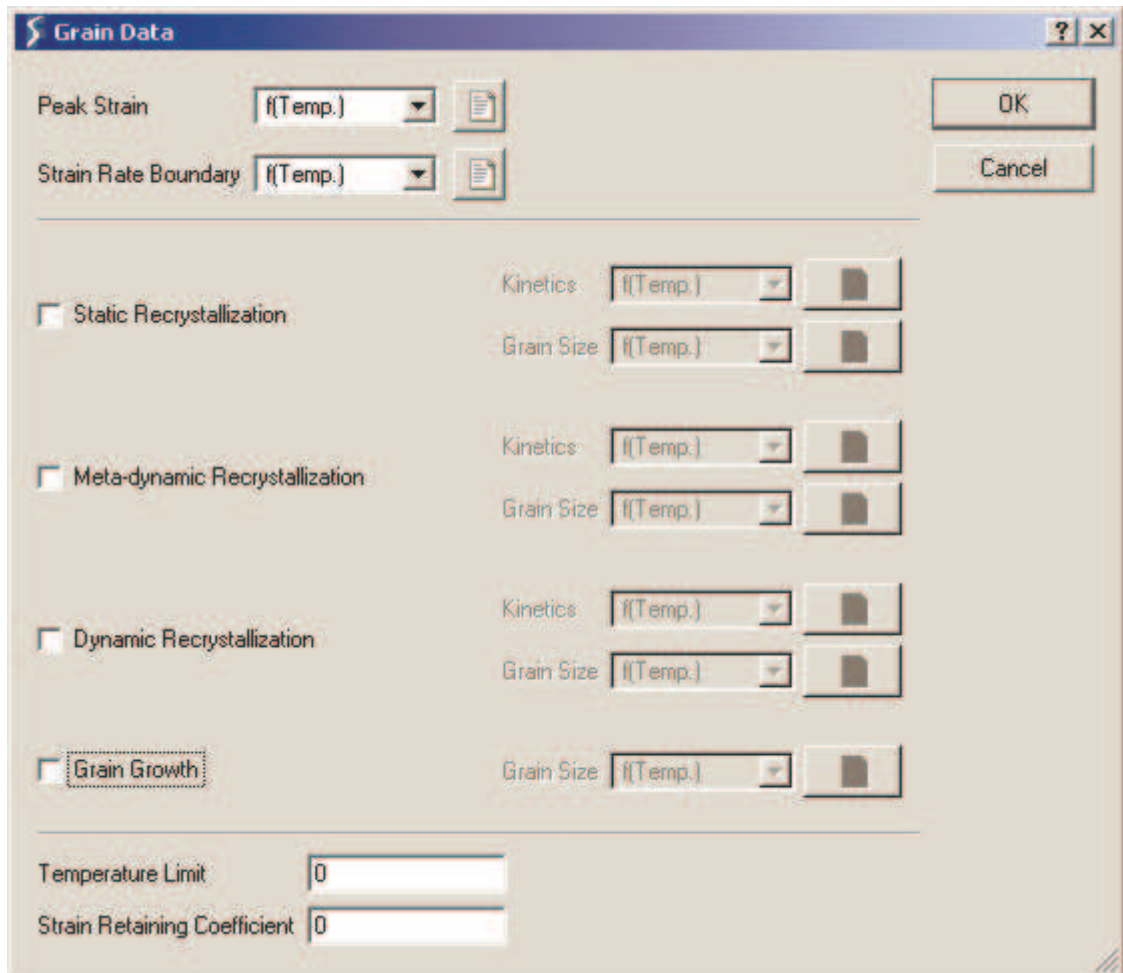


Figure 28: Material properties window (grain growth).

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.

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.

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 strain ϵ_c .

$$\epsilon_c = \alpha_2 \epsilon_p \quad (1)$$

where ϵ_p denotes the strain corresponding to the flow stress maximum:

$$\epsilon_p = \alpha_1 d_0^{n_1} \dot{\epsilon}^{m_1} \exp(Q_1 / RT) + c_1 \quad (2)$$

2. Kinetics

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

$$X_{drx} = 1 - \exp \left[- \beta_d \left(\frac{\epsilon - \alpha_{10} \epsilon_p}{\epsilon_{0.5}} \right)^{k_d} \right] \quad (3)$$

where $\epsilon_{0.5}$ denotes the strain for 50% recrystallization:

$$\epsilon_{0.5} = \alpha_3 d_0^{n_3} \dot{\epsilon}^{m_3} \exp(Q_3 / RT) + c_3 \quad (4)$$

3. Grain Size

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

$$d_{rex} = \alpha_8 d_0^{k_8} \epsilon^{n_8} \dot{\epsilon}^{m_8} \exp(Q_8 / RT) + c_8 \quad (\text{if } d_{rex} \geq d_0 \text{ then } d_{rex} = d_0) \quad (5)$$

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 $\dot{\epsilon}_{ss}$, static recrystallization occurs after deformation.

$$\dot{\epsilon}_{ss} = A \exp(b_1 - b_2 d_0 - Q_2 / T) \quad (6)$$

2. Kinetics

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

$$X_{srex} = 1 - \exp \left[- \beta_s \left(\frac{t}{t_{0.5}} \right)^{k_s} \right] \quad (7)$$

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

$$t_{0.5} = a_3 d_0^{k_3} \epsilon^{n_3} \dot{\epsilon}^{m_3} \exp(Q_3 / RT) \quad (8)$$

3. Grain Size

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

$$d_{rex} = a_6 d_0^{k_6} \epsilon^{n_6} \dot{\epsilon}^{m_6} \exp(Q_6 / RT) + c_6 \quad (\text{if } d_{rex} \geq d_0 \text{ then } d_{rex} = d_0) \quad (9)$$

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 $\dot{\epsilon}_{ss}$ (see equation (1)), meta-dynamic recrystallization occurs after deformation.

2. Kinetics

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

$$X_{mdrex} = 1 - \exp \left[- \beta_{md} \left(\frac{t}{t_{0.5}} \right)^{k_{md}} \right] \quad (10)$$

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

$$t_{0.5} = a_4 d_0^{k_4} \epsilon^{n_4} \dot{\epsilon}^{m_4} \exp(Q_4 / RT) \quad (11)$$

3. Grain Size

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

$$d_{rex} = a_7 d_0^{k_7} \varepsilon^{n_7} \dot{\varepsilon}^{m_7} \exp(Q_7 / RT) + c_7 \text{ (if } d_{rex} \geq d_0 \text{ then } d_{rex}=d_0) \text{ (12)}$$

Grain Growth

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

The kinetics is described by equation:

$$d_r = \left[d_{rex}^m + a_9 t \exp\left(-\frac{Q_9}{RT}\right) \right]^{1/m} \text{ (13)}$$

where d_{gg} denotes the grain size after growth, a_9 and m are materials constant, and Q_9 is an activation energy.

Retained Strain

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

$$\varepsilon_i = (1.0 - \lambda X_{rex}) \varepsilon_{i-1} \text{ (14)}$$

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,

$$d = X_{rex} \cdot d_{rex} + (1 - X_{rex}) \cdot d_0 \text{ (15)}$$

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
 _Effective strain
 c _Critical strain
 p _Peak strain
 0.5 _Strain for 50% recrystallization
 $\dot{\epsilon}$ _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
 d , md , s _Material data
 k_d, k_{md}, k_s _Material data
 _Material data

2.2.7. Hardness data [MIC]

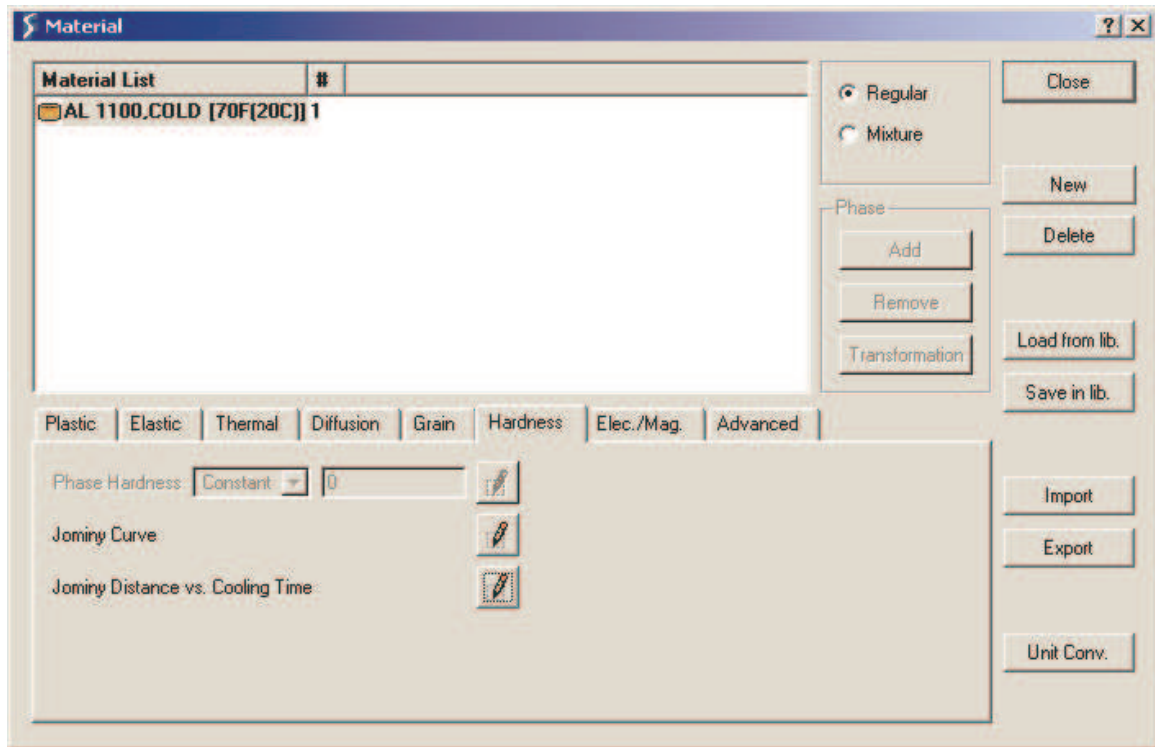


Figure 29: Material data window (hardness).

There are two methods in which the hardness of a object may be determined after a cooling operation. 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.

2.2.8. Advanced Features

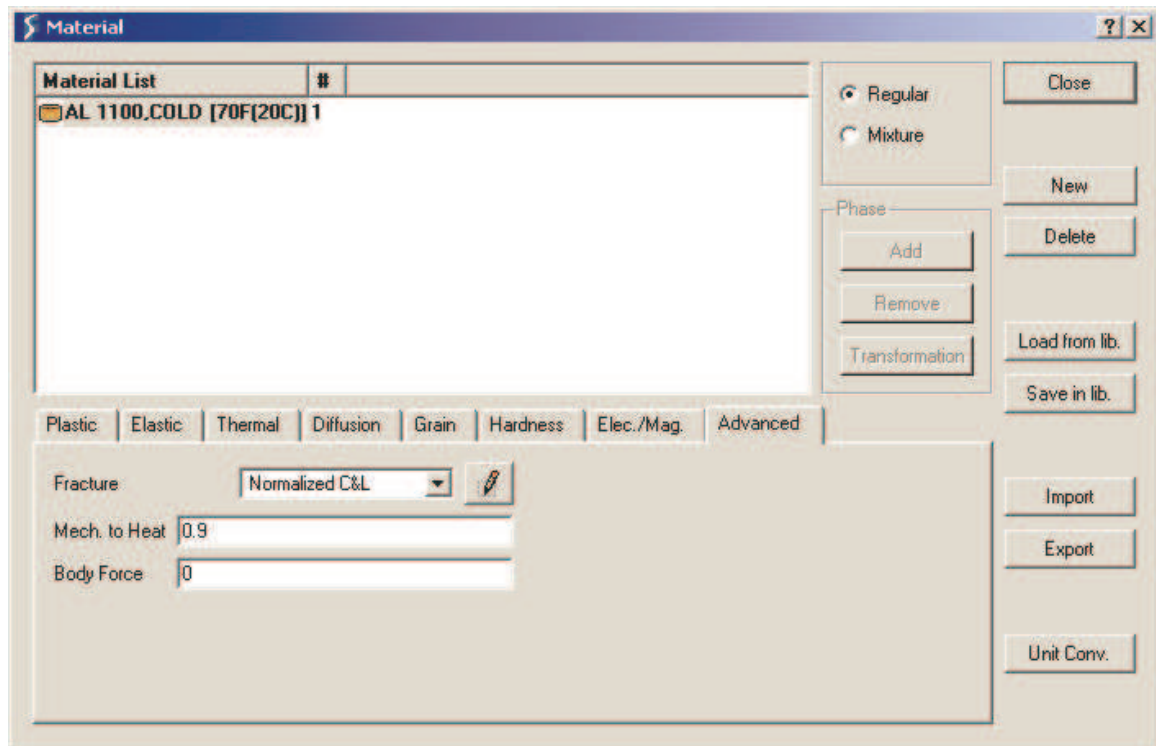


Figure 30: Material properties window (advanced).

Fracture Data (FRCMOD)

FRCMOD specifies the damage model that one wishes to use for damage calculation and for element deletion. Currently, the DEFORM fracture model only supports element deletion, however, in the future element splitting will be implemented. There are 10 different models to choose from. Cockroft & Latham is the damage model that is used to calculate damage in DEFORM. It is also possible to write a user subroutine which can be used for the damage model.

- Cockroft & Latham
- McClintock
- Freudenthal
- Rice & Tracy
- Oyane
- Ayada
- Osakada

- Brozzo
- Zhoa & Huhn
- Maximum effective stress / ultimate tensile strength
- User routine

If the Maximum effective stress / ultimate tensile strength model is selected the ultimate tensile strength (UTSDAT) has to be defined as a constant or a function of temperature.

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.

Force per unit volume (FPERV)

The following equation is used to calculate the body force for porous materials:

$$F = \int \rho f_{vol} dV$$

where F is force with unit of Klb for english and N for SI, f_{vol} is the force per unit volume, and ρ is the material density.

2.2.9. Material data requirements

Guidelines

1.

An isothermal forming problem with rigid dies and a rigid-viscoplastic workpiece

	Workpiece	Dies
Material Data		
Flow Stress	X	
Youngs Modulus		
Poisons Ratio		
Thermal Expansion		
Heat Capacity		
Conductivity		
Emissivity		

2.

A Non-isothermal forming problem with rigid dies and a rigid-viscoplastic workpiece

	Workpiece	Dies
Material Data		
Flow Stress	X	
Youngs Modulus		
Poisons Ratio		
Thermal Expansion		
Heat Capacity	X	X
Conductivity	X	X
Emissivity	X	X

3.

Heat transfer analysis

	Workpiece	Dies
Material Data		
Flow Stress		
Youngs 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 workpiece

	Workpiece	Dies
Material Data		
Flow Stress	X	
Youngs 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

	Workpiece	Dies
Material Data		
Flow Stress	X	
Youngs Modulus		X
Poisons Ratio		X
Thermal Expansion		
Heat Capacity		
Conductivity		
Emissivity		

6.

Decoupled die stress analysis non-isothermal

	Workpiece	Dies
Material Data		
Flow Stress	X	

Youngs 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 is 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 7.1 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 transformation 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) [MIC]

In DEFORM, the manner in which transformation is defined is in terms of transformation relationships. The basic unit for transformation relationships are phases. Phases can be grouped together to define a mixture. A mixture corresponds to a material such as AISI-1045 or Ti-6Al-4V. For each phase defined, there may be a transformation relation 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 needs to 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.

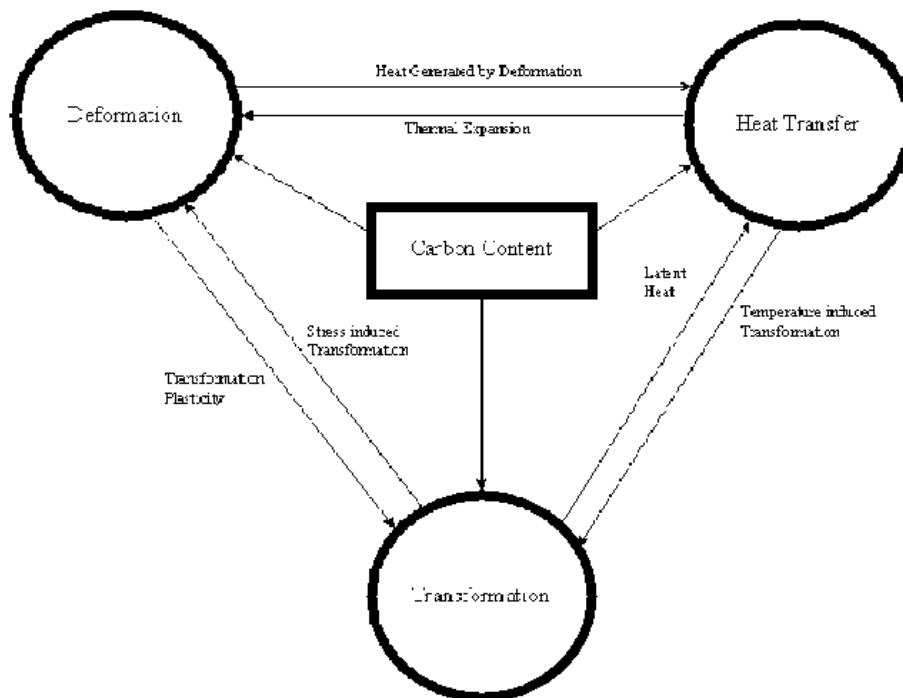


Figure 31: Relationship between various DEFORM modules.

2.3.2. Kinetics model (TTTD) [MIC]

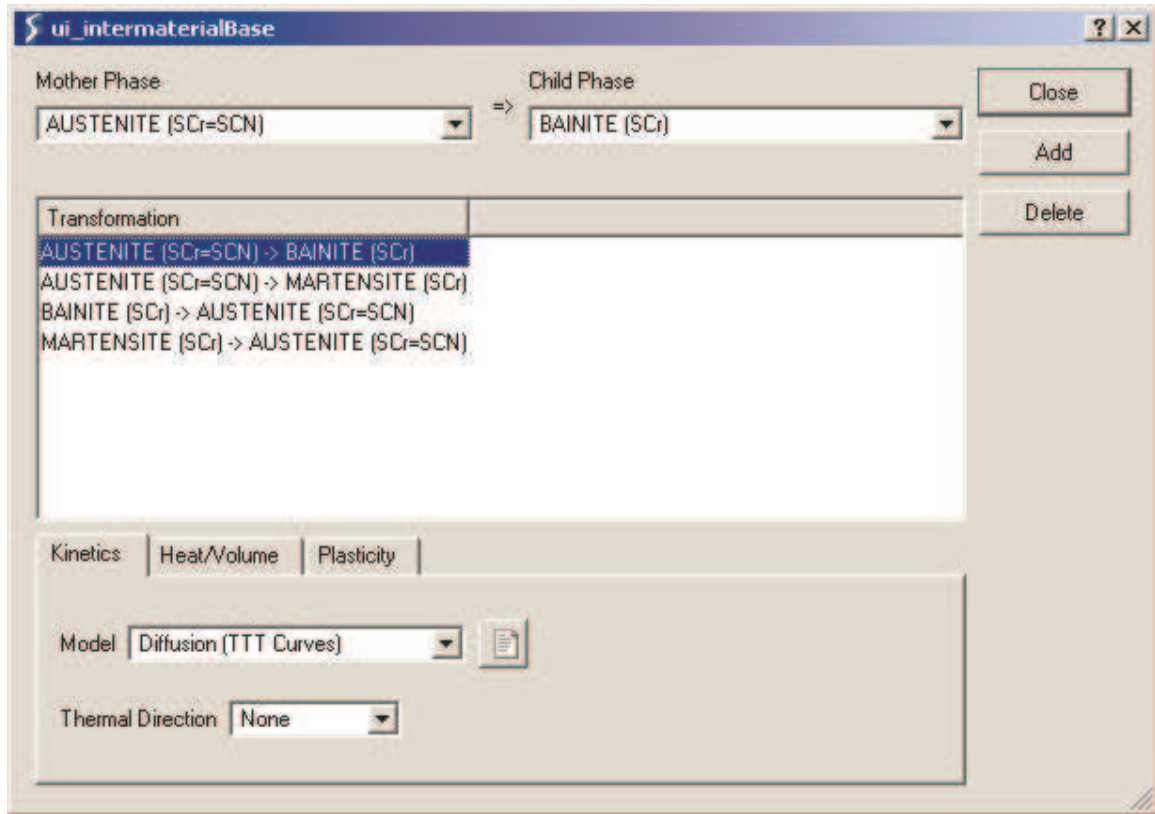


Figure 32: Transformation kinetics definition window.

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. There are two classifications for kinetics models, diffusion-type transformations and diffusionless-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 on the temperature, stress history, and carbon content. The diffusionless 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 is trying to solve 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 number.

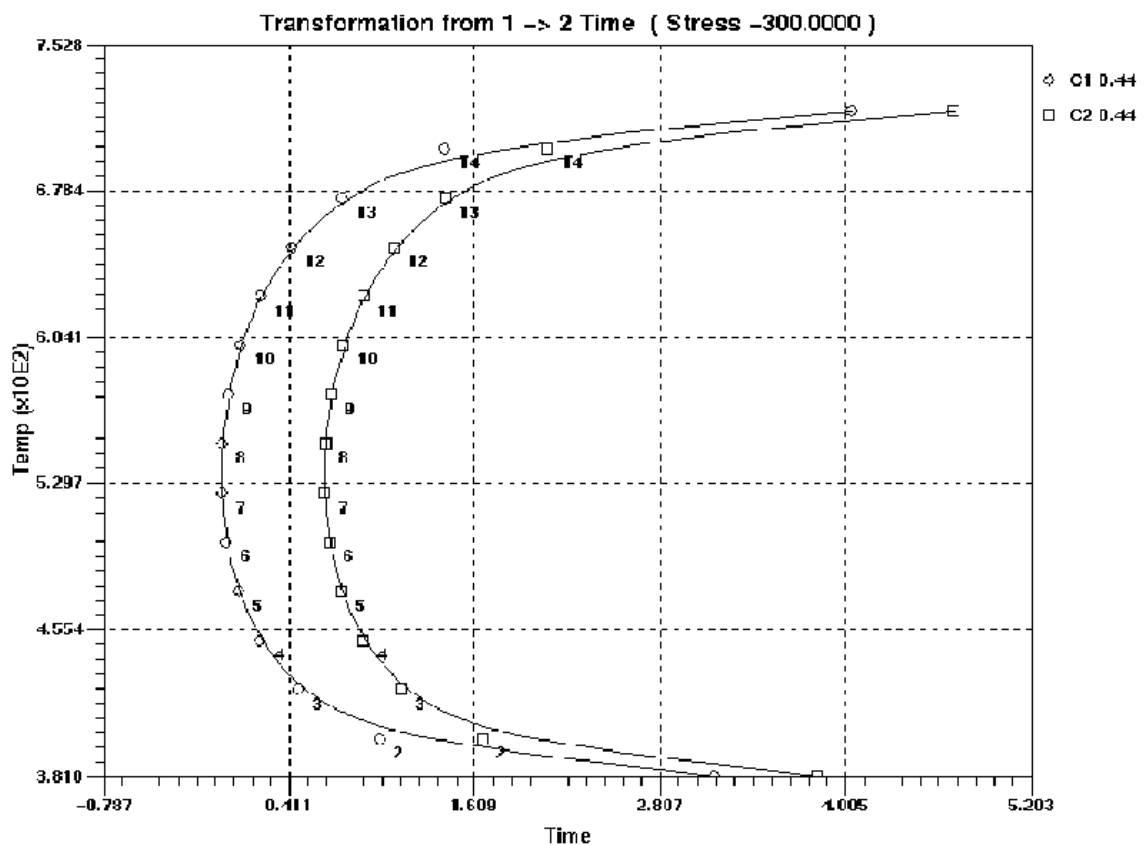


Figure 7.2 shows an example TTT diagram in DEFORM. In the case below, two curves are used to define the percent transformed for a given dominant atom content. Each curve has a value at which an amount of transformation is defined. The curve to the left has been defined in which starting pct of the transformation from austenite to bainite. The curve to the right has been defined in which ending pct of the austenite to bainite transformation.

Martensitic type (Tms, Tm 50 Table form)

The transformation start and 50 % level temperature are inputted as a table format by depending on carbon content and stress levels. This model is not currently available for the current release of DEFORM.

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 power n depends on the kinds of 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 $f_c(C)$ describe 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 currently available for the current release of DEFORM.

Martensitic type (function)

The volume fraction of diffusionless-type (martensite) transformation depended on temperature, stress and carbon content is introduced by modifying the 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 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)

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;

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

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 currently available for the current release of DEFORM.

User routine

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

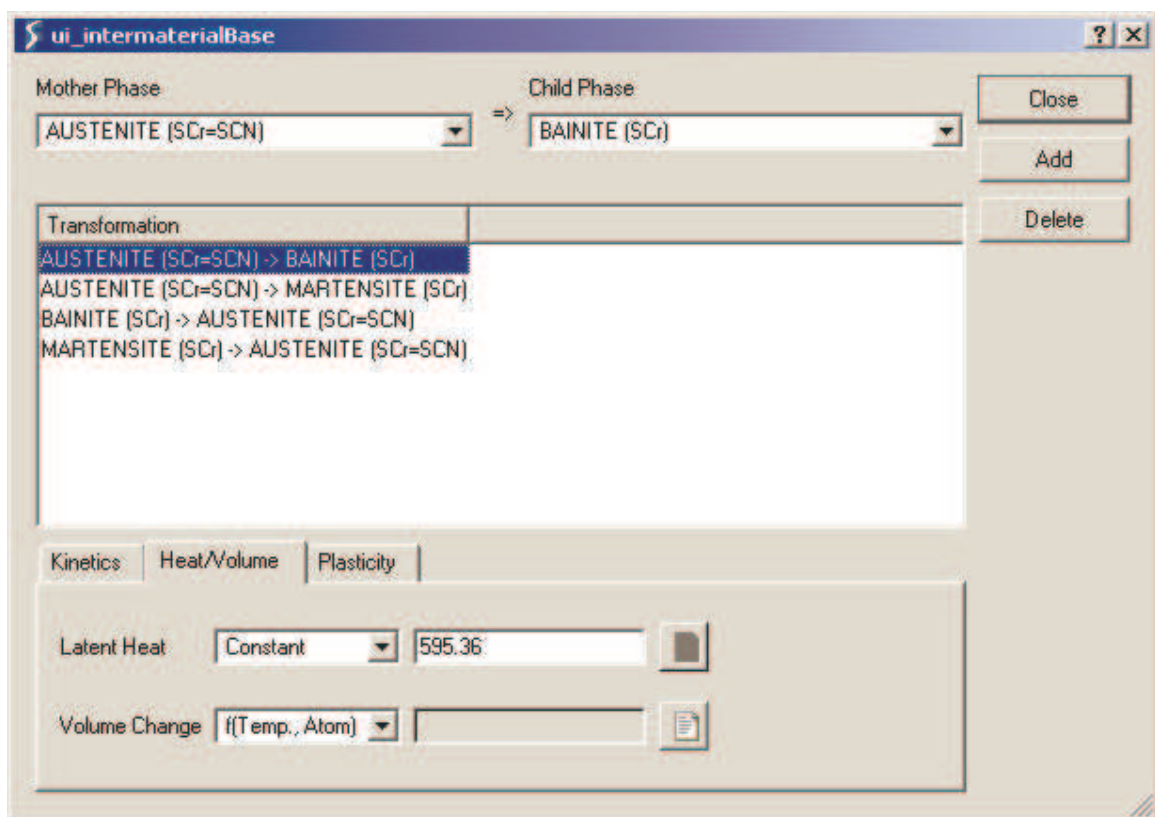


Figure 33: Transformation induced volume and latent heat window.

2.3.3. Latent heat (PHASLH) [MIC]

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.

2.3.4. Transformation induced volume change (PHASVL) [MIC]

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:

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

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 a increase in volume change and a negative sign means there is a decrease in volume over the transformation.

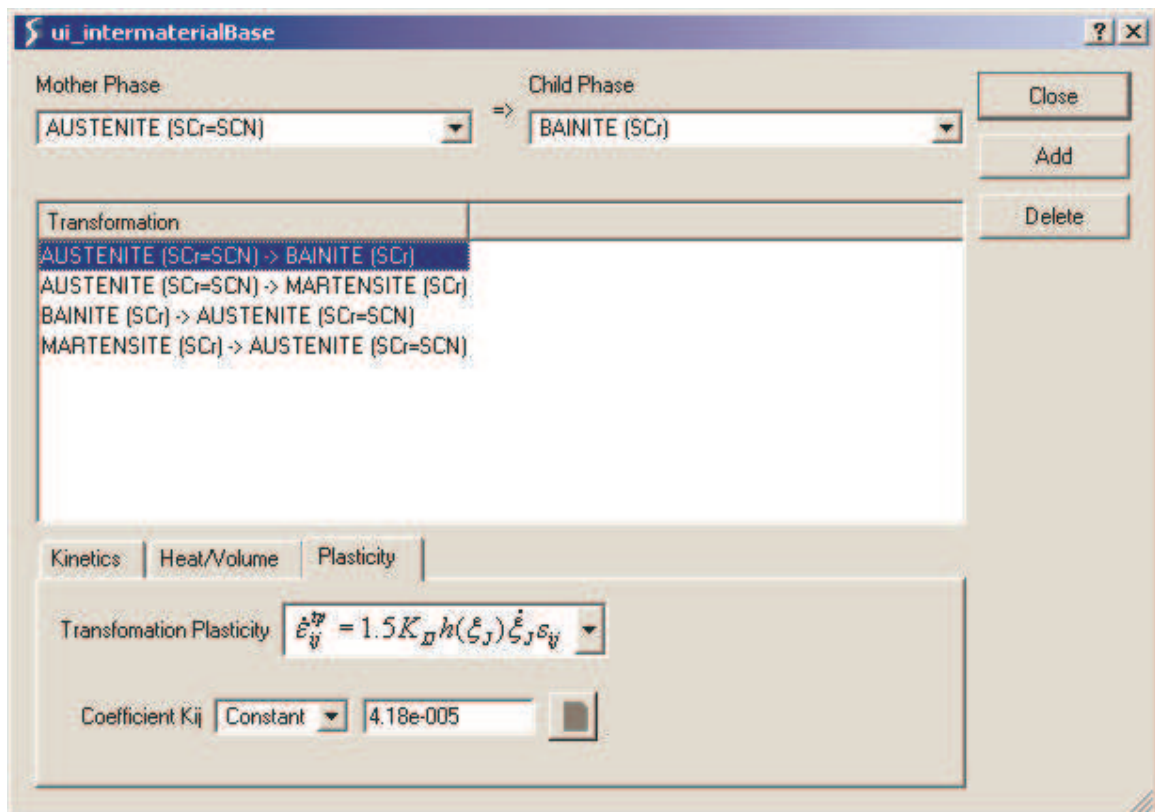


Figure 34: Transformation plasticity window.

2.3.5. Transformation plasticity (TRNSFP) [MIC]

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 :

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

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)

2.4. Object Definition

The objects window contains all the object specific data such as the geometry, the mesh, boundary conditions, movement, and initial conditions for all the objects in a simulation.

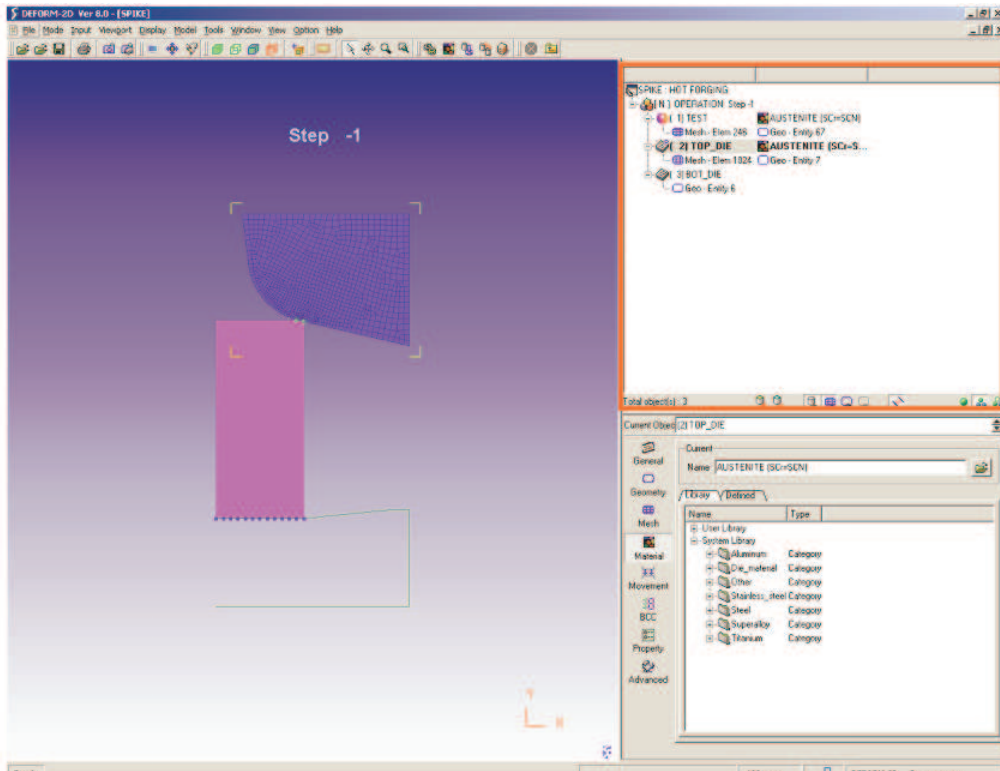


Figure 35: Preprocessor with the object list with a red box.

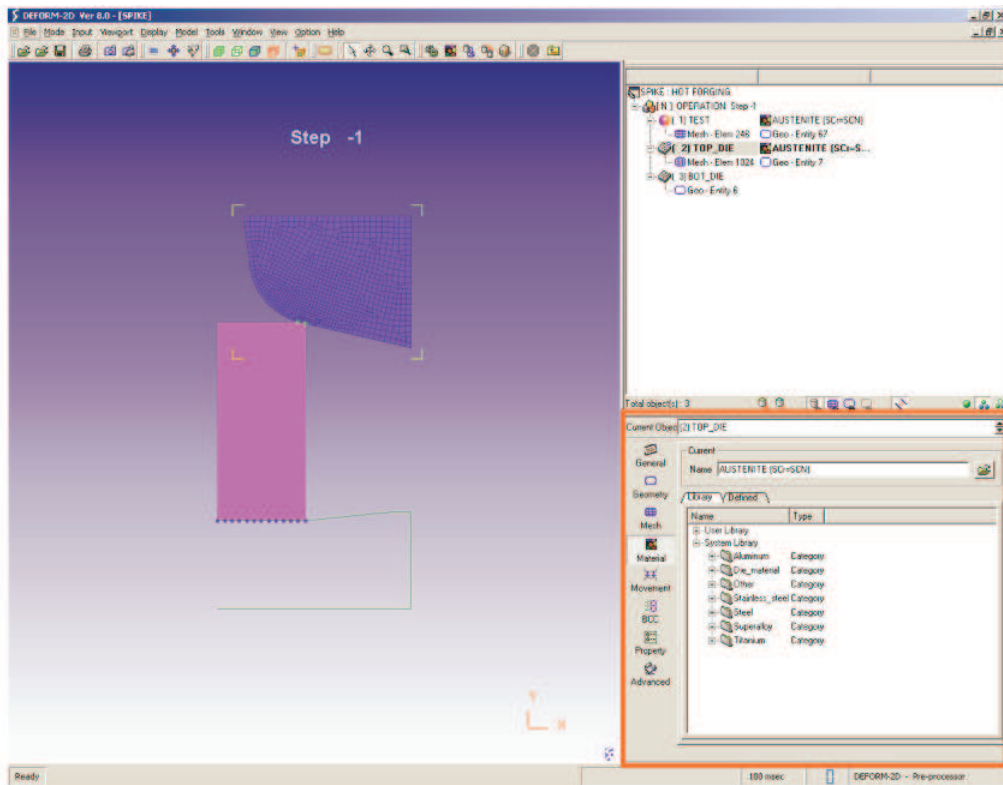


Figure 36: Preprocessor with the object properties window with a red box.

2.4.1. Adding, deleting objects



Figure 37: Insert and Delete object buttons with a red rectangle.

To add an object to the list of objects, click on the *Add 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. This will delete all entries associated with the object, 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 delete only the object geometry from the geometry window. This is useful for changing die geometries when performing two or more deformation operations on the same workpiece. 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 to 0.

2.4.2. Object name (OBJNAM)

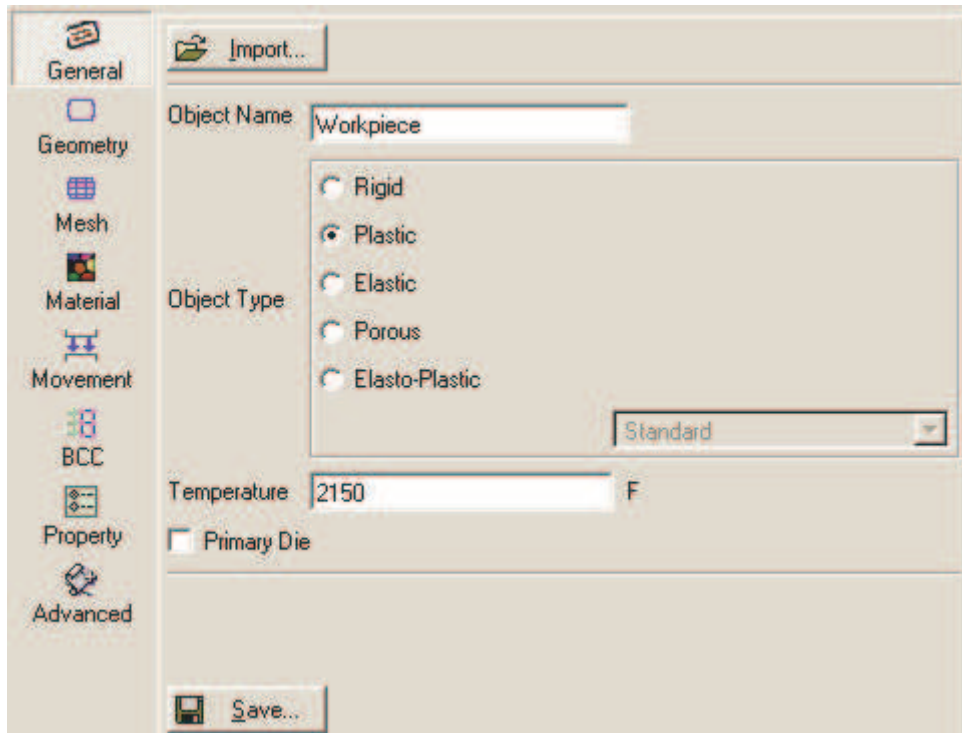


Figure 38: Object general properties window.

The workpiece and each piece of tooling must be identified as a unique object and assigned a hard-coded object number and name. The object name is a string of up to 64 characters. It is highly recommended that this be set to something meaningful (ie. punch, die, workpiece).

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 stopping criteria (SMAX, VMIN, LMAX)

The primary die is defined as a checkbox as seen in Figure 38. This selection is valid for only one object in a simulation..

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 geometry is represented by a geometric profile (DIEGEO). Deformation solution data available for rigid objects include object stroke, load, and velocity. The geometric profile is used for all deformation analysis and the mesh for the rigid object is used for all thermal, transformation, and diffusion calculations.

applications:

When used for the modeling of tooling, the rigid definition 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 except in cases where tool deflection is significant.

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

An elastic material recovers its original form completely upon removal of the force causing the deformation. The elastic material behavior is specified by a Young's modulus (YOUNG) and a Poisson's ratio (POISON). Elastic tools 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, then a single step die stress simulation be used. *It is important to note that the elastic formulation in DEFORM is meant only for small deformation and no rotation.* Thus, we would not recommend using the elastic formulation for the modeling of roller. Refer to the die stress tutorials in the online help for more information.

applications:

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

limitations:

When used to model elastic dies during deformation, simulation time is increased, often for very limited gains in information. 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.

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 workpiece, provides very good simulation of real material behavior. Accurately captures strain rate sensitivity.

limitations:

Does not model elastic recovery (springback), and is therefore inappropriate for bending or other operations where springback 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 in chapter 6. 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 (springback), and strains due to the thermal expansion. Useful for problems such as bending where springback 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.

applications

Appropriate for compacted, sintered powders no less than 70 % fully dense. Accurately models consolidation and densification during forging.

limitations

Cannot model loose powder or compaction processes.

2.4.5. Object geometry

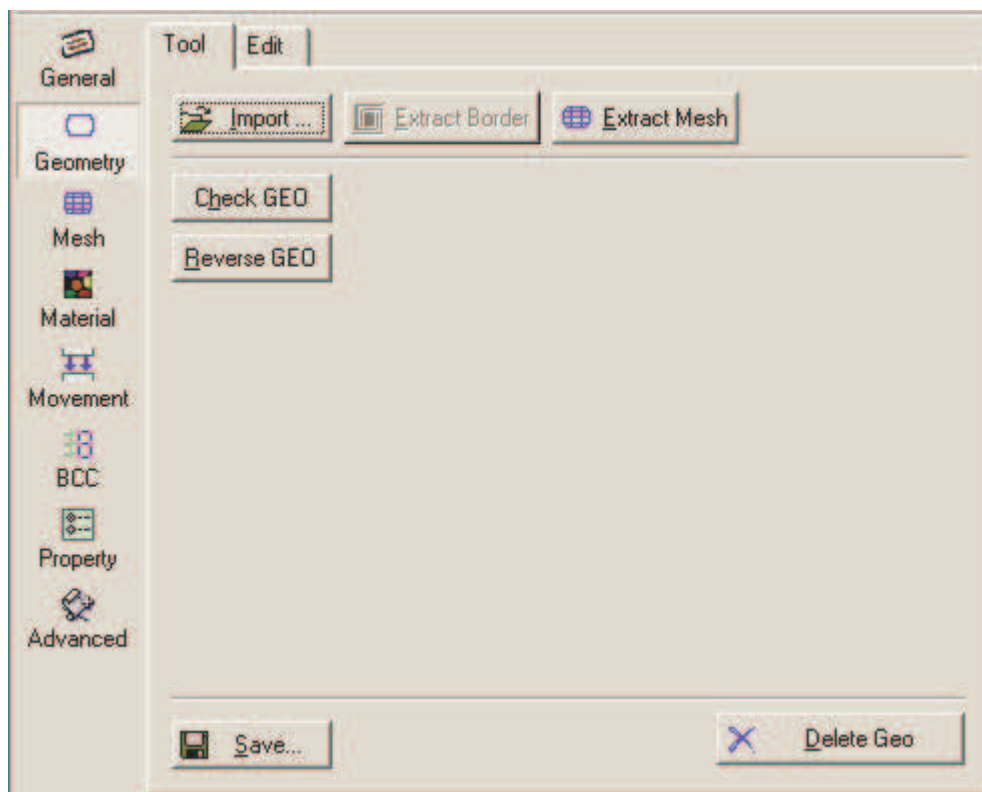


Figure 39: Geometry tool window.

In a DEFORM model, geometry definition plays several roles. Initial geometry may be read from a file, or created in the geometry editor. For deformable

objects the geometry of that is created in the pre-processor is not stored in the database as the shape of the object can change during the simulation. So once a database has been generated, the initial geometry of the object becomes the border of the finite element mesh.

The *Geometry Definition* window is used to create, import, modify, or view the geometry of a given object. The window appears upon selecting the *Geometry* icon from the *Objects* window.

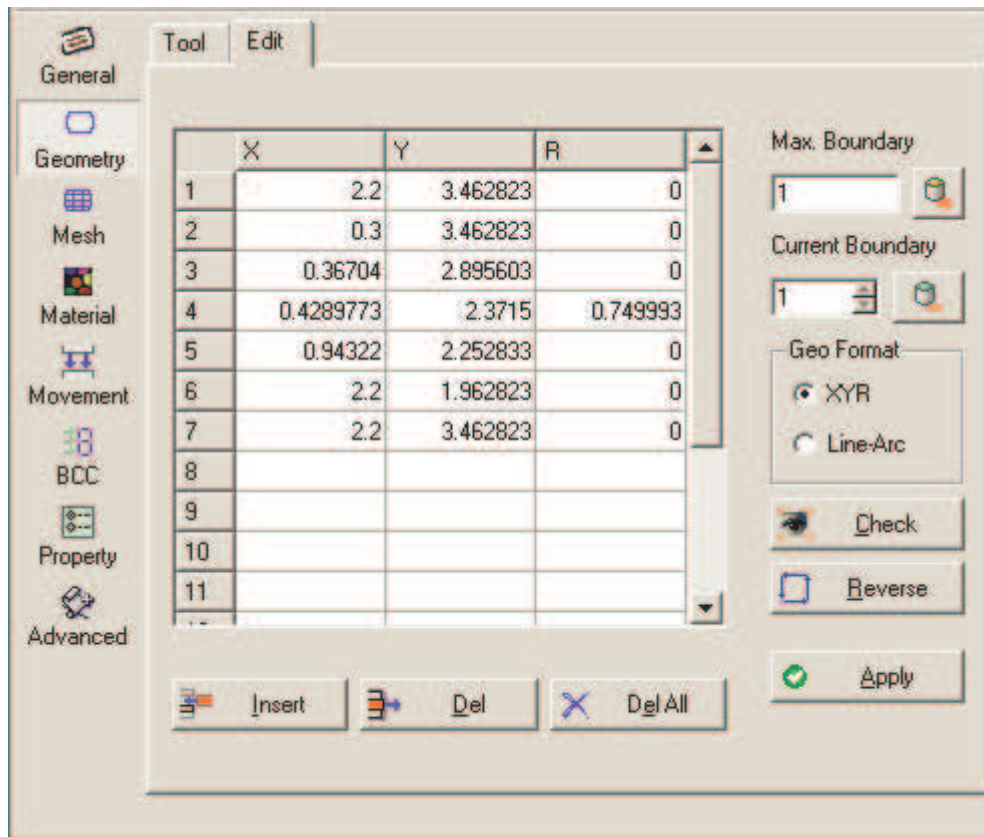


Figure 40: Geometry tool window (editing).

Geometry formats

XYR format input (DIEGEO)

The XYR format consists of defining an X coordinate, a Y coordinate, and a radius for every point of the geometry defining an object. An arc with the specified radius is drawn connecting the lines that would have intersected at the point defined by the X and Y coordinate.

The XYR Table appears directly in the Geometry window. This table allows specifying and/or editing an object's geometry through a number of points in the XYR format. X and Y are the x- and y-coordinates of the point and R is the radius of the point (if it is to define a curved line).

Line-Arc format input (DIEGEO)

Line-Arc format is similar to XYR format in that it can define arcs, but it is more entity oriented. The XYR format defines the connecting points and the connection type, but the Line-Arc format defines the lines and arcs that make up the object, not the connections. The primary reason that the Line-Arc format is used is because IGES files are formatted in the Line-Arc scheme.

AMGGEO format input

The AMGGEO format is the mesh generators internal format for handling geometries. This format can specify geometry as a set of connect points, the XYR format, the line-arc format, and as a set of boundary nodes.

Geometry rules

There are several conventions that must be followed when creating or editing object boundaries.

Orientation

The endpoints of line segments are defined numerically. For all die definitions, all objects should be numbered such that the die block is on the left, and open air (contact zone) on the right as you move along the points in the order. Failure to follow this orientation may cause any of the following problems:

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

Direction

Objects must be defined counter clockwise (CCW). The orientation direction may be checked using the point numbering button in the geometry display window.

If the current connectivity is clockwise (either after drawing or after importing), it can be reversed. The geometry reverse function reverses the current connectivity of the object by clicking the *Reverse* button.

Edges

The die starting and stopping point must be away from the contact zone, unless points are on the axis of symmetry. Tooling which crosses the center-line should do so at 90 degrees: other angles may lead to non-convergence. In case of a pointed punch, a very short line segment should be added at the center.

Blended fillets

The connectivity between blended fillets is ill-defined in IGES format. To avoid problems, it is helpful to define a very short line segments between arcs. Blended fillets can cause DEFORM-2D to continuously give an incorrect geometry message, even once the geometry has been checked and corrected. To correct this, simply add an extremely small line segment between the two fillets.

End points

The radius of the first and last points must be zero.

Clearance fits

Tooling with close clearance fits should be drawn to overlap slightly. Failure to overlap tooling may allow nodes to slip between the punch and the die, and cause problems when the mesh regenerates. A "negative jacobian" error in the message file will result.

Geometry checking

Always check geometry. DEFORM has a checking algorithm that checks for geometric errors, but every type of error can not be caught. Below are a few common geometric errors and how they are corrected by DEFORM.

Note: Clockwise/Counterclockwise orientation is NOT checked in geometry checking.

GEOMETRY ERROR	DEFORM CORRECTION
Duplicate points	Remove the duplicate point
Adjacent points are collinear	Remove the collinear point(s)
Corner radius is so large that at least one of the tangent points lies outside the tangent line defined by the adjacent points	Reduce the corner radius so that both points lie within the tangent line segments
Arc interference at adjacent points: opposite arc orientations	Adjust the radii of the points so that the common tangent point lies on the line connecting the points
Arc interference at adjacent points: same arc orientation	Adjust the radii so that the common tangent point becomes the intersection point of the two

	arcs projected on the common tangent line
Line entity with zero length	Remove entity
Arc entity with zero radius	Remove entity
Adjacent entities cross	Modify entities so that the intersection point
	becomes the ending point of one entity and the
	starting point of the other entity
Unconnected adjacent entities	Add a line segment to connect the entities

Importing existing geometries

Geometry may be imported from a neutral file, a DEFORM native graphics file (AMGGEO), or created using the geometry editor. When importing IGES files, use the mouse to select the object you wish to import. Click on any line segment in the object. Any segments connected to this object will also be selected and highlighted.

Note: It is necessary to use the mouse to select the object to be imported even if there is only a single object in the drawing file.

Creating geometries

Objects can be created in tabulated form. When creating objects, keep in mind that it is very difficult to define complex geometric shapes by adding and moving points. For complex objects, it is suggested that you enter the geometry in tabular form. It is also possible to enter the geometry graphically and then modify it from the table option or vice-versa.

Multiple boundaries

It is possible to create an object with multiple internal holes in the 2D pre-processor. Geometry can be entered for each surface in the X,Y,R format or by importing IGES format files. The point numbering for the hole geometry must be clockwise, opposite the normal DEFORM numbering system. The *Check Geometry* feature will ensure that objects are oriented properly relative to each other. However, the outside object must still be oriented properly.

When importing an IGES file, multiple boundaries may be selected for a single object by pressing the Shift key on the keyboard and selecting the boundaries with a mouse.

2.4.6. Object meshing

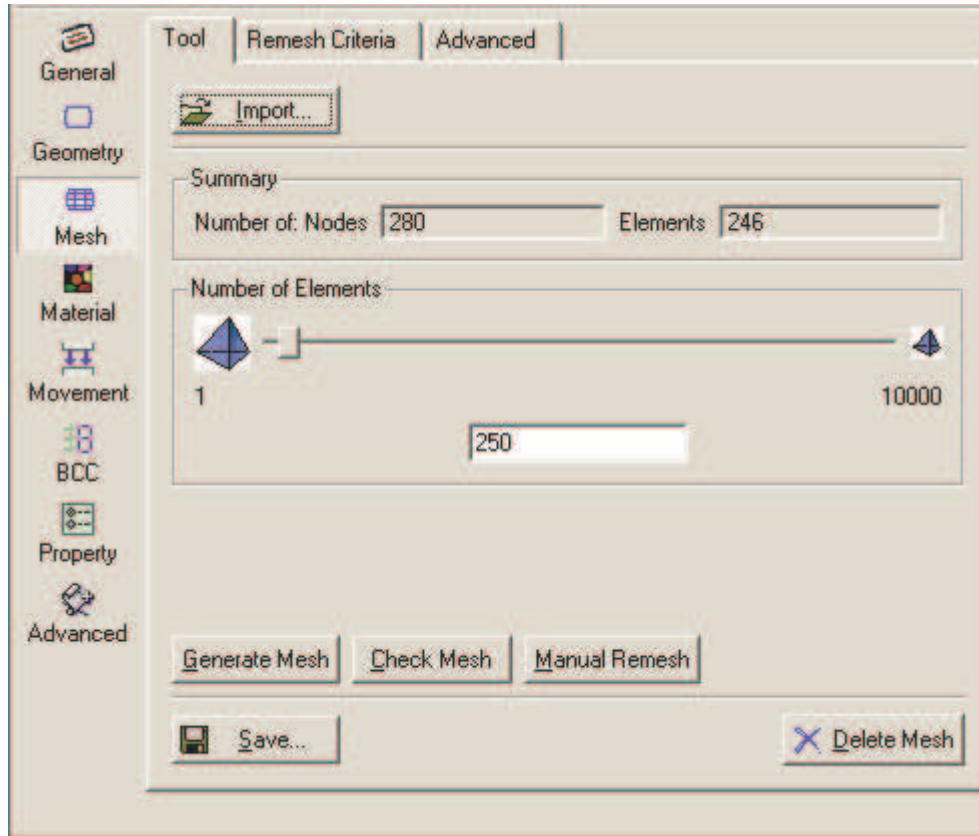


Figure 41: Object meshing window.

The *Mesh Generation* window allows the user to generate a mesh for the current object. The user has two methods of controlling mesh density :

- **System Defined** method uses a system of weights and assigned windows to control the size of elements during the initial mesh generation and subsequent automatic remeshings.
- **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 mesh generation only (this density specification is not referenced during automatic remeshing).

Mesh density refers to the size of elements that will be generated within an object boundary. The mesh density is primarily based on the specified total number of elements and "Point" or "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.

Mesh density is defined by the number of nodes per unit length, generally along the edge of the object. The mesh density values specify a mesh density ratio between two regions in the object. They do not refer to an absolute number of nodes. So, at least two different density regions must be specified.

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. 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 and remeshing problems
- Too coarse a mesh in regions with localized surface effects (ie high damage along a surface) may cause peak values to decline due to interpolation errors during remeshing.

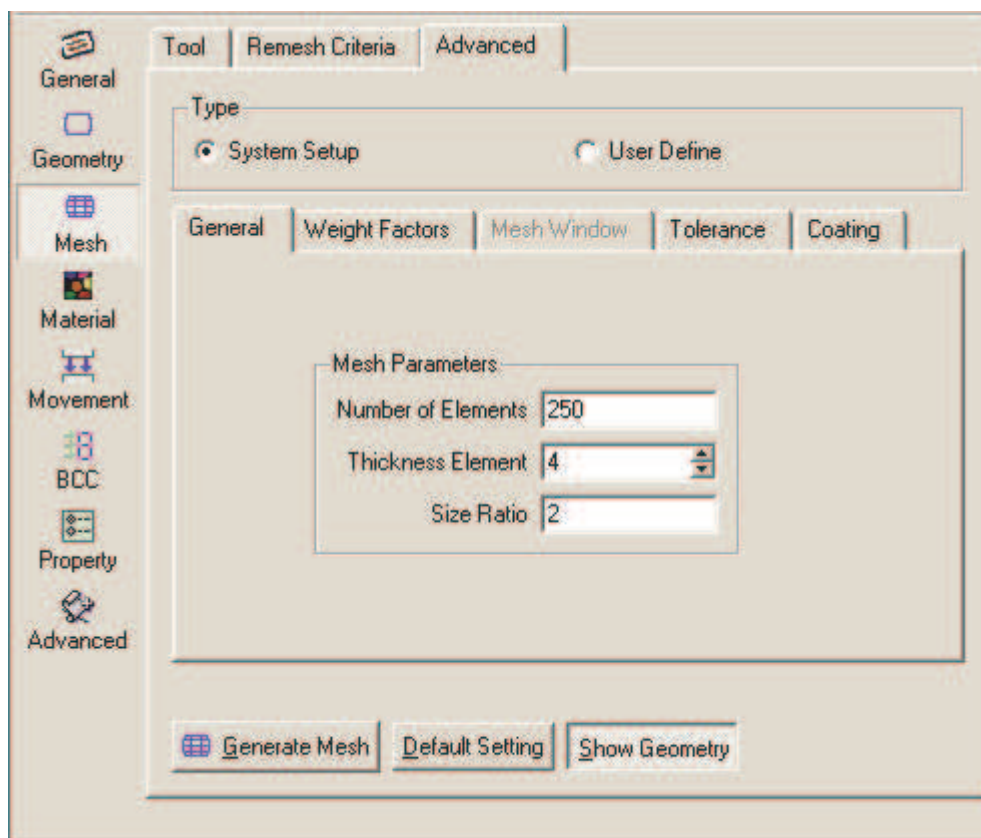


Figure 42: Advanced meshing options (general settings).

Basic mesh controls

Number of elements (MGNELM)

The number of mesh elements represents the approximate number of elements that will be generated by the system. The Automatic Mesh Generator (AMG) takes the value for MGNELM and generates a mesh that will contain approximately the same number of elements. The error between the number of specified elements and the number of generated elements is typically on the order of ten percent. When the mesh is generated, the specified total number of elements is used in conjunction with the "Point" and "Parameter" controls to determine the mesh density.

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 node densities. 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. If SizeRatio = 0, the element size ratio will not be a factor in the mesh density distribution.

Number of thickness elements (MGTELM)

The max thickness ratio is one of several ways to control the mesh density during automatic mesh generation (AMG). The number of elements in thickness direction represents the approximate number of elements that will be generated by the system across the thickness direction of any region of the part. The Automatic Mesh Generator (AMG) takes the value for MGTELM and generates a mesh that will have that number of elements across the thinnest portion. For instance, if MGTELM is set to 4, the AMG will try to have 4 elements across the thickness of the geometry.

The thickness direction of an object is perpendicular to a branched centerline axis for each region of the part. The total number of elements to be generated in a mesh is controlled by the value of number of elements in keyword MGNELM. If the value of thickness elements results in a mesh that contains more than the value specified in MGNELM elements, the value of MGTELM will be scaled down so that the mesh contains approximately MGNELM elements. If the value of MGTELM results in a mesh that contains fewer than MGNELM elements, the remaining elements will be distributed to other user specified mesh density controls (curvature, strain, strain rate, and temperature).

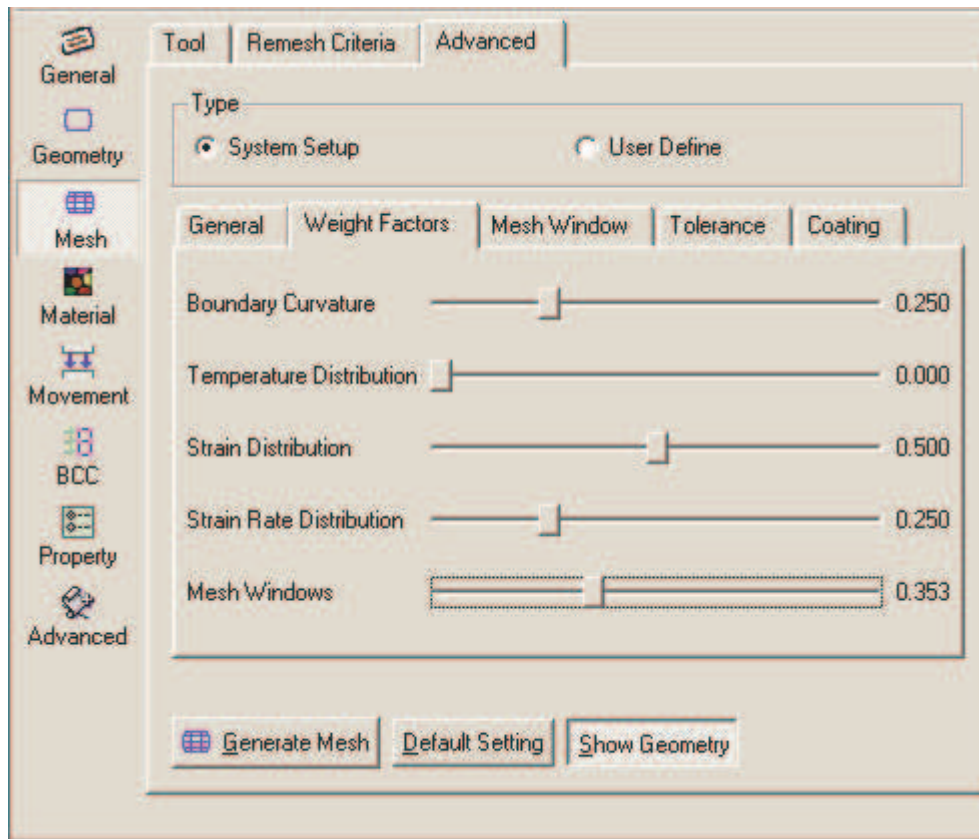


Figure 43: Advanced mesh settings (weighting factors).

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 than a region with a uniform high temperature.

Boundary curvature based weighting factor (MGWCUV)

The Boundary Curvature weighting will apply a higher mesh density to curves on the objects boundary. 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.

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.

Strain base weighting factor (MGWSTN)

To keep a fine mesh in areas of high strain this weight can be adjusted.

Strain rate based weighting factor (MGWSTR)

If there are areas on the deforming object which see high strain rates and localized deformation then using this weight will put a fine mesh in areas where there is a high strain rate gradient

Temperature based weighting factor (MGWTMP)

This weight can be used to specify fine elements in areas of high temperature gradients.

User defined windows weighting factor (MGWUSR)

The user defined windows weight is used in conjunction with *Mesh Density Windows*. User-defined distribution weighting will apply a higher mesh density to areas with a specified density window. If this parameter is set to 0, the mesh windows are ignored during automatic remeshing.

Note : When an object is deformed, the user-defined mesh window will move according to the velocity component assigned to that window. This window will carry the mesh density weight with it throughout the simulation.

User defined mesh density

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 button 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 point with a weighting of 4 will have a mesh that is two times as dense as a point 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.

Boundary density

The boundary density specification is used to place a density value on the boundary of an object. Enter the desired density in the Density box in the Display window. Now select the Add Points button and click on the object's boundary. A green point should now appear with the density value shown next to it. To delete a value, select the Delete Points button and click on the point you wish to delete.

Internal density

The internal density specification is used to set a density value for the mesh inside the object's boundary. Just click on the interior density specification button and enter the desired density value. Now click on the Add Points button and click within the object to place the density values. A yellow point should appear with the density value next to it. To delete the interior densities click on the Delete Points button and select the points you wish to delete.

Combined density

The Combined Density Specification allows a density definition of both surface and interior density values.

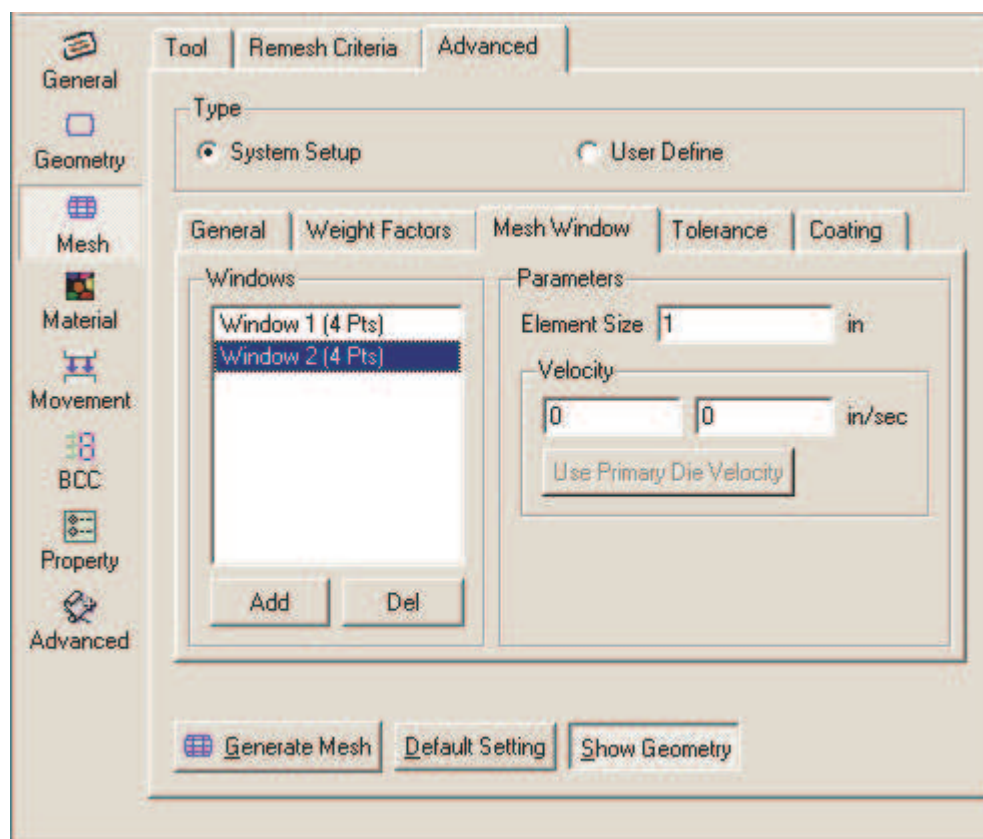


Figure 44: Advanced mesh settings (mesh windows).

Mesh density windows

The Mesh density window concept is similar to a user defined mesh density. Where the window crosses the workpiece border, a specified density value is assigned. However, the mesh density window is used for remeshing, user defined mesh densities are not. It can also be assigned a velocity and it can be defined in an area where the workpiece has not yet been deformed through. The mesh density window specifies an area in space which will move with and object

during deformation. This area has a mesh density definition applied to it and will cause the area to be meshed with an appropriate mesh density.

There are two important facts that should be noted :

1. There must be at least two mesh density windows. The reason is that the value (density level) is compared to another window's density value.
2. The mesh density window must intersect the object's boundary.

Defining mesh density windows

1. Select a window number under the mesh density window header. Up to 10 different windows can be defined.
2. Click Geo button to enable the geometry editor.
3. Add Points, click button to add points. Create a window by defining 3 or more points in the desired area.
4. Once a window is defined, its relative mesh 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. Be sure to set the weight of user-defined areas to a non-zero value under weighting factors.

Mesh density windows have the following data associated with them :

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. The velocity can be in the X direction, the Y direction or a combination of the two.

Mesh density window movement

A constant velocity can be assigned to the mesh density window. This is useful in situations where a higher density needs to be defined around a moving object, such as a punch. The window should be given the same velocity and direction as the punch. 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.

Mesh generation

When all of the Mesh parameters have been set, a mesh can be generated by clicking on the Generate 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.

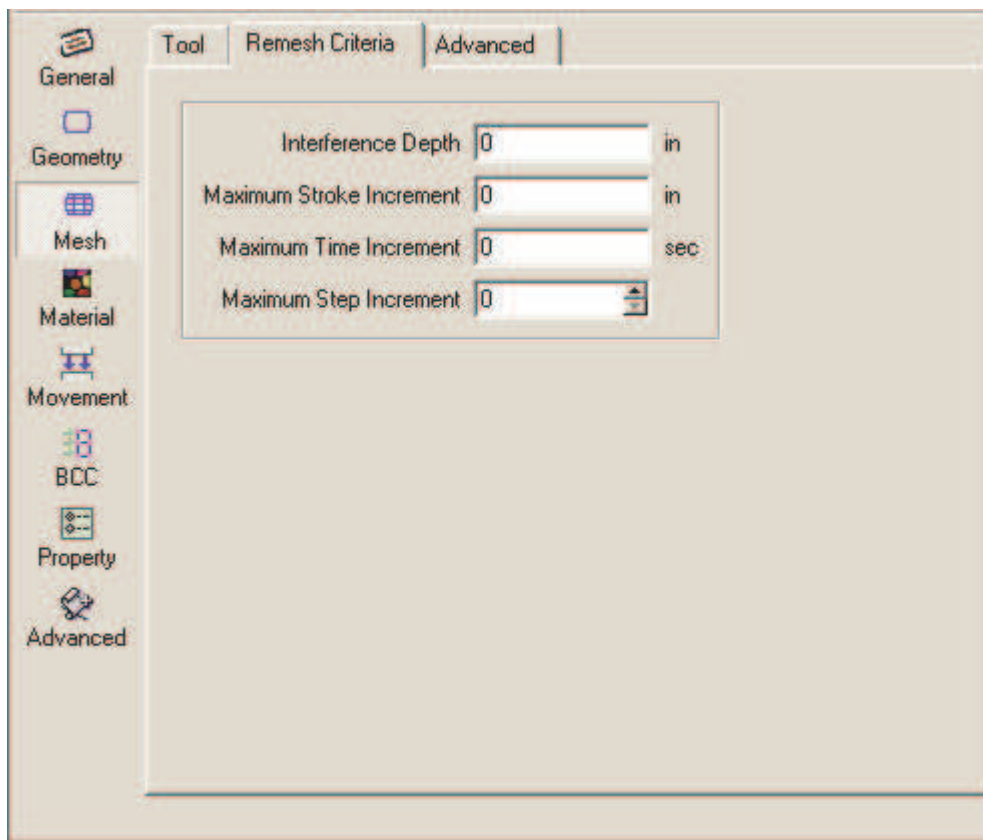


Figure 45: Automatic remeshing criteria window.

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. 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.

Maximum interference depth (RMDPTH)

The maximum Interference Depth is used to start a remeshing procedure. If any portion of a master object penetrates an slave object beyond the depth specified under RMDPTH, remeshing will be triggered.

The interference depth controls the initiation of a remeshing procedure based on the depth of interference between a slave object and a master object. The depth of interference is the depth an element edge of the slave object crosses the surface of a master object. The object to be remeshed must be a slave object.

The interference depth parameter should be used with extremely sharp corners where the corner radius is nearly the same size as the adjacent element edge length. The remesh depth should be set to roughly half of the element edge length. Too large an interference depth may cause excess volume loss. Too small a value may cause too many remeshings, leading to slow run times and excessive interpolation error.

If mesh penetration is occurring, the first course of action should be to use mesh density controls to place a smaller mesh in the offending region. If problems are still occurring, then interference depth can be used.

Maximum stroke increment (RMSTRK)

Anytime the maximum stroke increment is exceeded by the stroke increment of the primary die since the last remeshing step, a new remeshing step will be initiated.

Maximum time increment (RMTIME)

Anytime the Maximum Time Increment (Value of Elapsed Time) has elapsed since the last remeshing step, a new remeshing step will be initiated. The keyword RMTIME controls the initiation of a remeshing procedure based on the process time measured from the last remeshing. It is the value of process time allowed to elapse between remeshings of an object. For a value of 10 for RMTIME, the object will be remeshed at least every 10 seconds. This is helpful when the simulation stops prematurely because of a negative jacobian error. Before the mesh becomes unusable, the object can be remeshed. The elapsed process time between remeshings will not be used to determine when the object is remeshed.

Maximum step increment (RMSTEP)

Anytime the Maximum Step Increment (Number of Steps) has occurred since the last remeshing step, a new remeshing step will be initiated. The keyword RMSTEP controls the initiation of a remeshing procedure based on the number of simulation steps measured from the last remeshing. It is the value of

simulation steps allowed to elapse between remeshings of the object. For a value of 15 for RMSTEP, the object will be remeshed at least every 15 steps. This is helpful when the simulation stops prematurely because of a negative jacobian error. Before the mesh becomes unusable, the object can be remeshed.

Advanced mesh options

These criteria need not be changed except for very rare cases. Among these cases are objects with very fine details, objects requiring very steep gradients in mesh density, etc... Before changing these values, the user may want to consult the staff at SFTC to make sure that changing these values will perform the task that the user requires.

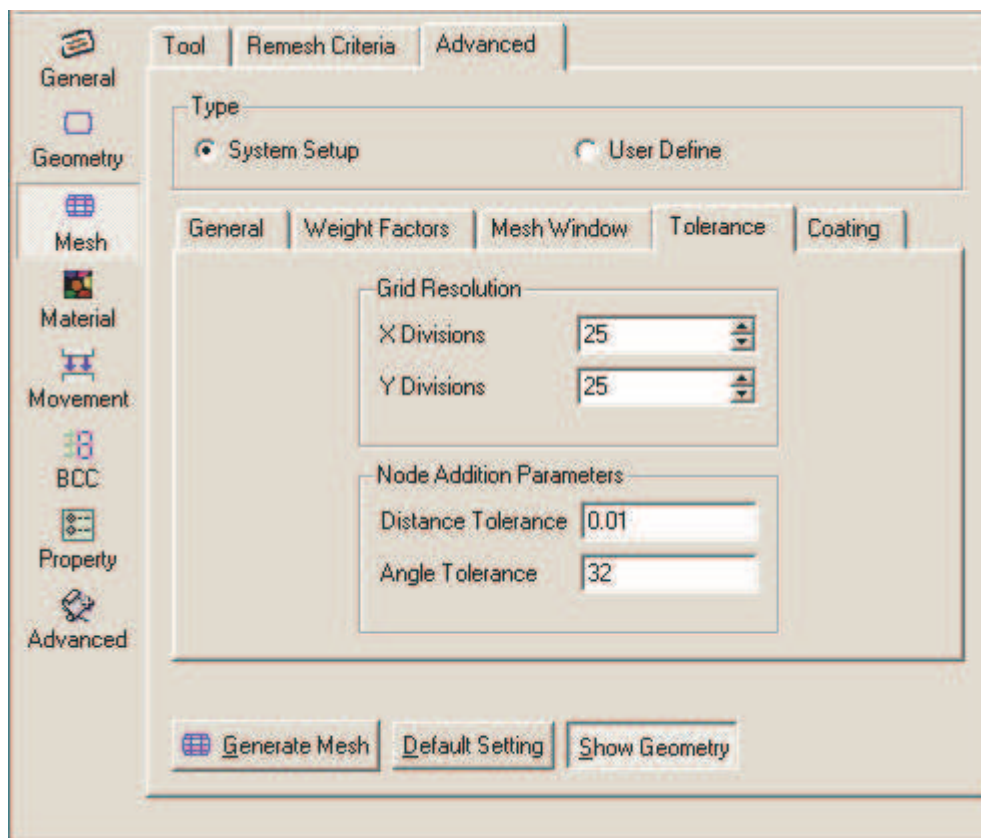


Figure 46: Advanced mesh settings (tolerances).

Grid resolution (MGGRID)

When an object is meshed in 2D, a sampling grid is required to discretize density of the mesh throughout the starting geometry. Grid resolution specifies the spacing of the sampling grids which are used to sample mesh densities. Increasing the value of Xdivision or Ydivision will result in sharper gradients between areas of differing mesh density. In the case of blanking, where a very high mesh gradient is required over a narrow region, these values may need to

increased to capture high changes in mesh gradient over short distances.

Node addition parameters (MGERR)

The node addition parameters specify the maximum distance and angle error permitted between the object boundary and its associated grid element side. The distance and angle tolerances are used to capture critical boundary geometry that might otherwise be lost when the mesh is generated. If an object is required to capture very small features, the maximum distance can be decreased or if a node needs to be placed on a shallow angle, the angle error can be decreased as well. Rarely will the user ever have to change these values.

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 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. Extract the border of the object (may be done automatically).
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 mesh
8. Interpolate data from the old mesh to the new mesh
9. 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
10. Generate a database and start simulation.

If after remeshing the mesh visibly distorts or the dies change regenerate the mesh again, and interpolate data but not boundary conditions. If boundary conditions are not interpolated, it is necessary to recreate all velocity, heat transfer, interobject, 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.

Meshing objects with multiple boundaries

The most difficult feat in meshing an object with multiple boundaries is to correctly define the geometry. The user should carefully refer to the previous section on defining multiple boundaries. It is imperative that these recommendations be followed very carefully to successfully mesh an object with multiple boundaries. Once this geometry is correctly defined, meshing the object is performed in the same manner in which a solid object would be performed. One point to be careful of is that if the object is very thin with respect to the characteristic size, the user should be careful to specify a large number of elements and possibly boost the grid resolution in the advanced meshing parameters (MGGRID).

2.4.6. 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 *Object, Material* dialog.

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 workpiece 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, bainite, austenite, and martensite. The 1040 mixture properties are assigned to the object, and the volume fraction is set to 100% pearlite under element properties.

2.4.7. 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. For heat treat problems with variable carbon content in the workpiece, 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.

2.4.8. Object properties

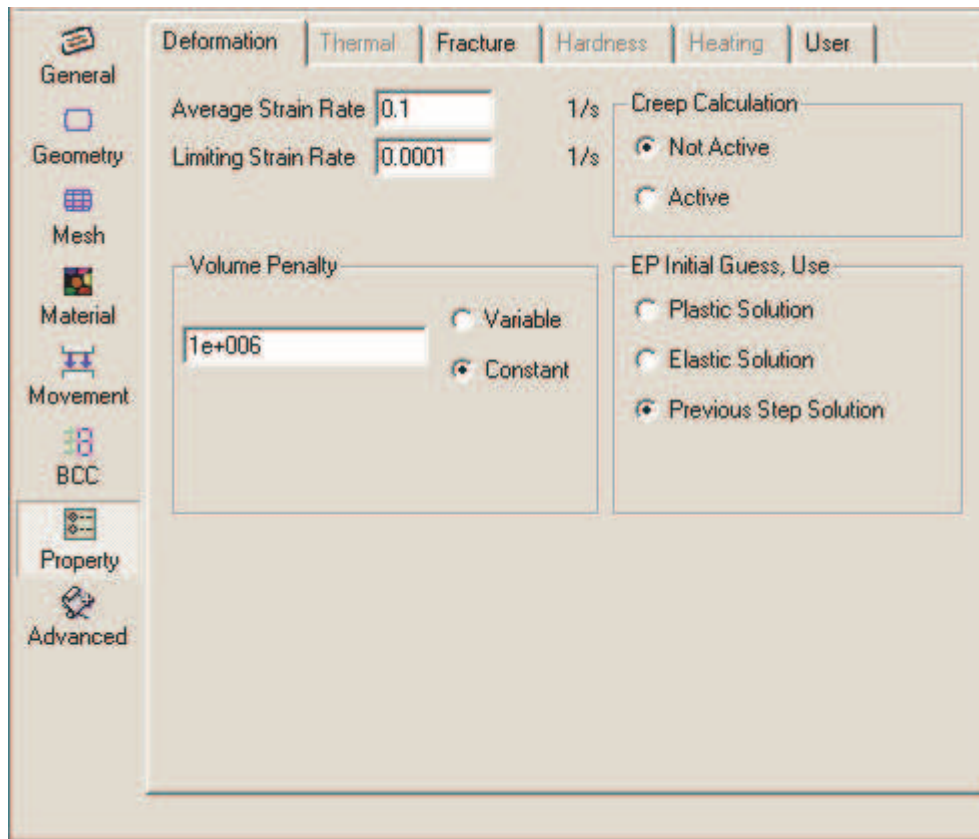


Figure 47: Object properties window.

Miscellaneous object parameters which affect either thermo-mechanical behavior of the object, or numerical solution behavior, are specified in the *Object-Properties* menu or the *Object-Advanced* menu.

Deformation properties

Limiting strain rate (LMTSTR)

The limiting strain rate defines a limiting value of effective strain rate under which a plastic or porous material is considered rigid.

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 workpiece.

DEFORM automatically maintains the ratio between average strain rate and limiting strain rate. Generally, the value of limiting strain rate should be 0.1

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.

Generalized plane strain control (ZSTR)

Generalized plane strain control allows certain object to have thickness direction deformation for plane strain element. The thickness direction deformation can be controlled either by prescribed velocity or traction in thickness direction. This option is available for elasto-plastic material with both velocity and traction control and for rigid plastic material with velocity control only.

The object lies between two bounding planes which may move as rigid bodies with respect each other, thus causing strain of the thickness direction of the object. Let $P_0(X_0, Y_0)$ be a fixed point in the reference planes. The length between P_0 and its image in the other plane P_1 is $t_0 + \Delta u_z$, where t_0 is the initial thickness and Δu_z is change in length in thickness. The thickness strain is define as

$$\Delta_z = \ln (t / t_0)$$

where t_0 and t are the thickness of object at the initial and current configuration respectively. In this definition the plane is only allowed to move parallel to the reference plane, therefore all the points in the object will have same thickness strain.

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.

Elasto-plastic initial guess (ELPSOL)

The convergence of an elastic-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 problem should be tried.

Creep solutions (CREEP) [MIC]

Activates creep calculations for a particular object. For more information on creep calculations, refer to Creep (CREEP).

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

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:
Equation

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.

Fracture properties

Ductile fracture of a deforming workpiece can be modeled in DEFORM. If the fracture function is turned on, material separation will be modeled for any elements which exceed a critical damage value, specified in the Material Properties-Advanced-Fracture menu.

This feature is useful for modeling shearing and blanking, machining, fracture of deformable installation fasteners (pop rivets) and other applications.

Fracture is modeled by deleting any elements which exceed the critical damage value. Therefore, to limit volume loss, an extremely fine mesh should be used in any region where fracture is expected.

The FEM simulation is temporarily stopped to perform element deletion. The stopping may be triggered by a given number of steps, or whenever the damage value in a specified number of elements exceeds the critical value.

Fracture step (FRCSTP)

The step interval at which the simulation should be stopped to perform element deletion. If no elements are above the critical damage value, none will be deleted.

Fracture elements (FRCNEL)

The number of elements which must be above the critical damage value for the simulation to be stopped to perform element deletion. A typical value is around 4.

Fracture method (FRCMTH)

At the writing of this manual, only the element deletion method of fracture has been implemented.

Hardness properties [MIC]

Material hardness predictions can be based on:

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

Hardness data for a material can be entered in the *Material Properties* menu (Hardening model (HDNRUL)). 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.

2.4.9. Object boundary conditions

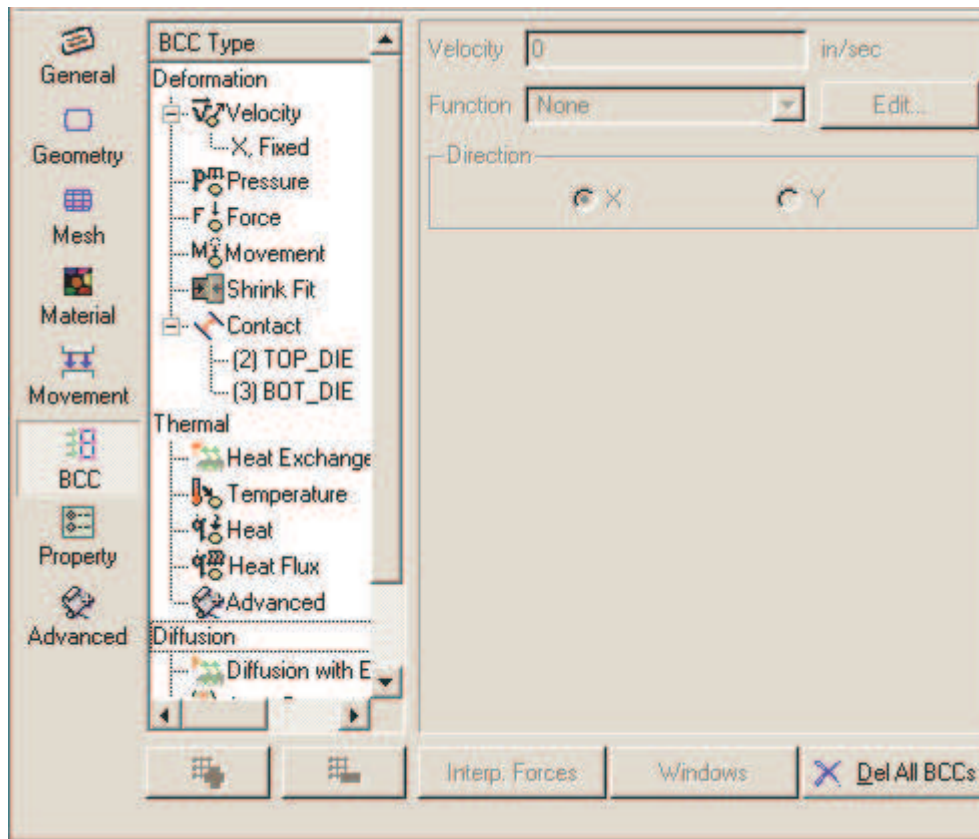


Figure 48: Object boundary condition window.

Boundary conditions specify how the boundary of an object interacts with other objects and with the environment. 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. In version 7.0, some of the boundary condition definitions have been changed from a node based definition to an element edge-based definition.

The purpose for changing from a node-based definition to an edge-based definition is to reduce ambiguity at corners. If heat exchange with environment is defined on an edge and heat flux is set to zero at an adjacent edge there is an ambiguity at the corner. If the corner node is set to heat exchange with the environment, then the definition at the edge with one heat exchange BC and one heat flux BC is not clearly defined. The purpose of the edge definition is to eliminate this problem in any case where the boundary condition acts over the length of an element edge such as pressure, heat flux and atomic diffusion from the environment.

Defining object boundary conditions

Boundary conditions are specified and enforced at nodes or element edges 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.
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.

Plane

Select a node on a plane. All nodes on that plane will be selected.

Surface Patches

Click the surface patches icon. Surfaces will be extracted based on a 30° angle between adjacent element faces. Selecting any node within a given patch will select all nodes on that patch.

Regions

Points defining a closed freeform region are defined. Any nodes within the region will be selected. This feature should be used with caution because it is relatively easy to inadvertently select unintended nodes.

To apply the boundary conditions click the Generate BCC's button. Colored markers will highlight the edges 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. In version 7.0, if the previous method of defining the boundary conditions at the nodes was used, it will still be shown as boundary conditions at the nodes.

Deformation boundary conditions

Velocity BCC

Velocity of each node can be specified independently in the x, y, and z directions. 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 workpiece is pulled through a die. *NOTE: If parallel symmetry planes are to be defined, velocity boundary conditions can only be used on one plane. A rigid surface should be defined on*

the other.

Force BCC

Force boundary conditions specify the force applied each node. The force is specified in default units. For die stress analysis, the force that the die exerted on the workpiece 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.

Caution should be used in specifying force boundary conditions. If the part is remeshed, the total force acting on the object will change as the number of nodes changes. Where possible, it is better to define a rigid surface, attach the nodes to that surface, then apply a force to that surface.

Pressure BCC

The pressure boundary conditions specifies a uniform, or linearly varying, force per unit area on the element faces connecting the specified edges. In version 7.0 and above, only normal pressure is supported. If a tangential component of pressure is required, the user is recommended to use force boundary conditions. Two values for the normal pressure are required, the first value is the beginning value of pressure from the beginning point where pressure is set, the second value is the value at the end of where the pressure is specified. The pressure is linearly interpolated between the start and the end. The keywords that replace the old definition of pressure are ECCDEF and ECPRES.

Displacement and shrink fit BCC

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 BCC

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

Advanced deformation BCC

The Advanced boundary condition displays interobject boundary contact conditions on a given object. This is the same information displayed in the Inter-Object BCC's window. There is no physical significance to the x, y, or z components of contact. Rather, the "directions" are dictated by numerical convenience. Contact conditions are first assigned to the Z direction. If that position is occupied by another value, conditions are assigned in the Y, then X directions. Refer to the BCCDEF section under *Node Properties* for more information

Thermal Boundary Conditions

Heat exchange with the environment BCC

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. In version 7.0, this boundary condition is specified on the element edges rather than on the boundary nodes. This will reduce the ambiguity upon defining symmetry planes. On a corner element, the two edges will be defined consistently as opposed to the case where the nodes were defined and one of the edges was ambiguous. The new keyword for heat exchange is ECCTMP.

Default heat exchange with the environment occurs to the ambient environment as described above. However, heat exchange windows may be specified using the heat exchange windows icon. Heat exchange for nodes within these windows is controlled by the parameters set for each window.

Heat flux BCC

Specifies an energy flux per unit area over the face of the element bounded by the nodes. Units are energy/time/area. This is defined in terms of edge definition starting with version 7.0. The previous method is supported in 7.0 but may not be supported in versions after 7.0. The keywords for heat flux are ECCTMP and ECHFLX.

Nodal heat BCC

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

Temperature BCC

Specifies a fixed temperature at the given nodes.

Advanced Temperature BCC

The purpose of this boundary condition definition is to allow the user to have the flexibility to specify all the various types of heat boundary conditions on the same edge. The user can either specify a user-subroutine number or a local heat transfer definition. If the user wants to specify a user routine, the User Routine Number should be specified. The User Routine number specified will correspond to the subroutine the boundary condition will correspond to. Refer to User Routines for more information on how to use these user-defined boundary conditions. If the routine number is left zero, the user may then define a local defined boundary condition where the environmental temperature, the convection coefficient, the emmisivity and the heat flux needs to be specified the edge. All four of these variables may be defined as either constants or functions. To apply a local user defined boundary condition, set the variables you want, set the local defined number to a unique value and apply this to a set of element edges. The

new keywords for local edge definition are ECCDEF, ECTMFN and LOCTMP.

Diffusion Boundary Conditions [DIF]

Diffusion with the environment BCC

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. The new keyword for this is ECCATM.

Fixed atom content BCC

Specifies a fixed dominant atom content at the given nodes.

Atom flux BCC

Specifies a fixed dominant atom flux rate over the elements bordered by the indicated edges. This is now specified using an edge definition starting in version 7.0. Atom flux may be defined as a constant or as a function. The new keywords for this are ECCATM and ECAFLX.

Advanced Diffusion BCC

The purpose of this boundary condition definition is to allow the user to have the flexibility to specify all the various types of diffusion boundary conditions on the same edge. The user can either specify a user-subroutine number or a local diffusion definition. If the user wants to specify a user routine, the User Routine Number should be specified. The User Routine number specified will correspond to the subroutine the boundary condition will correspond to. Refer to User Routines for more information on how to use these user-defined boundary conditions. If the routine number is left zero, the user may then define a local defined boundary condition where the environmental atom content, the reaction rate coefficient, and the atom flux needs to be specified the edge. All three of these variables may be defined as either constants or functions. To apply a local user defined boundary condition, set the variables you want, set the local defined number to a unique value and apply this to a set of element edges. The keywords for this are ECCATM, ECAFLX and LOCATM.

2.4.10. 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 Interobject data section. If a node is specified to be

in contact with a particular object, it will be 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 Interobject Contact relation (CONTACT) section.

Contact boundary conditions can be displayed for a given object using the *Objects, Boundary Conditions, Advanced Deformation BCC's* icon.

2.4.11. Object movement controls

Movement controls can be applied to rigid objects and boundary nodes of meshed objects. The surface defined by these nodes can be thought of as a "rigid surface". During the simulation, the constrained nodes will move synchronously in the speed and direction defined by the movement controls.

Translational Movement

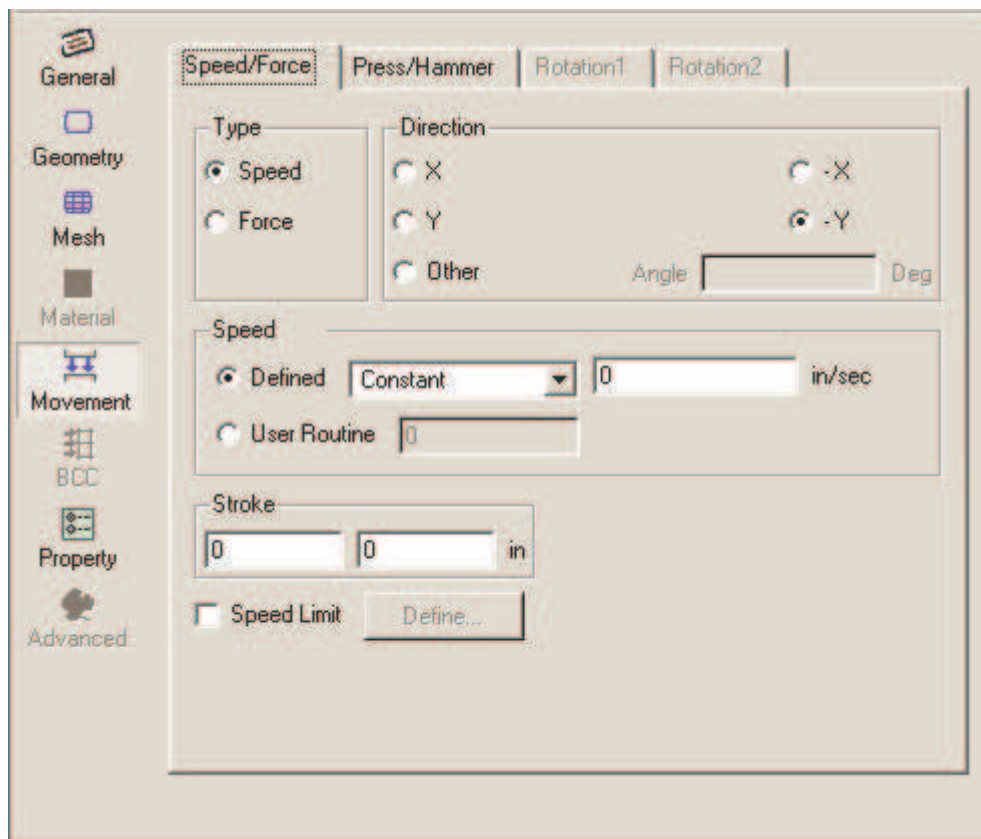


Figure 49: Object movement window (speed/force).

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 all the way around an object boundary. 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, normal to the appropriate 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.

Force control

For force control, the speed of the object is constrained such that the specified load is maintained. When the object is rigid, the load is the resultant load applied by a non-rigid object due to the relative motion of the two objects. When the object is elastic, plastic, or porous, the load is the sum of the nodal loads of all nodes movement boundary condition codes.

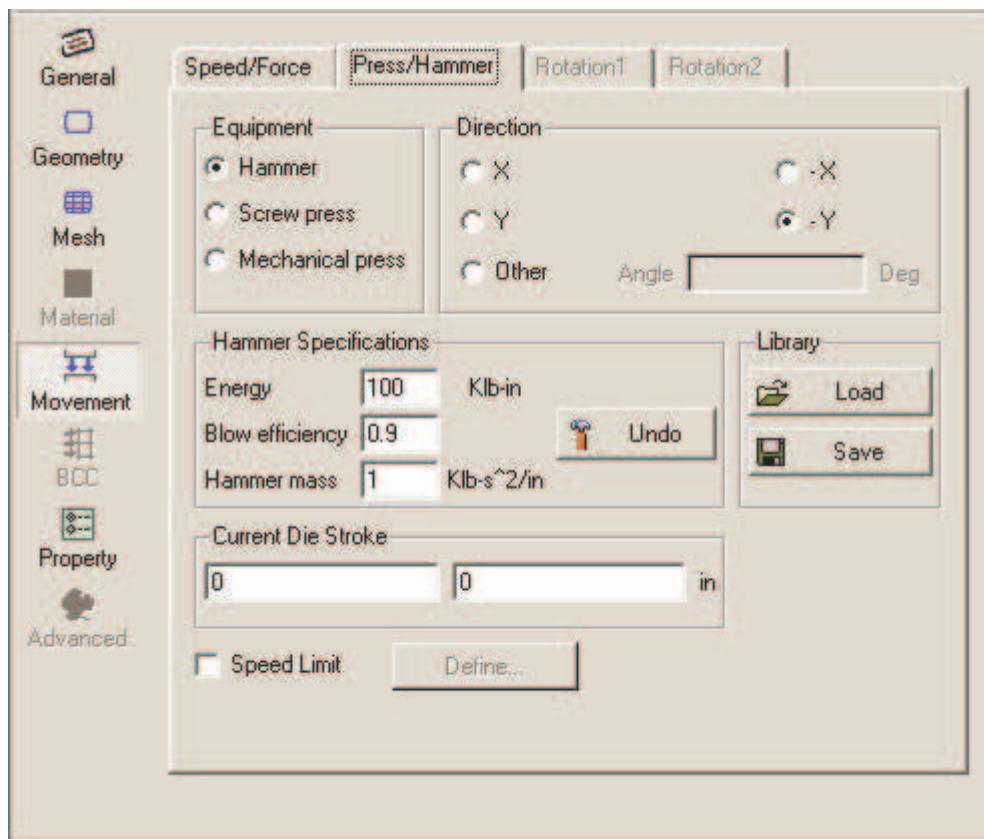


Figure 50: Object movement controls (press/hammer).

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.

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

$$\eta_B = \frac{W_U}{E_T}$$

where W_U is the energy consumed for the plastic energy of workpiece and E_T the total kinetic energy of ram.

There are basically two types of hammer, one is 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 builds up the blow 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, 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:

$$\Delta E_D = L_T \Delta s$$

where L_T is the deformation load at the ram and Δs is the ram travel during Δt . After the increment, the blow energy is adjusted by:

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

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

Screw press energy

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 between 13 and 17 degrees, to a linear movement of the main ram. On contact with the workpiece, the complete kinetic energy of the flywheel and the ram is transformed into useful work (work on the workpiece) and losses (elastic deformation work in the workpiece 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. The resultant load due to inter-object contact is calculated for all objects with a MOVCTL boundary constraint. Screw controlled movement can only be specified for rigid objects. Motion of which is controlled by screw press parameters can only be applied in the X, Y, -X, or -Y 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-m 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 $I = 2E_T/\omega^2$ 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.

Hydraulic press

To use the hydraulic press model the user must input two parameters.

1. **Speed** The speed of the press can be input as a constant or as a function of time or stroke.
2. **Speed Limit** The speed 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.

Mechanical press

The Mechanical Press type replicates the cyclic motion of a mechanical press. The "Mechanical Press" option simulates the motion of objects driven by a mechanical press. The DEFORM parameters required to simulate the motion are the total displacement of the press (Dt_{tot}) relative to the current displacement (D_{cur}), and the number of strokes per unit of time (S'). Using these parameters, DEFORM can compute the die speed at any point. Motion of which is controlled by mechanical press parameters can only be applied in the X, Y, -X, or -Y directions.

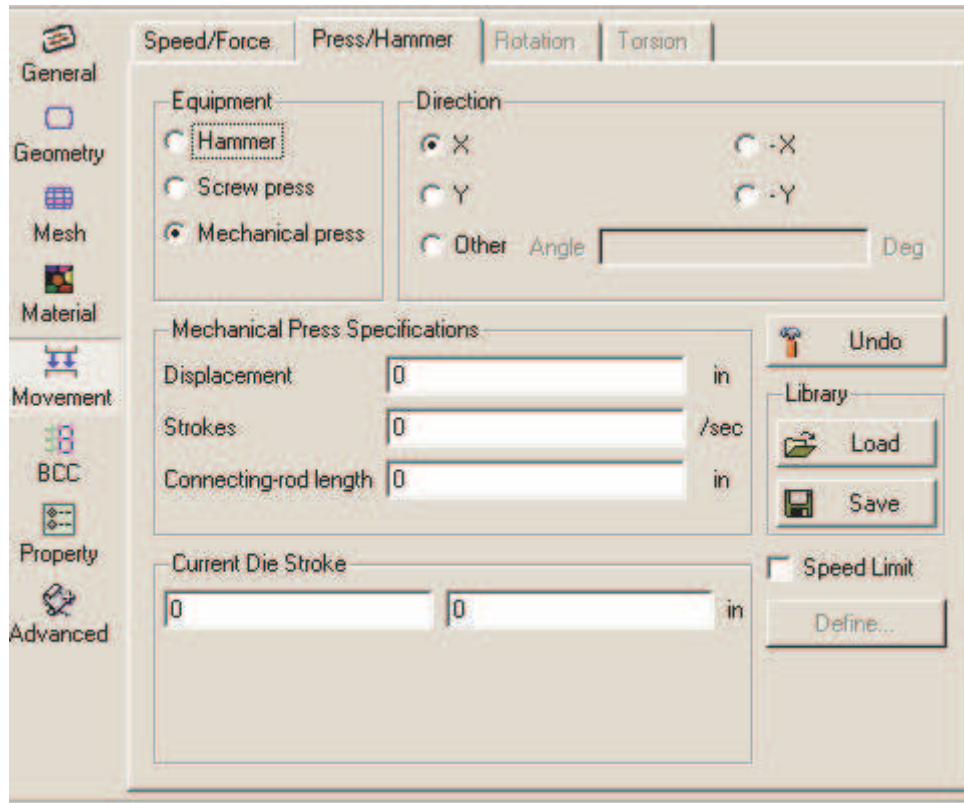


Figure 51: Mechanical press definition.

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 movement of a mechanical press are :

1. **Displacement** The Displacement for the mechanical press represents the total stroke of the press per cycle. In English units this is inch per cycle, in SI units this is mm per cycle.
2. **Strokes per time** The Strokes per time represents the frequency of the press blows. This is a measure of blows per second or cycles per second.
3. **Current Die Stroke** This value is the amount of the stroke length for which the die or punch is in contact with the workpiece. In the case of a die moving in the $-Y$ direction, the current die stroke should have a minus sign and the magnitude should be the distance from the top of the stroke to the current position of the 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 52, 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 be left as zero and it's contribution to the ram speed will not be considered.

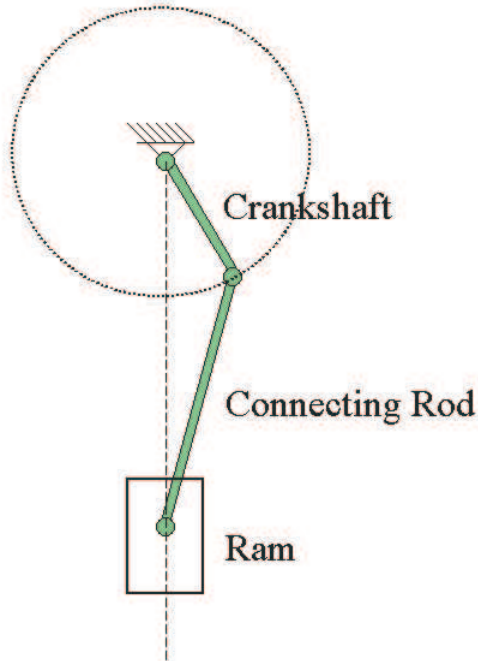


Figure 52: Sketch of a simple direct crank drive.

The Current Die Stroke specifies the displacement of an object from simulation start with a movement control boundary constraint (MOVCTL). It is valid for any object with movement control applied to it. For example, you designate the movement type as mechanical press with a total movement of 8 inches. If the process is to be started mid-stroke, just set the Y stroke component to 4 inches.

The stroke of the primary object (PDIE) can be used to define movement control:

- simulation time step size (DSMAX)
- object movement (MOVCTL)
- simulation termination criteria (SMAX, VMIN, LMAX)

Rotational movement

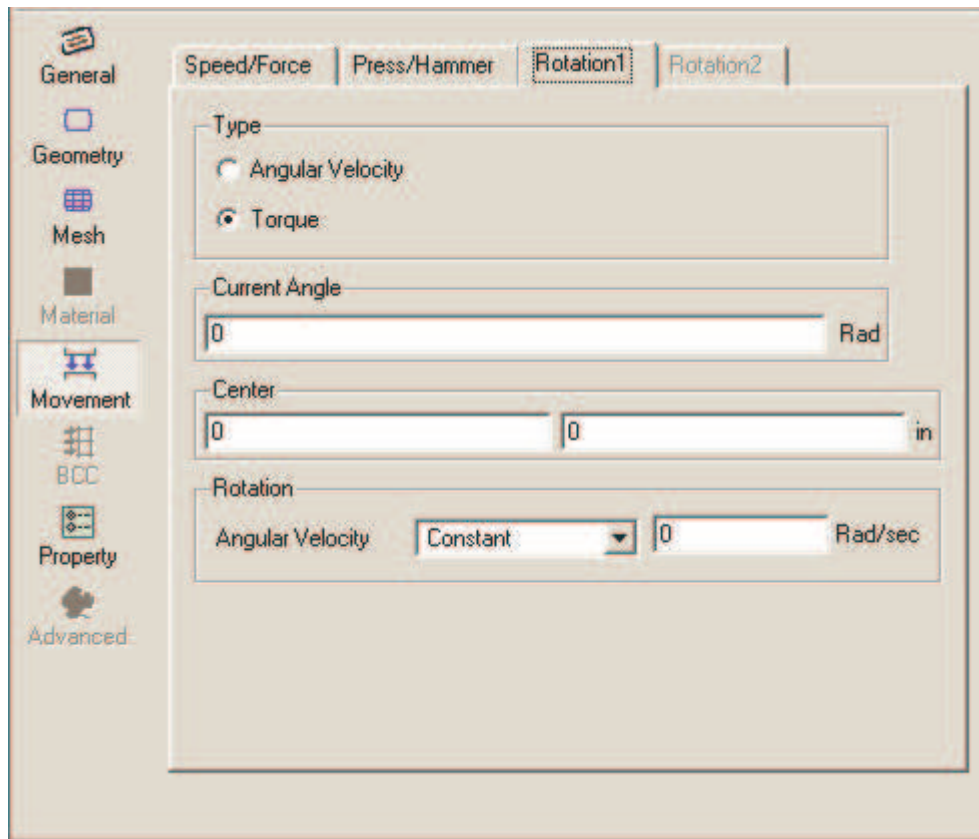


Figure 53: Object movement window (rotational) – plane strain case.

Rotational movement controls are applicable only in the case of plane strain and torsional formulations. Rotational movement is defined by an angular velocity about a fixed center of rotation. 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.

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.

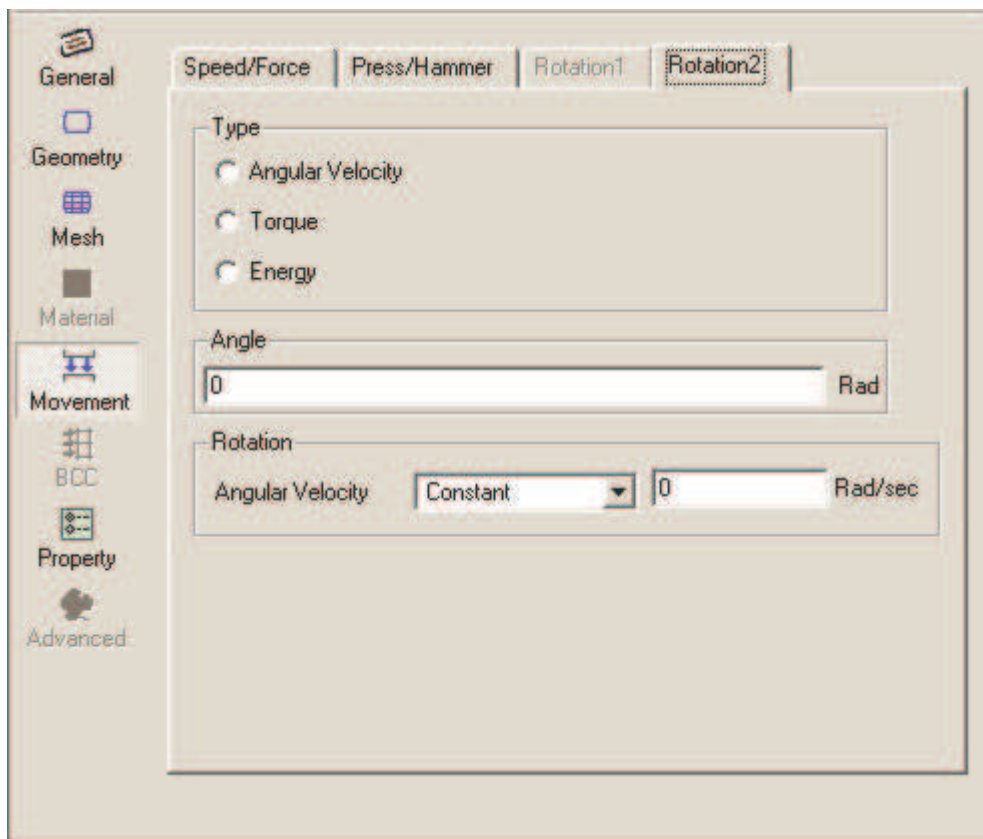


Figure 54: Object movement window (rotational) – torsion element case.

Movement control user subroutines

Complex die movement can be defined using user defined FORTRAN subroutines. Please refer to User Routines for a description of how to implement user defined subroutines.

2.4.12. Object node variables

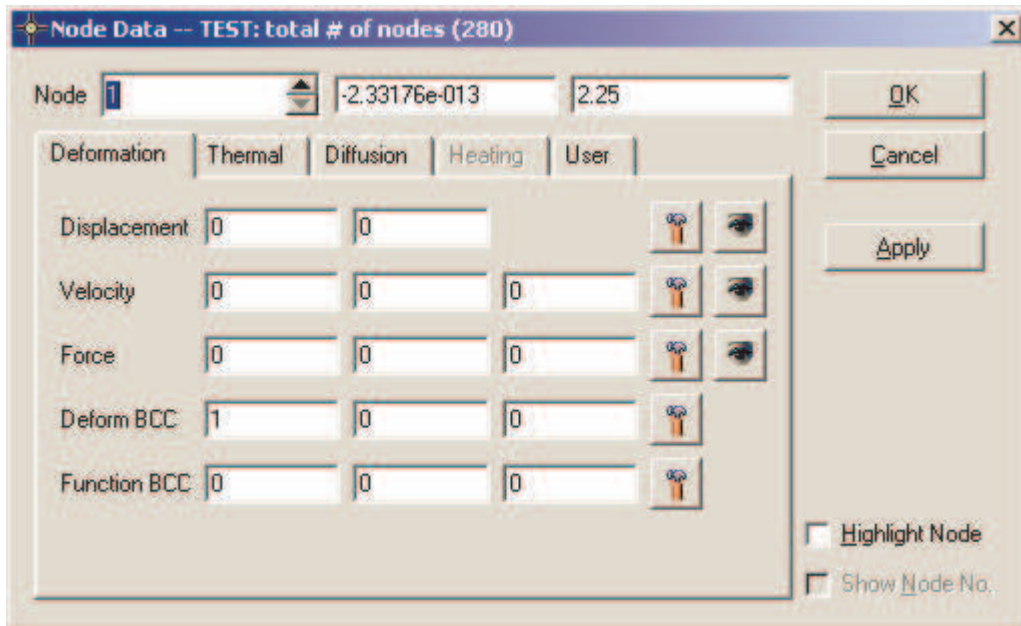


Figure 55: Object node data window.

The nodes window displays all available information about nodes. All information can be modified, and many of the variables can be plotted.

Features of the display window include:

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

i icon 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

Select line or shaded contour display of node variable values

Slicing

Displays sliced object geometry. Refer to the description of the slicing function in the postprocessor for more details.

Save AMGGeo

Saves the object geometry in AMGGeo format.

Volume

Calculates and displays the volume of the mesh.

Position

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.

Deformation 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.

Thermal

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

Diffusion [DIF]

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 workpiece.

Boundary condition code (BCCCRB)

BCCCRB specifies the atom transfer boundary constraint code for individual nodes.

User variables**User node variables (USRNOD)**

User node variable values can be defined using FORTRAN subroutines. Refer to User Routines for more information on the subroutines. Each node value may accept both a name and a value. 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 tracked.

2.4.13. Object element variables

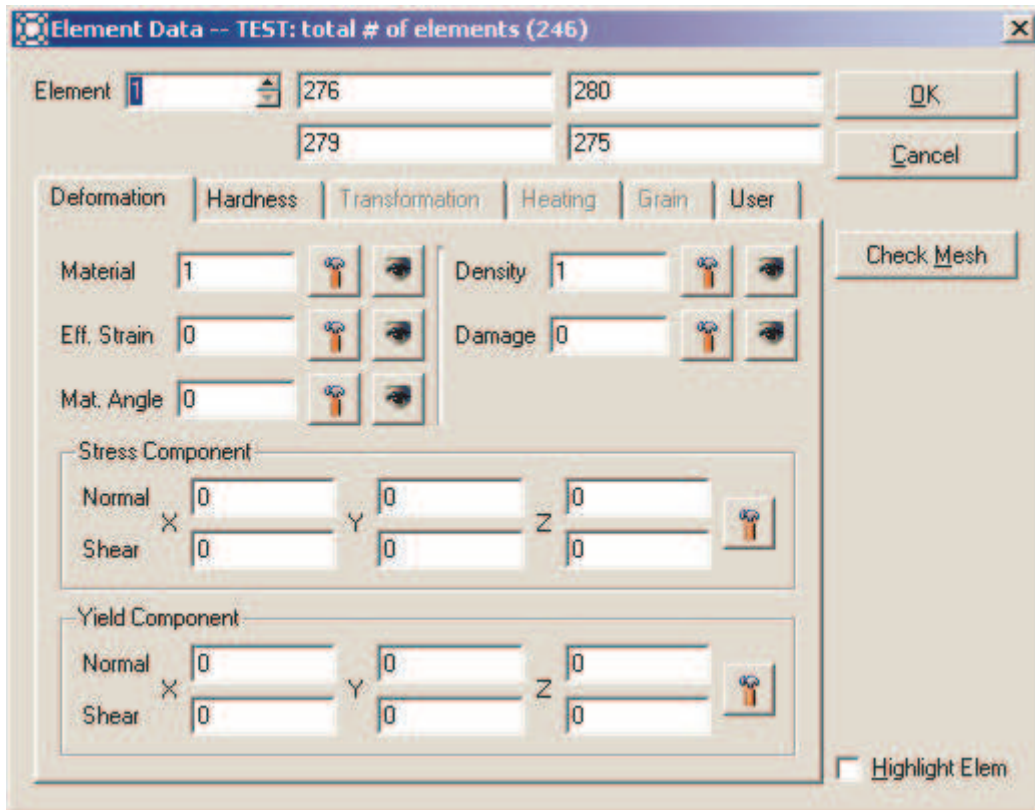


Figure 56: Object element data window.

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. In a plane strain model, X1 is the value of σ_x , X2 is σ_y , X3 is σ_z , and X4 is τ_{xy} . In an axisymmetric model, X1 is the value of σ_r , X2 is σ_z , X3 is σ_θ , and X4 is τ_{rz} .

Yield surface translation tensor (YLDS) [MIC]

YLDS specifies the yield surface translation tensor for kinematic hardening.

Hardness [MIC]

Hardness (HDNOBJ)

HDNOBJ1 specifies hardness and HDNOBJ2 cooling time from high temperature to low temperature for each element.

Transformation [MIC]

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.

Volume fraction limit (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.

Grain size (GRAIN)

Current this is not implemented

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:

$TICF = \frac{t}{t_{incubation}}$ where t is the time increment and $t_{incubation}$ 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.

User variables

User element variables (USRELM)

User element variable values can be defined using FORTRAN subroutines. Refer to User Routines for more information on the subroutines. Each element value may accept both a name and a value. 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. The only caveat is a large number of variables defined can lead to a large database file. The user specified element variables can be viewed, initialized, modified, or tracked.

2.5. Inter Object Definition

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. 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

Also covered in the inter object controls is object positioning and generation of inter object boundary conditions.

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.

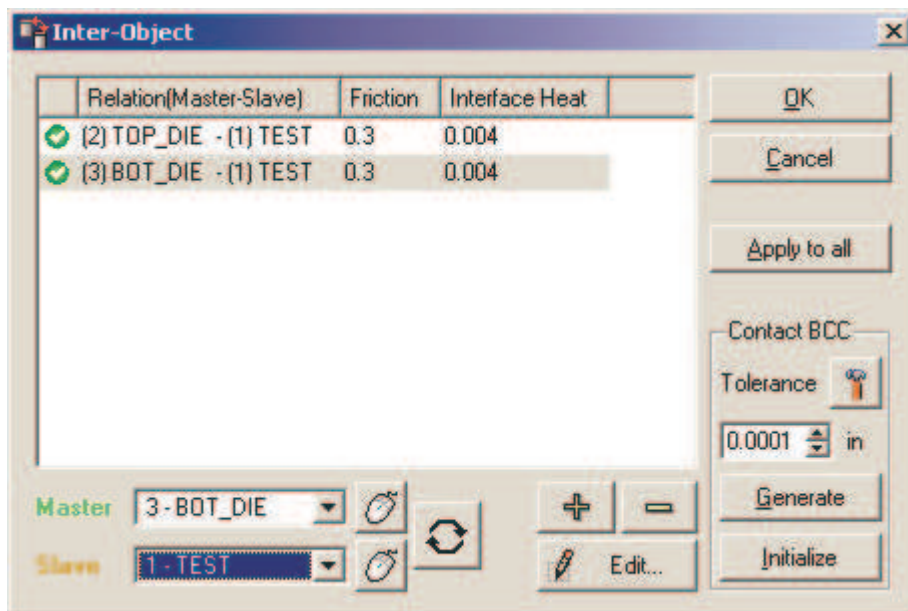


Figure 57: Inter-Object window.

Contact relation (CNTACT)

The contact relation parameter is used to set the Master/Slave relationship between workpiece, 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 for every pair of deformable objects that may contact each other during the simulation.

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 with 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, the node is considered to have separated from the master object.

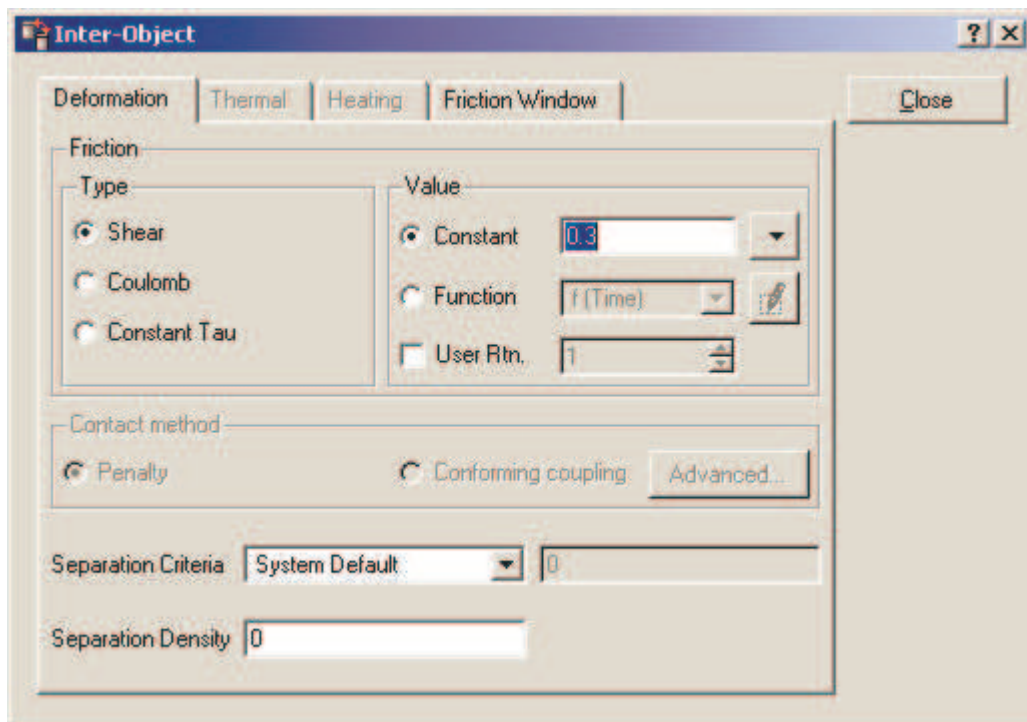


Figure 58: Inter-object settings window.

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 constant shear, coulomb friction, and constant shear stress.

Constant Shear (sticking)

Constant shear friction is used mostly for bulk-forming simulations. The frictional force in the constant shear definition 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 (sliding)

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 objects; 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.

Tau (shear stress)

The Tau model of friction lets the user set the shear stress between two objects in contact due to friction. The value of shear stress the user may set can be a constant or a function of time.

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 workpiece. The friction in turn affects the metal flow at contact surfaces.

Typical values (using constant shear only):

(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.

Note:

- Constant Shear (sticking) friction is used mostly for bulk-forming simulations.
- Coulomb (sliding) friction is used for sheet metal forming operations since it most closely resembles the type of friction encountered in this process.
- Tau friction can be used if the user needs to specify the shear stress on a surface.

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 pct 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.

Friction windows (FRCWIN)

Friction windows allow the user to specify different friction coefficient values for different contact regions of the same object pair. The friction window defines a friction coefficient for the specific area that is defined in the display window.

The friction coefficient specifies the friction encountered by any object in contact with the workpiece. The three friction models allowed are constant shear, coulomb and tau friction. All types of friction can be defined as a constant value or as a function of pressure or time.

Note : If there is not a friction window assigned for portions of objects that are in contact. The global friction coefficient assigned in the inter object interface will be used.

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. 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.

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 Density (SEPDEN)

Separation density is used to model the behavior of porous objects that have not been fully compacted. It defines the separation criterion of contacting nodes involving porous objects. Unless the density of the material is greater than density, nodal separation is not considered.

Relative rotation (FRCROT)

Relative Rotation is used in axisymmetric simulations only. This specifies the rotation difference between two objects about the axis of symmetry. The units are radians/second. For example, if a top die is rotating while moving downward, the interface between the top die and the billet should have the same angular speed (rad/sec) of the top die applied to it. This enables DEFORM to calculate the heat generation from friction due to rotation for processes such as inertia welding. Relative rotation can be defined as a constant or a function of time.

Modeling inertia welding is possible with DEFORM, but is not a straightforward process. Comparison between DEFORM results and known experimental cases should be tested before results for unknown cases are used.

Self Contact

An object can have a self contact condition imposed on itself. This will allow the simulation to continue even if a fold or defect develops. Self contact is defined by selecting the same object in both the Object 1 and the Object 2 list, and assigning a master-slave or slave-master relationship to itself.

Caution: Be careful when using the self-contact condition. If not monitored, a fold might develop and cover itself up during the simulation allowing the user to believe that the process is acceptable when in fact, it may be problematic.

2.5.2. Positioning

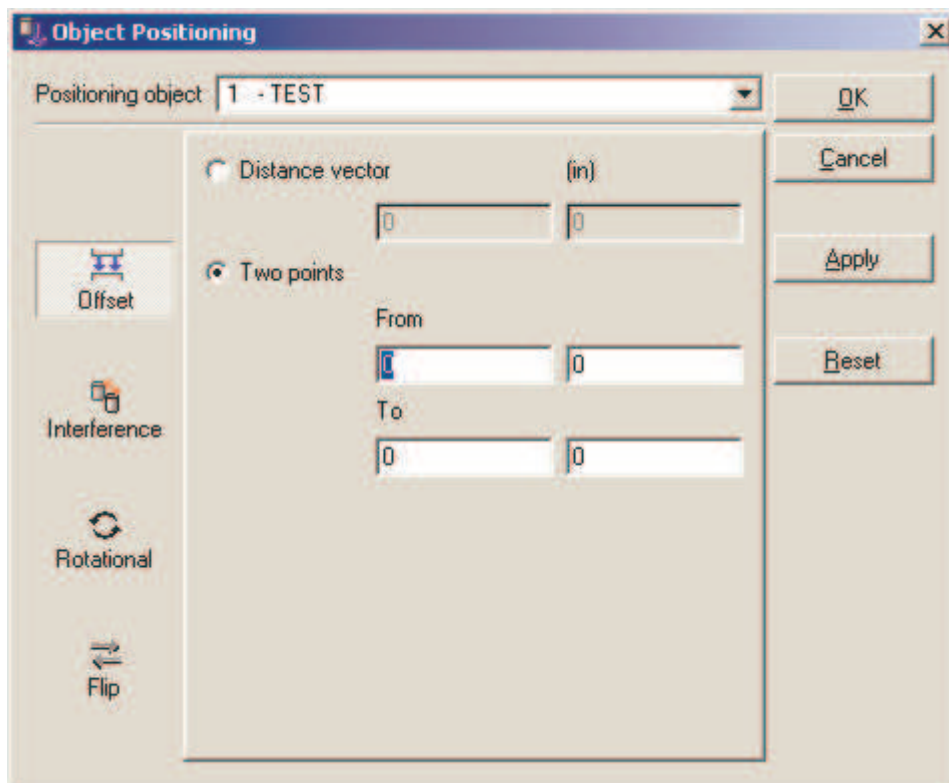


Figure 59: Object positioning window (offset)

Offset Positioning

Offset positioning is used to move the object to position by a given displacement in the chosen direction. One input required is the object to be moved should be highlighted in the position table and the distance the object should be moved should be entered in the distance boxes. The other option for movement is two point positions

Interference positioning

Interference positioning moves the object to be positioned towards the reference object so that they interfere.

Common Question : What is a reasonable value of Interference ?

Answer : The interference should be a small fraction of the height of the smallest element in the mesh before generating inter object boundary conditions.

Rotational positioning

Rotation positioning allows the user to rotate any object. To use this option, select the object that is to be rotated. Select the direction (clockwise, counter-clockwise), the center of rotation, and the total angle that the object will rotate.

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, or nodes which are inside the object, 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.

Note : If master object geometry is improperly defined clockwise instead of counterclockwise, or if the tolerance band is set too wide, error messages will be generated.

2.6. Database Generation

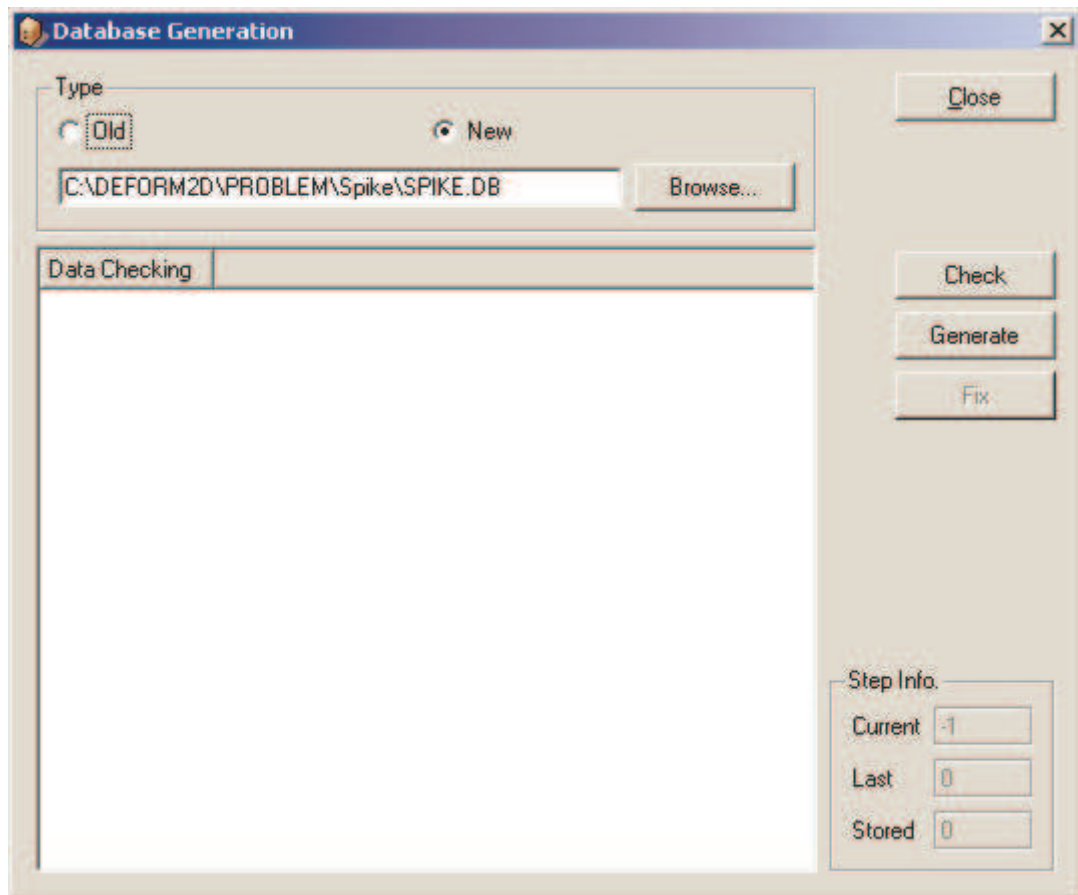


Figure 60: Database generation.

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 will be checked as it is written.

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.

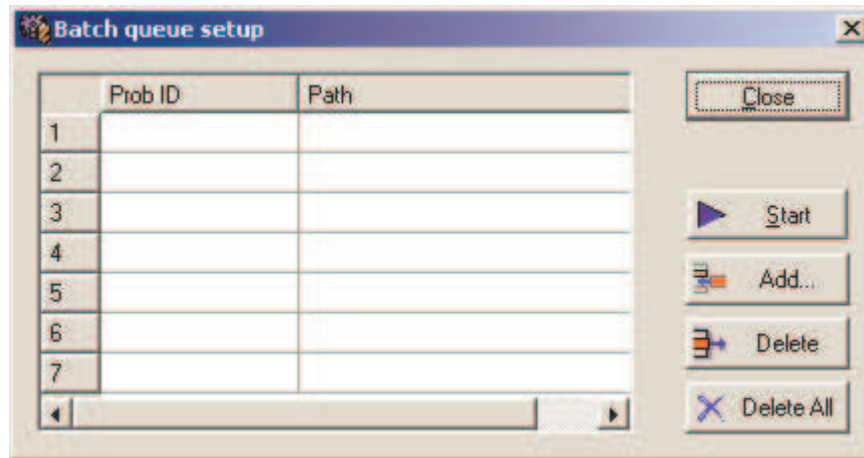


Figure 62: Batch queue window.

3.6. Process monitor

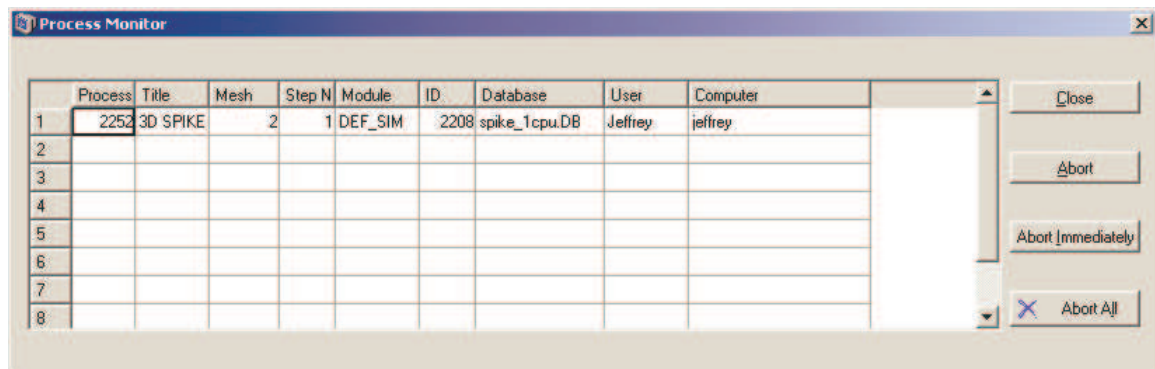


Figure 63: Process monitor.

The process monitor displays the status of any simulations running on the CPU. The process will not be displayed while remeshing is occurring. 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.

These files are found in the main 2D and 3D directories respectively, under the main DEFORM directory. The user needs to have write access to this directory when running these programs. When the user runs the INSTALL2D or INSTALL3D programs, the user will be presented with a number of options. The user should select the first option to generate the status file. This should solve the problem with the Process Monitor.

Note: the behavior of the process monitor is erratic on some operating systems.

3.7. Stopping a simulation

A simulation may be stopped using the Abort 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.

On Windows 95/98/NT press ctrl-alt-delete on the keyboard to bring up the task manager. Find the DEF_SIM.EXE process in the process table, and click on the End Process button.

On Unix, from a terminal window, type the command

```
> ps -ef | grep DEF_SIM.EXE
```

a process table will be displayed. Find the process named DEF_SIM.EXE, and find the associated process id (should be a number up to 5 digits under the column PID.). Use the command :

```
> kill -9 PID
```

where PID is the process ID number.

3.8. Troubleshooting problems

In order to determine why a simulation has not achieved a finished solution, a bit of expertise is required in simulation of forming processes. The three areas in which information can be obtained to determine why a simulation has unexpectedly stopped are the following files:

- **Message file**
- **Log file**
- **Database file**

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. For slightly more information on what an iteration is and why it is necessary, please review the introductory section on fem simulation and the metal forming process. The log file is an account of the module execution

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.

3.8.1. 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 DEFORM_DIR, run INSTALL2D and select the option to regenerate the status file.

Note: If the user gets the message Command not found, the user should run the INSTALL2D by typing ./INSTALL2D. This will force the operating system to override the paths and force the system to run only the file in the current directory.

3.8.2. 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. Every remeshing introduces error in the solution as data is transferred (interpolated) from an old mesh to a new mesh. If this is done at every step then the simulation results may not be accurate. So as a check to prevent remeshing at every step, the FEM engine stops a simulation if this condition is encountered. Also, this prevents the DEFORM system from encountering a neverending 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. The way in which to check for this is to plot the node velocities in the Preprocessor. It is not recommended for the displacement of the nodes each time step be larger than the edge length of the elements.

3.8.3. 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.

3.8.4. Negative 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:

- Any time geometry is imported into DEFORM using IGES (in the case of 2D), it is subject to a number of issues that could make it illegal. Overlapping of endpoints in blended fillets, duplicate points, overlapping lines and other problems can occur. DEFORM has a geometry checking functionality that should always be used on created or imported geometry. If this is neglected, a small mismatch during the simulation can lead to a node (which has no mass) sticking in this area and leading to subsequent remeshing problems. If the problem stops with the negative jacobian message, read a previous starting step into the preprocessor and check all geometry of dies (this includes checking for the orientation of the die geometries by looking at the vertex numbering). If these recognize an error, this is most likely the culprit. The error can be corrected and rerun from that step.
- A closed forging lap will usually lead to this problem. Look at your progressions carefully and see if 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 the following:
 1. 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
 2. Declare the master-slave relationship to itself thus a closed lap will continue the simulation process.
- A die geometry defined by a closed loop must have the starting/stopping point away from the contact zone between the die and workpiece. If this is not the case, it is possible for a node to think it has penetrated the die at

this seam and lead to subsequent serious problems during remeshing.

- 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 substepping criteria, this can lead to a problem resulting from a mesh distorting severely during the first step. Time step size should be small enough to allow for multiple steps between remeshing. In the case of simple or developmental problems, 40 to 50 steps are reasonable. In the case of very complex industrial problems, 200 to 300 steps are not excessive. A complex industrial problem running 25 steps with no substepping is an excellent formula for disaster.
- 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.

3.8.5. Solution does not converge

There are two common reasons for a solution not converging.

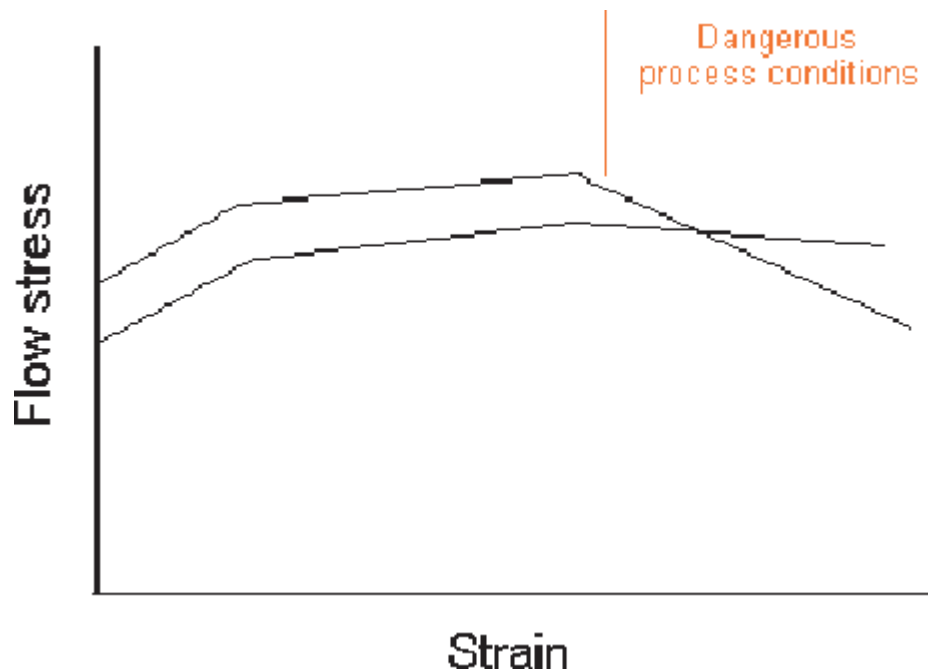
1. The material has a large rigid body motion. Much of the deforming body has a very low strain rate or is rigid.
2. The material is not strain rate sensitive.

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 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.

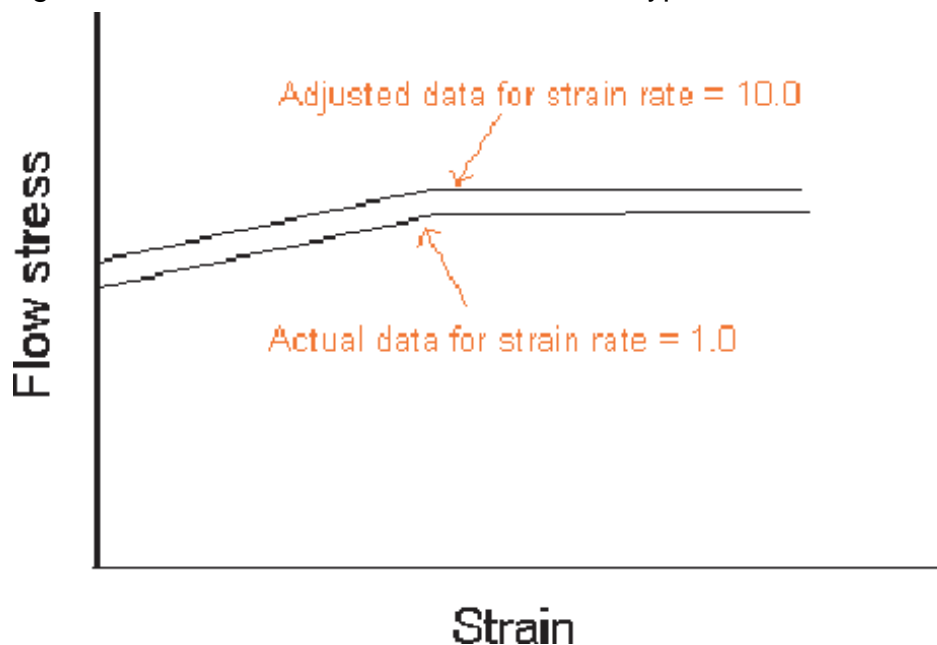
- Check your material data versus your process conditions to insure that no "strange" material properties are being passed to the FEM engine. Be particularly aware of extrapolation issues. For example, if your process conditions are in an area that is outside of the defined flow stress region, this "reverse 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. For the sake of guidelines, very simple developmental problems should run about 50 steps. Simple to average

problems should run about 100 steps. This might include a typical blocking operation or a very simple finisher. Very complex operations such as high reduction extrusions or piercing should run well over 100 steps with 200 being common for industrial problems. This may be accomplished through either controlling the time step or a control modifier that will lead to substepping such as DEMAX or SLDERR.

- 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 Figure 11.2 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 nonconvergence 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.

3.8.6. 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.

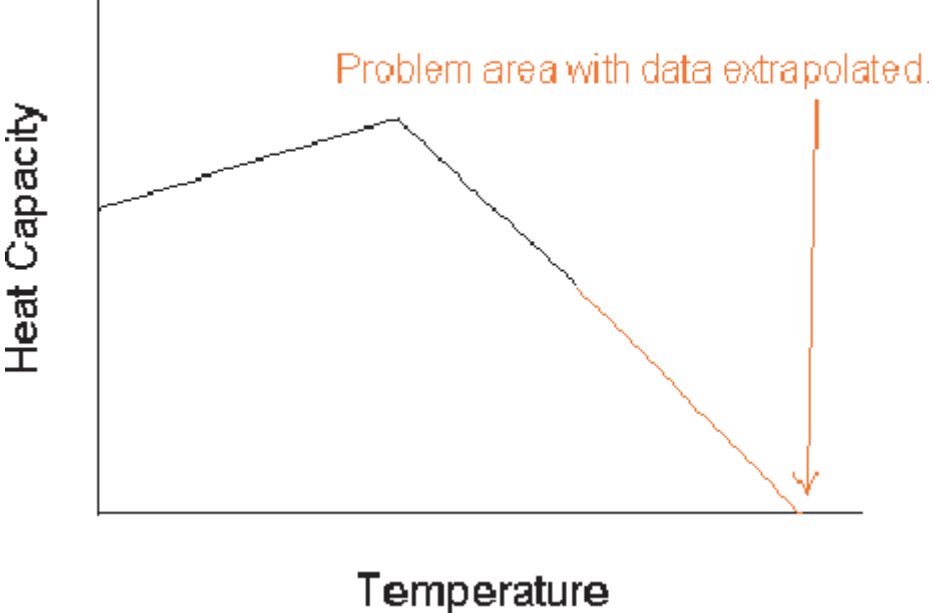
3.8.7. Zero pivot

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.

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 Figure 11.3 for a further description.

Heat capacity vs Temperature



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.

3.9. Database Management

Database Purging

It is now possible to remove steps form a database. This is called **purging** a database. To use this for **mode** select **purging**. Next use the Browse icon next to the text box labeled **Old** to locate a database from DEFORM 7.1 or greater database. Creation information about the data base will be displayed in the text boxes below the database name. The steps in the database will be listed in the Steps scroll region. The **steps highlighted in white will be the ones purged (removed)**. The icons **None**, **Range** and **All** are used to select the steps to be converted. The individual steps can be clicked on to highlight them or UN highlight them.

Once the database for purging has been found, and the steps selected for purging, type in the new database name in the text box labeled **New**. Next click on the **Purge Database** icon. You should have a new converted database.

Chapter 4: Post-Processor

With the new release of DEFORM-2D, we are proud to support OpenGL as an option for the 2D graphics rendering in our Post-Processor. The approach will enable the user to view the results with a much more attractive appearance.

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 that are saved by the simulation engine are available in the post-processor. Information which is available from the post-processor include:

- Deformed geometry, including tool movements and deformed mesh at each 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.
- Graph of any state variable between two points that can be specified by the user. These points can follow the boundary or linearly follow the line formed between the points.

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

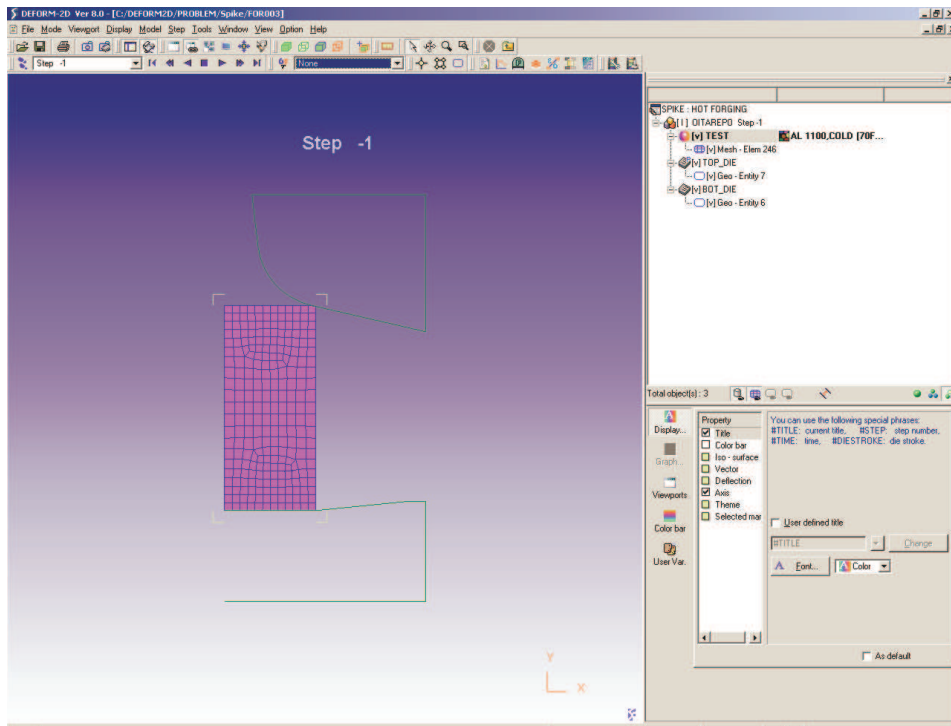


Figure 64: DEFORM-2D Post-Processor.

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. It is the graphical display for all results information, including geometry and meshes, contour and vector plots, and graphs.

Graphic Utilities

The graphics utilities window provides view manipulation and other utility functions for the object window. Features include zoom and pan, 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.

Control Window

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.

Object Display Modes

DEFORM has 3 different object display modes as seen in Figure 65:

- 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.

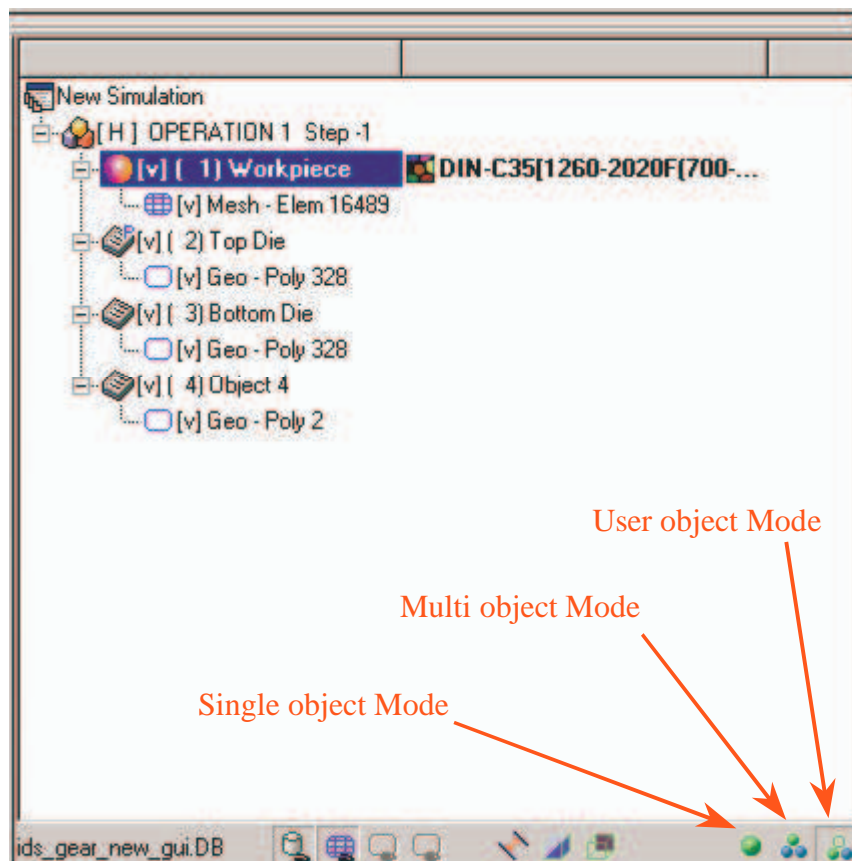


Figure 65: Detail of Object Tree showing object mode selection icons

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 66.

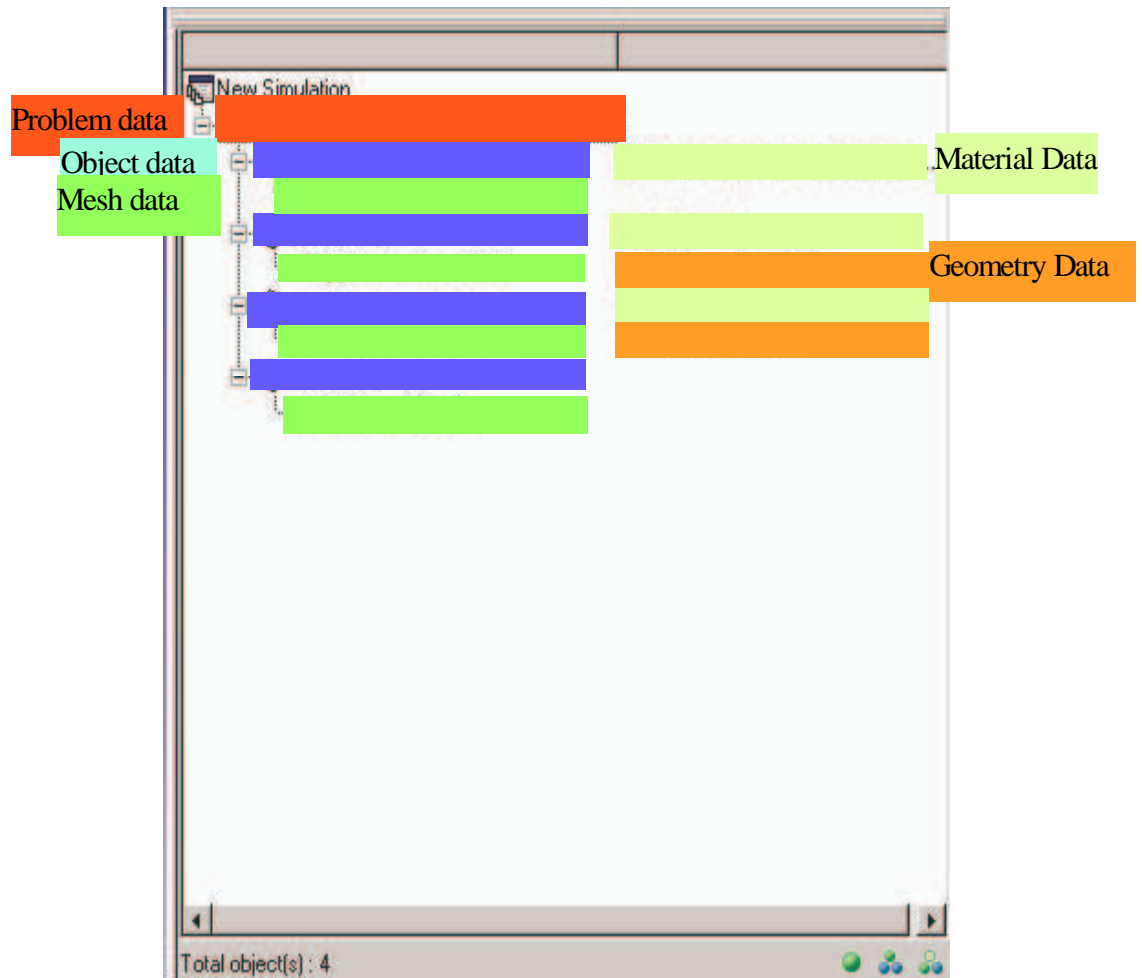


Figure 66: Levels in the object tree. Right mouse selections at each level accesses a context appropriate menu.

The **Problem Data** menu contains the following commands:

- Turn on all objects
- Turn off all objects
- Turn on all workpieces - 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.



-  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

4.3. Display Window

The display window is used to graph data and to aid in the analyzing of the simulation. The display window is shown below.

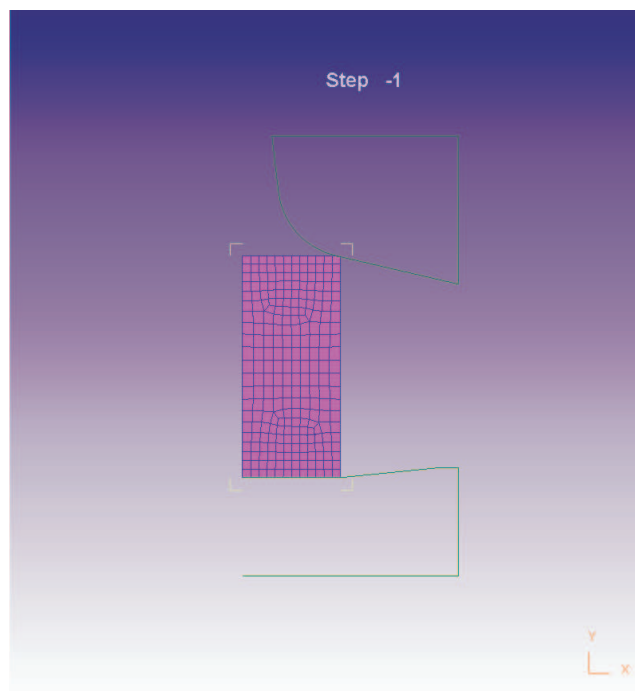


Figure 67: Post-Processor display window.

Most of the common graphical control features in the display window are accessed from the buttons surrounding the simulation heading box and viewport along the top of the window. The viewport is the specific drawing area in which graphics are displayed for the current step. (It is also referred to as the data display box)

The simulation heading box is used to display information of the current database. The simulation title is displayed along with the disk path and name of the database currently loaded. Also, it lists the current step number, date and time.

The data display box is used to display the selected graphical data for the problem.

Clicking on the right mouse button in the display window shows a shortcut to other options such as view manipulation, refresh, fit, plot options, capture image, etc... If mode is clicked, a view manipulation toolbar appears. This tool bar can be moved by dragging it to other parts of the screen. Note that this toolbar is dockable and if dragged to any side of the screen, will combine with that given side.

4.4. Graphic Utilities

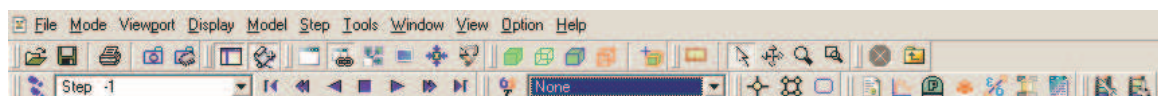


Figure 68: Graphics utilities.

Multiple viewport control

The viewport control icon allows the display of from one to four viewports. Each viewport can display a separate data set. The primary viewport is indicated with a green border. Commands to modify the object view, or to change displayed, will only affect the primary viewport. View sizing by zoom or pan can be copied to all active viewports using the *Same As This* icon in the graphic utilities window.

Select

The select button is a general pointing utility. It performs several functions which vary with context. It can be used to report coordinates of an arbitrary point, to select a node or an element, to change the primary viewport, and many similar functions.

Dynamic zoom

The dynamic zoom dynamically changes the size of the region of the object which fills the active viewport. The view can be changed by left clicking the mouse in the active viewport, then rolling the mouse backward or forward to increase or decrease the amount of the object which is displayed.

Zoom Window

The zoom window function allows close up inspection of a small region of the currently display object or graph. The zoom region is selected by clicking and holding the left mouse button, while dragging the mouse to enclose the selected region with the displayed box.

Pan

Pan adjusts the region filling the active viewport without changing the size of the displayed object.

Measure

The measure feature will display net distance, and x and y component of distance between two selected points. The measure is performed by clicking and holding on the first point, then releasing over the second point.

The measurement marks are not erased until the viewport is refreshed, and as such are also convenient for marking arbitrary reference points on the screen, which will maintain their location through view changes.

Refresh

The refresh icon redraws the screen, removing any measurement marks, and updating any changes made to the color pallet.

Fit all

Fits all displayed objects or graphs inside the current viewport.

Same as

Copies the view size of the primary viewports to all other active viewports.

Print

Prints to a postscript printer or creates a postscript file of the current step.

Print all steps

Prints all selected steps to a postscript printer or file. Steps may be selected or deselected using the *Steps* option on the post-processor controls window.

Capture image

Writes the current screen image to a file using the specified image format.

Animate

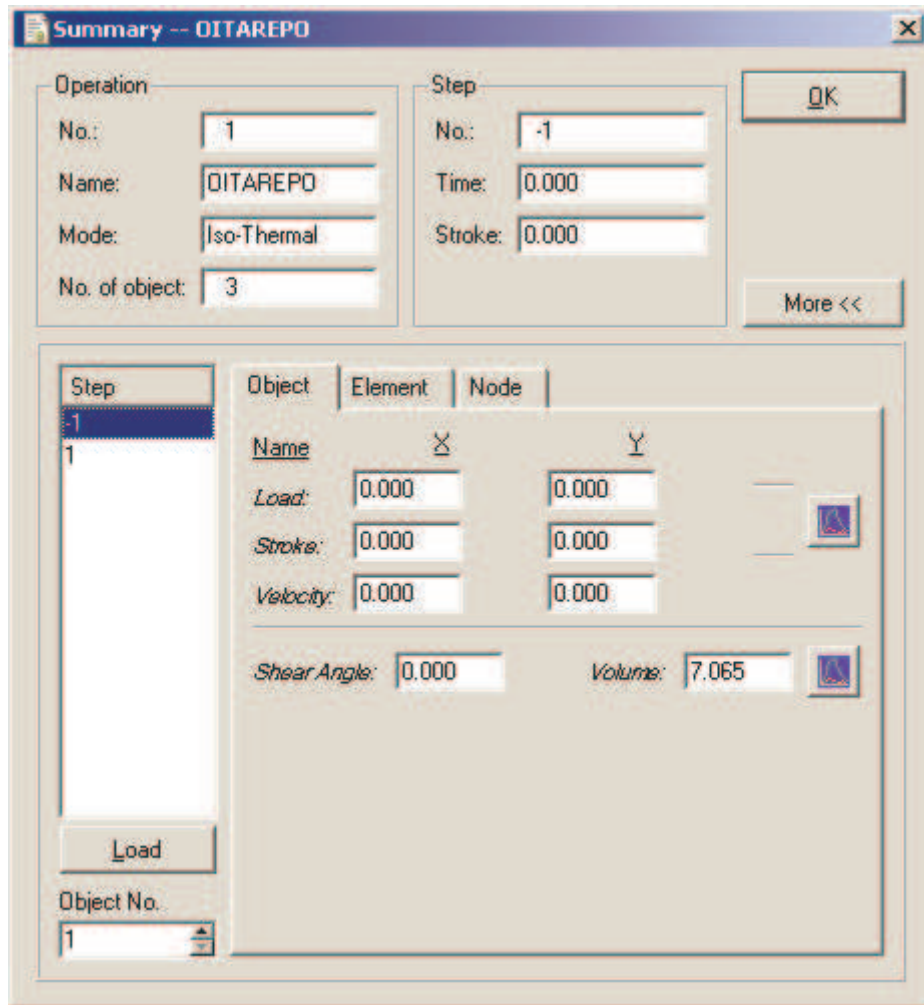
Writes images of all selected steps to a an image file using the specified image format. Steps may be selected or deselected using the *Steps* option on the post-processor controls window.

4.5. Post-Processing Summary

The post-processor controls window is divided into two sections. The top sections consists 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.

4.5.1. Simulation Summary

This is very similar to the old DEFORM simulation summary. 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, 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.



Simulation Summary Window

4.5.2. Object Mirroring

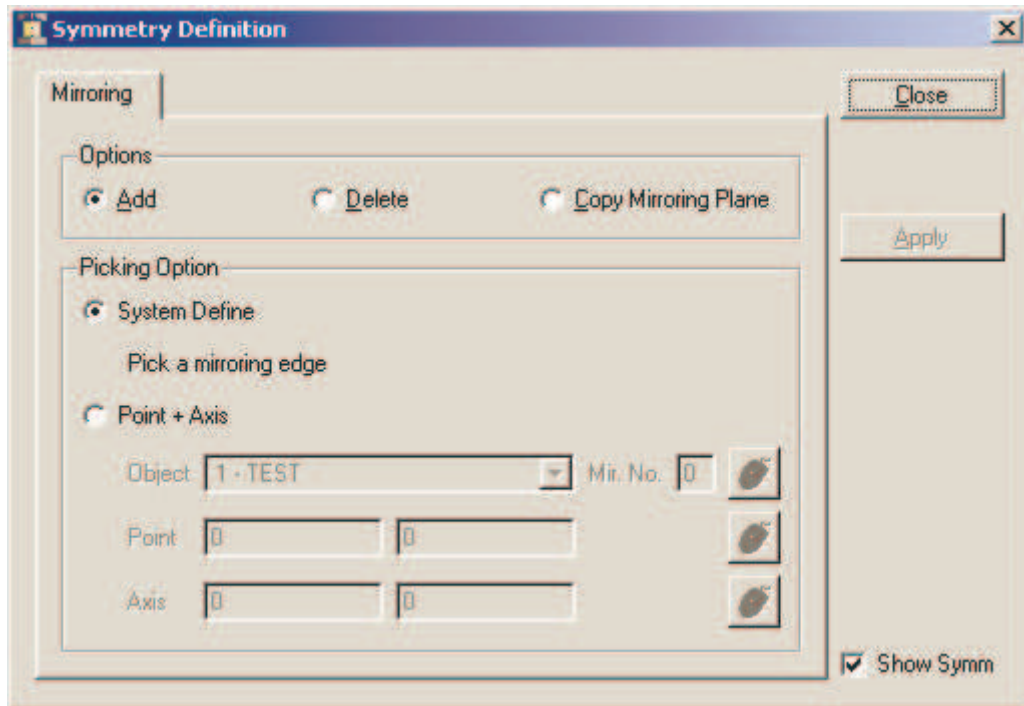


Figure 69: Object mirroring window.

The purpose of object mirroring and symmetry is to allow the user to visualize the both sides of the centerline of a part. If a part lies on the centerline, symmetry can be added by clicking on the add option and picking the region of the part that touches the centerline. If the part does not lie on the centerline, there are two ways in which to mirror the part.

1. Click the Point + Axis button under the Add tab and specify the centerline of the part by selecting the proper object, setting point to (0,0) and set the axis to the centerline axis (0,1). Click Apply to see the symmetry.

2. Click the Copy Mirroring Plane button and copy an existing symmetry line from an already mirrored object to an unmirrored object. Click Apply when finished.

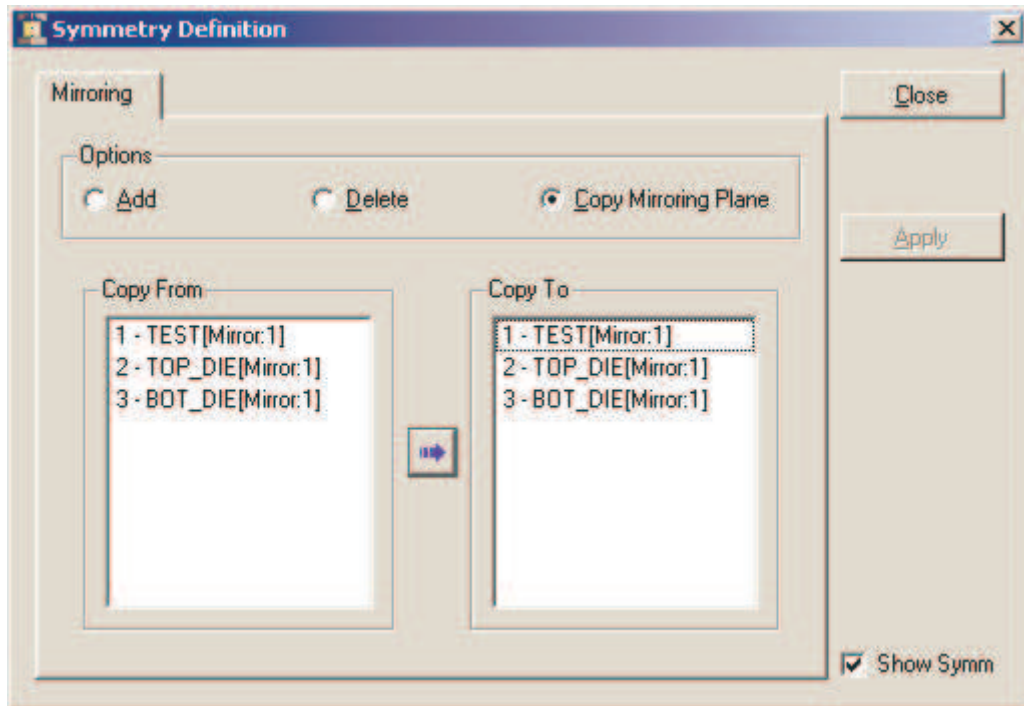


Figure 70: Copy mirroring plane window.

4.5.3. State Variables

In order to plot state variables, the variable and the component of the variable must be selected. Also, one of the scaling types: 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.

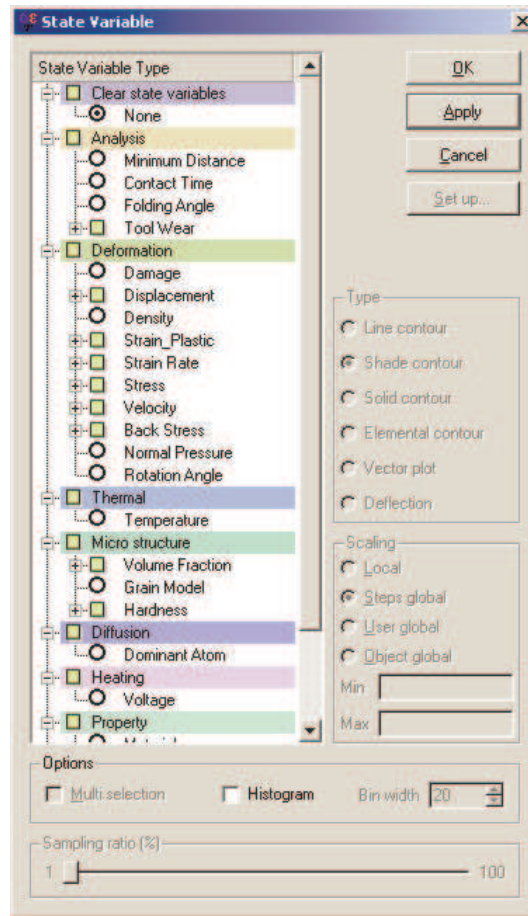


Figure 71: State variables window.

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

Shows the state variable with each element shaded with a constant color with very discrete color transitions of the color bar. This makes it easy to extract values from each element and to visualize low and high value regions.

Elemental Contour

Shows the state variable with each element shaded with a constant color with very smooth color transitions of the color bar. This makes it easy to visualize low and high value regions.

Vector Plot

This plots variables as vector quantities, i.e. with a scale and a direction. Only certain variables are applicable to be plotted in this manner such as velocity and displacement.

Advanced Plotting Options

The advanced plotting options changes different plotting values such as pixel resolution, vector size etc...

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.

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.

User Variables

User defined variables allows one to plot any user defined variable in the post-processor.

Usage

State variable plots display contour or vector plots of each of the state variables.

Line contours display colored contour lines at each contour level. Shaded contours fill the entire object geometry with color representing solution values. The data ranges from a specified minimum to maximum data values.

The colors can be modified by selecting the *Options* ⇒ *Colors* ⇒ *Shaded Contours* option from the topline menu on the controls window. Colors can be changed by double clicking on the *Base contour colors* color block and selecting a new color from the base pallet.

Scaling

The minimum-maximum range of contour plots can be based on:

Local

The maximum and minimum state variable values for the current step are used. This provides better resolution for a single step, but will cause colors to shift over multiple steps, making it difficult to track the evolution of variables.

Global

The maximum and minimum state variable values are taken from all selected steps. This makes it easier to track evolution of a variable over several steps, but sacrifices resolution on a single step.

User defined

The user can specify maximum and minimum contour levels. This is particularly useful for state variables exhibiting small regions with extreme values. Resolution of the remaining area may be lost because of autoscaling to an extreme peak. Resetting the maximum and minimum values to more reasonable values will wash out the peak values, but give much better resolution to intermediate values.

Interpreting state variables

Damage

Damage generally relates to the likelihood of ductile 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 (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 which means that the material is 100% dense.

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 change in length by original length that is fine for small deformations, but begins to break down 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

$$\epsilon = \ln \frac{l_f}{l_0}$$

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 type of strain saved is the total plastic strain.

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. Strain rate is defined as the instantaneous plastic strain rate.

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 components 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.

Back Stress

Back Stress is defined as the translation of the yield surface caused by

loading of a material. This requires the user to use a kinematic hardening rule. In large deformation processes like forging, this is usually negligible.

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.

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

Grain size is not implemented in DEFORM release 7.0. It will be implemented in a future release.

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.

Material

The material group of a given object can be plotted. In most cases, this will show the object as a solid color unless multiple material groups have been defined for the same object.

User Variables

User defined nodal variables can be stored using user defined FORTRAN subroutines. Refer to the section on user subroutines for more information.

4.5.4. Flow Net

The flownet is a post-processing tool that traces a pattern through deformation states. The flow net window allows the user to apply a pattern to an object and observe how the pattern deforms as the simulation progresses. As the mesh deforms, so does the pattern, however, unlike the FEM mesh, the pattern remains intact throughout the remeshings. Thus, the flownet is much like physically etching a pattern on a cross section of a workpiece and having the ability to view the workpiece at various stages of the formation process. The start step list will contain all of the steps in the display window currently loaded by default. Select the starting step for defining the flow net and then select the object you want to have the flow net. In order to generate a pattern for the given object, click on the flownet patterns. It should be noted that flownet patterns can be generated for plastic and porous objects.

How to define Pattern Generation

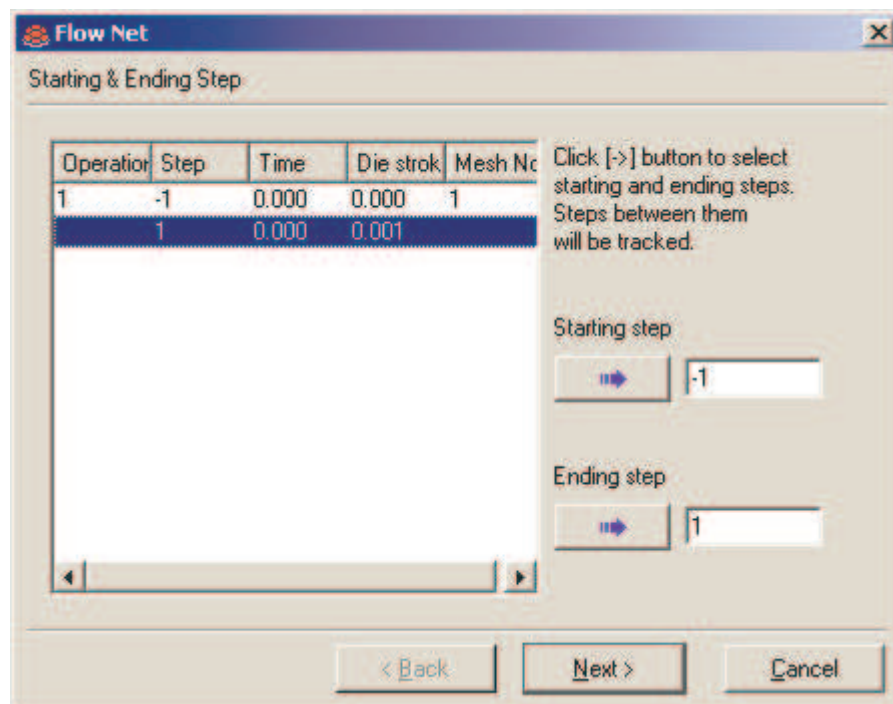


Figure 72: Flow Net Window (initial).

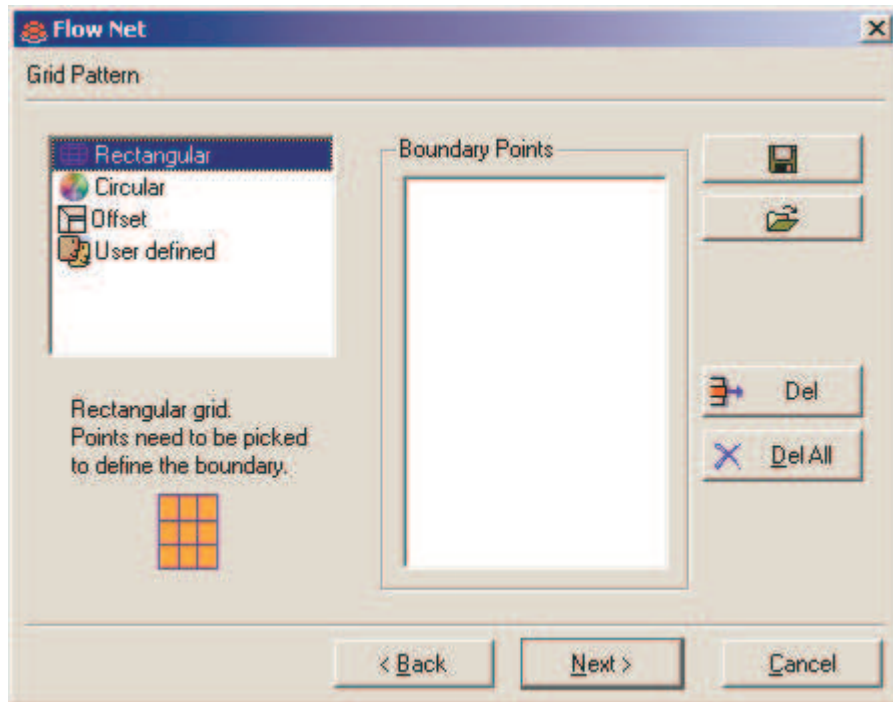


Figure 73: Flownet pattern generation.

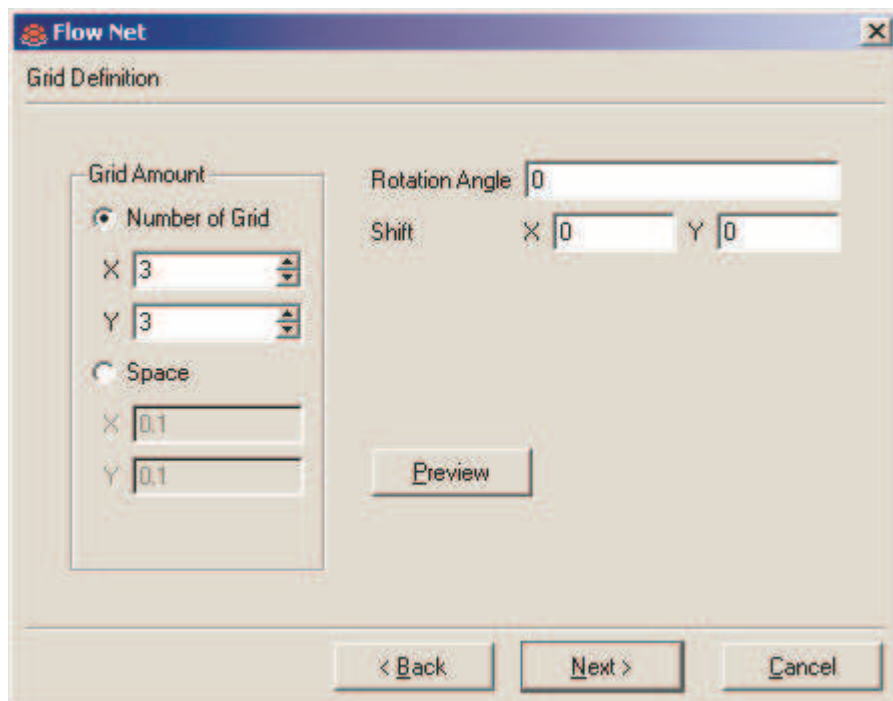


Figure 74: Flownet grid settings window.

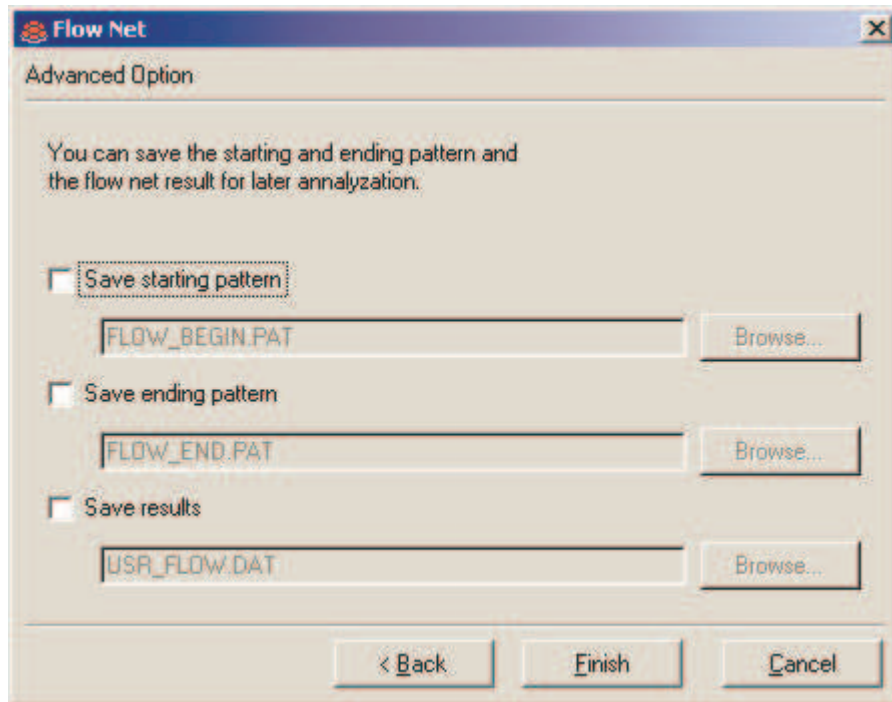


Figure 75: Advanced flownet settings.

The flownet is a post-processing tool that traces a pattern through deformation states. The flownet window allows the user to apply a pattern to an object and observe how the pattern deforms as the simulation progresses. As the mesh deforms, so does the pattern. However, unlike the FEM mesh, the pattern remains intact throughout the remeshings. Thus, the flownet is much like physically etching a pattern on a cross-section of a workpiece and having the ability to view the workpiece at various stages of the forming process. The start step list will contain all of the steps currently in the display window. Select the starting step for defining the flow net and then select the object if you want to have the flow net. In order to generate a pattern for the given object, click on the flownet patterns.

A pattern can be tracked backwards by selecting a later step in the database as the starting step, then selecting an earlier step as the ending step.

Rectangular Grid

The rectangular grid option will generate a grid composed of perpendicular lines within the desired region. Rectangular patterns are typically used when the material texture is of interest. This pattern is very similar to the grain flow which would be seen if a cross section of the part were etched. If a rectangular pattern

is selected, you must specify the grid origin, orientation angle, and line spacing in X and Y. The grid can be controlled by the following parameters:

Origin This is the origin in X/Y coordinates at which the grid will begin to be generated.

Grid Spacing This is the distance (DX and DY) between each grid point in the X and Y directions.

Angle This will determine the angle at which the grid will be drawn in degrees.

Circular Grid

The circular grid can generate a field of circles of a given radius and distance apart within the desired region. Circular patterns are typically used to monitor directional orientation of flow. If a circular pattern is selected you must specify the grid origin, circle radius, center to center distance between circles, number of segments in the circle, and whether clipped circles should be included. The circular grid is controlled through the following parameters:

Origin This is the origin in X/Y coordinates at which the grid will begin to be generated.

Radius Specifies the radius of each circle.

Center-Center Specifies the vertical and horizontal distance between each center of neighboring circles.

No. Segments Sets the number of line segments the circles will be composed of. (The circles are actually constructed of three or more line segments, default is 12)

Inc. Clipped Includes any partial circles that were clipped by the region's boundaries. (default is Yes)

Offset Curve

The offset curve will draw an identical surface at a specified distance within the objects border. The border offset pattern is typically used to capture tendencies toward lap formations. If a border offset is selected you must specify the distance the border is offset. The offset curve is controlled through the following parameter:

Offset Distance: The offset distance is a positive value specifying how far inside the region the identical border should be positioned.

User Defined

If you wish to create your own initial pattern or use a previously generated

pattern, you may do so by reading in a flow pattern file. A pattern file consists of a list of point coordinates and a list of connectivity sets. The points are points of intersection within the flownet. The type of grid is determined by the connectivity list. This list determines each curve separately as the number of points in a sequential pattern. If the starting and ending point indices are the same, the curve is closed.

The user defined pattern must be read by a pattern file. The pattern generated can be saved to this file by setting the save option to yes. If the pattern is being saved, the file name must be specified in the pattern File text box

The material point data format is:

```

Numpt
1          X(1)      Y(1)
.          .         .
.          .         .
Numpt  X(Numpt)  Y(Numpt)
NumCv
1          CvSz(1)   pt(1)      pt(CvSz(1))
.          .         .         .
.          .         .         .
NumCv   CvSz(NumCv) pt(NumCv) pt(CvSz(NumCv))

```

where

- Numpt**: Number of material points
- X(i)**: X coordinate of ith material point
- Y(i)**: Y coordinate of ith material point
- NumCv**: Number of curve
- CvSz(i)**: Number of points in ith curve
- pt**: Point index of curve (refers to indices in first list)

Example Case:

Taking a round billet from the SPIKE.KEY file in the DATA directory and placing a 3x3 rectangular grid on it will yield an initial flownet that resembles Figure 76. This flownet is stored as the file seen in Figure 77. As seen in Figure 77, the grid points are first stored and the connectivity is stored in the 2nd section. The grid points are seen in as labeled in Figure 76a and the connectivity curves are labeled as Figure 76b.

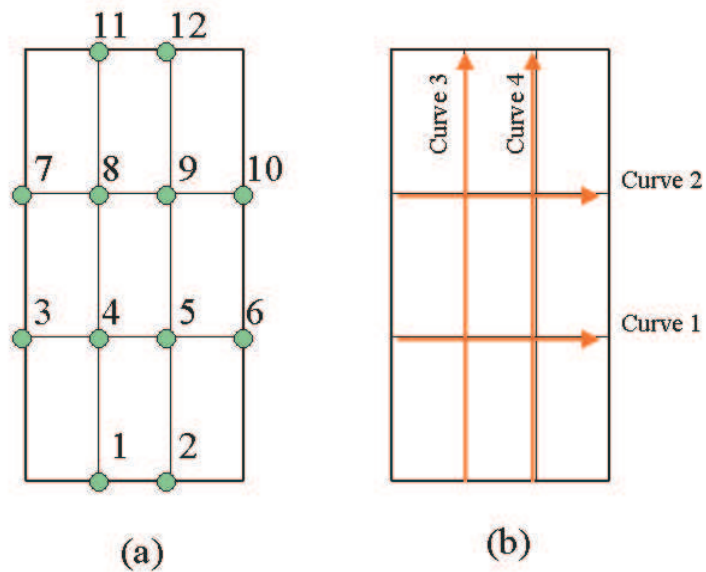


Figure 76: The result of placing a 3x3 flownet on a round object. The points are labeled in (a) and the curves are labeled in (b).

12					
1	12.70000000000000	0.00000000000000			
2	25.40000000000000	0.00000000000000			
3	0.00000000000000	25.40000000000000			
4	12.70000000000000	25.40000000000000			
5	25.40000000000000	25.40000000000000			
6	38.10000000000000	25.40000000000000			
7	0.00000000000000	50.80000000000000			
8	12.70000000000000	50.80000000000000			
9	25.40000000000000	50.80000000000000			
10	38.10000000000000	50.80000000000000			
11	12.70000000000000	76.20000000000000			
12	25.40000000000000	76.20000000000000			
4					
1	4	3	4	5	6
2	4	7	8	9	10
3	4	1	4	8	11
4	4	2	5	9	12

Figure 77: Sample pattern file corresponding to example case. Compare the point connectivity to the figure above.

Defining a Region

A flow net region is defined in the display window. It can be selected within an object's border, or within a specified region within the object. In order to define a region, you must add at least three points within the object's border. Once the points have been defined, just select generate pattern in the pattern generation \ tracking window.

Add Points

Delete Points

Delete All Points

4.5.5. Point Tracking

In point tracking you can track up to 200 points over the course of a simulation. In order to use point tracking, first select the desired starting step from the start step list in the point tracking window. Next, you should select the object from the object table. Once this has been accomplished, click on the Define Material Points button. You can add the points that you wish to track in the display window.

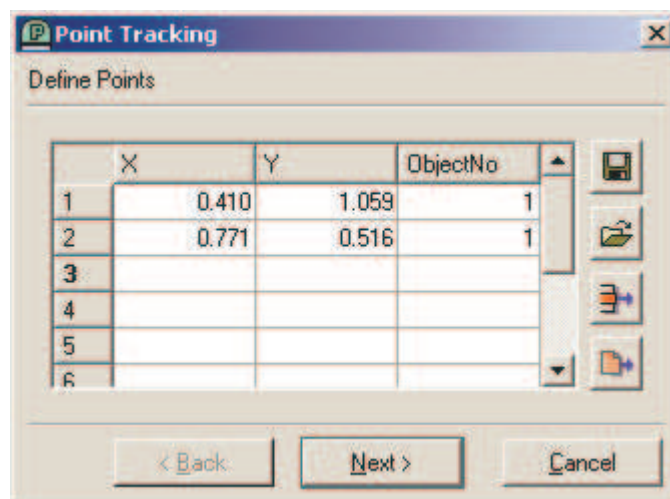


Figure 78: Point Tracking initial window.

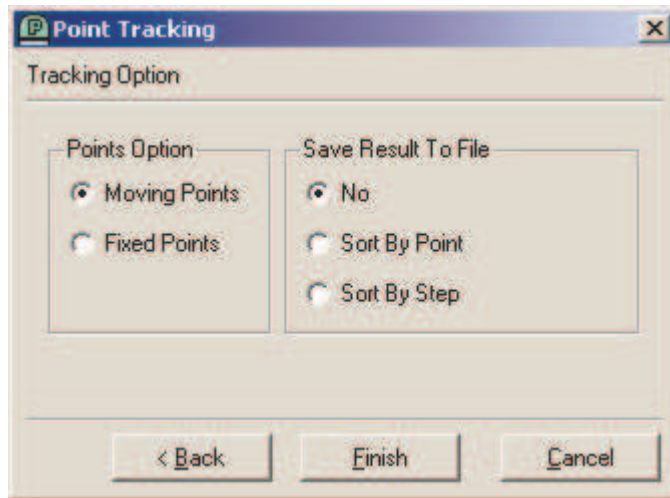


Figure 79: Point tracking (tracking option window).

4.5.6. Load Stroke

The graphs window is used to generate load, speed, torque, angular velocity, and volume vs. time (or stroke) plots for the objects. 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 up the closest saved step from the database (of the list of steps which are displayed on the step list). The default object that is selected is the primary die and the default load is the Y Load.

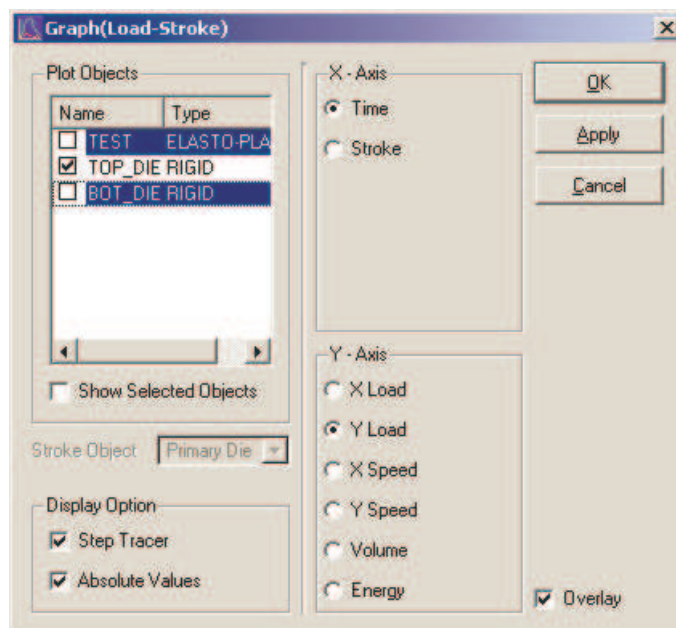


Figure 80: Load-stroke window.

It is possible to plot a load stroke curve for one object by selecting the object in the object table. It is also possible to plot the load stroke curve for the primary die. In order to do this turn Follow PDIE on.

By default when one plots a load stroke curve for a multiple operation problem. All load stroke curves will appear on the plot. It is possible to plot one at a time. In order to do this turn Follow Oper on.

4.5.7. Steps

The step options such as adding and subtracting steps from the step list are accomplished from within this window. By default when DEFORM loads up, only positive steps are displayed in the step list.

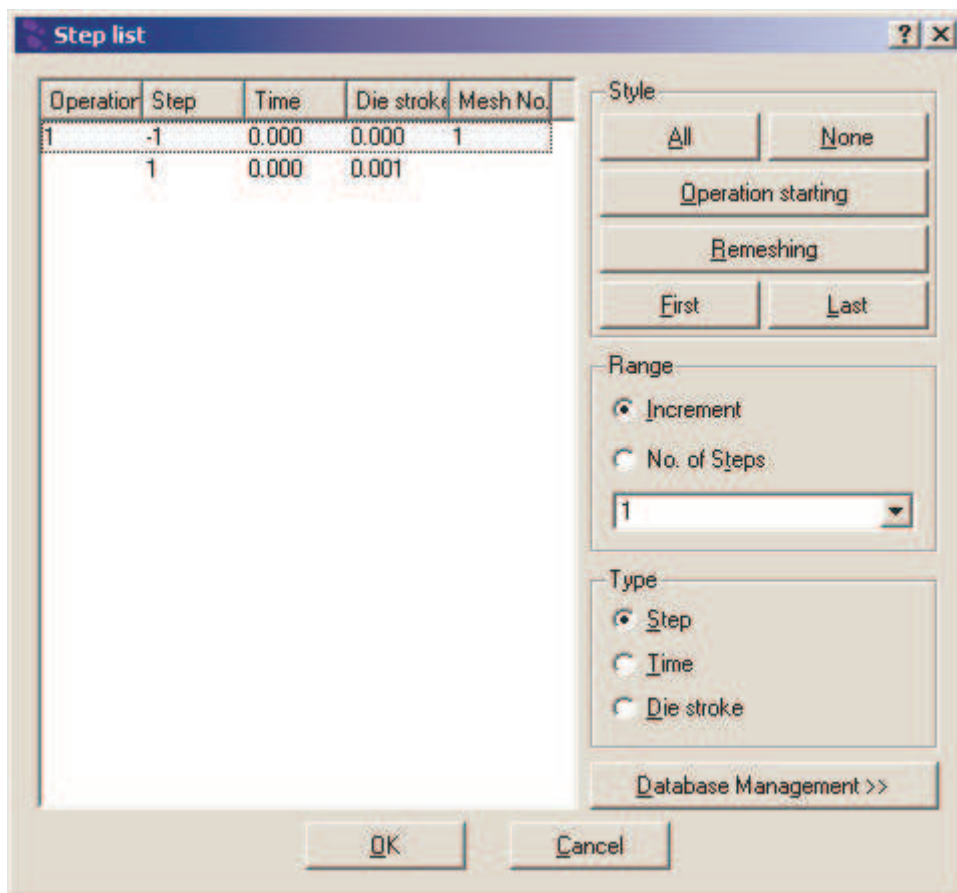


Figure 81: Step Selection Window.

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

None

No steps are selected for use in the Post-Processor.

Toggle Remeshing

Toggle 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.

All Steps

All steps will be selected for use in 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.

4.5.8. Database Management

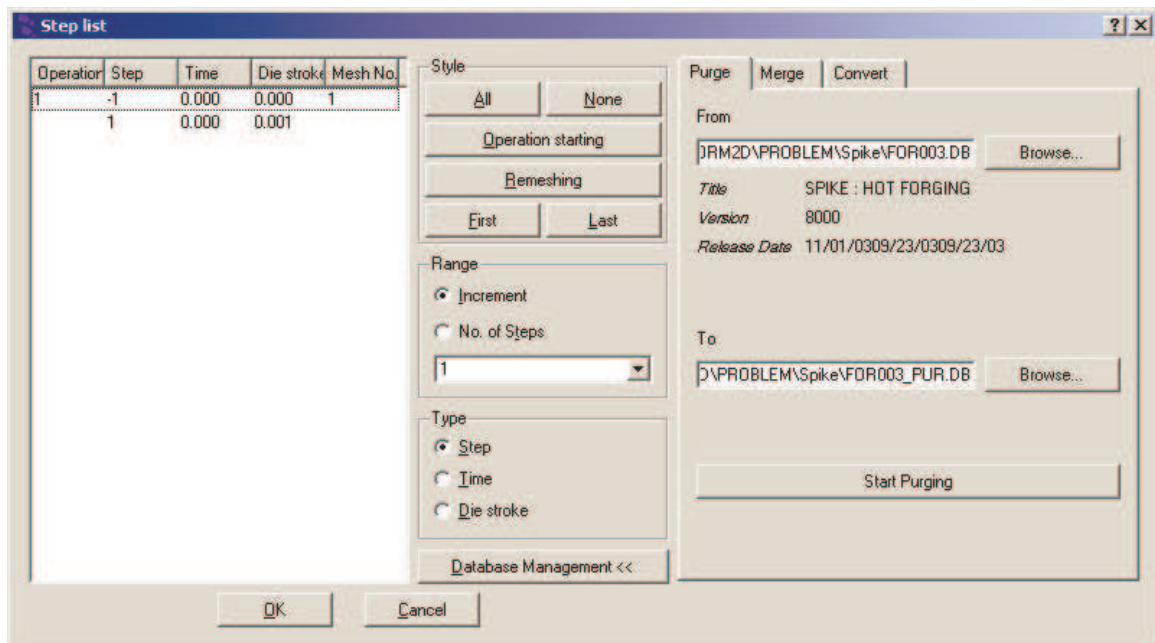


Figure 82: Database management window.

The database management window has the purpose of providing database purging.

4.5.9. Nodes window

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.

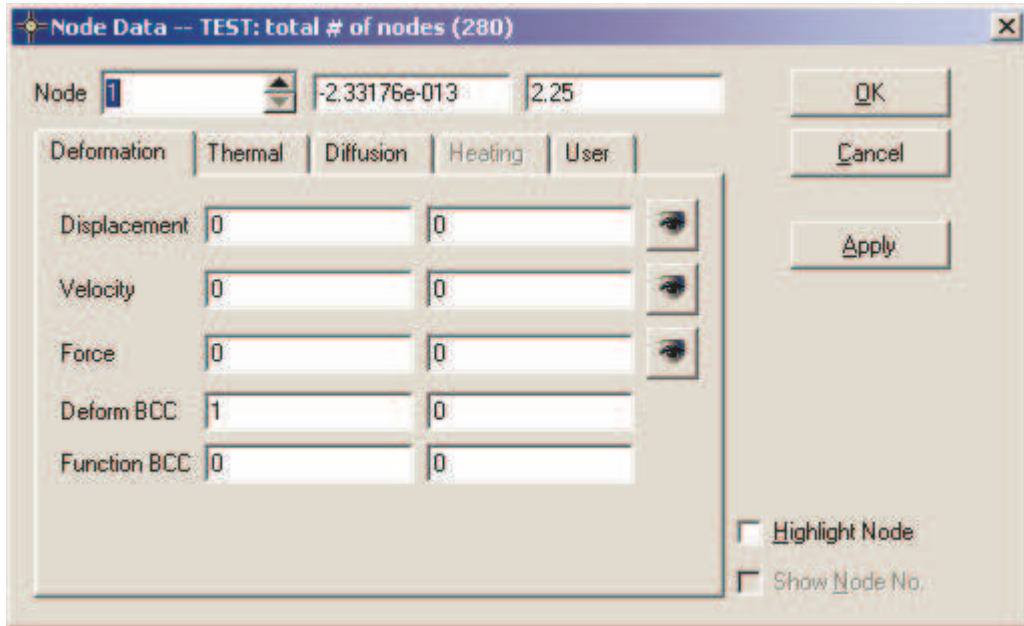


Figure 83: Nodes data window.

4.5.10. Elements window

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.

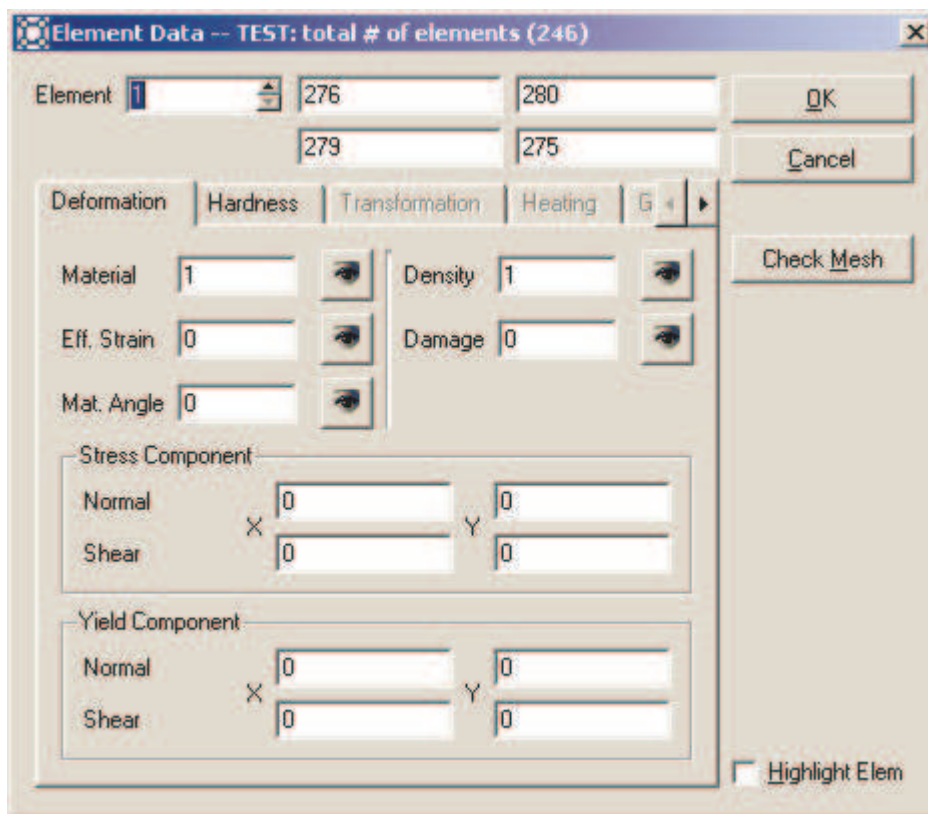


Figure 84: Elements data window.

4.5.11. Object Edges

Some values are defined on the boundary of each object. These values are known as element edge data. The value for these conditions is available in the object edges window. In many cases, boundary conditions act upon the surface of an object rather than on a node or an element, such as a pressure. This is one such case where the definition is edge-based. The data for this information is available in the object edges window.

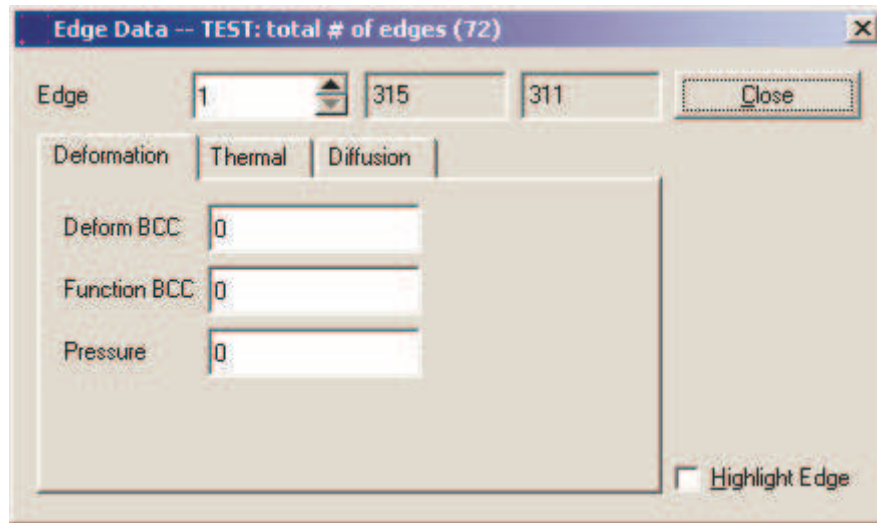


Figure 85: Object edges window.

4.5.12. 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. Many different options are available through this window.

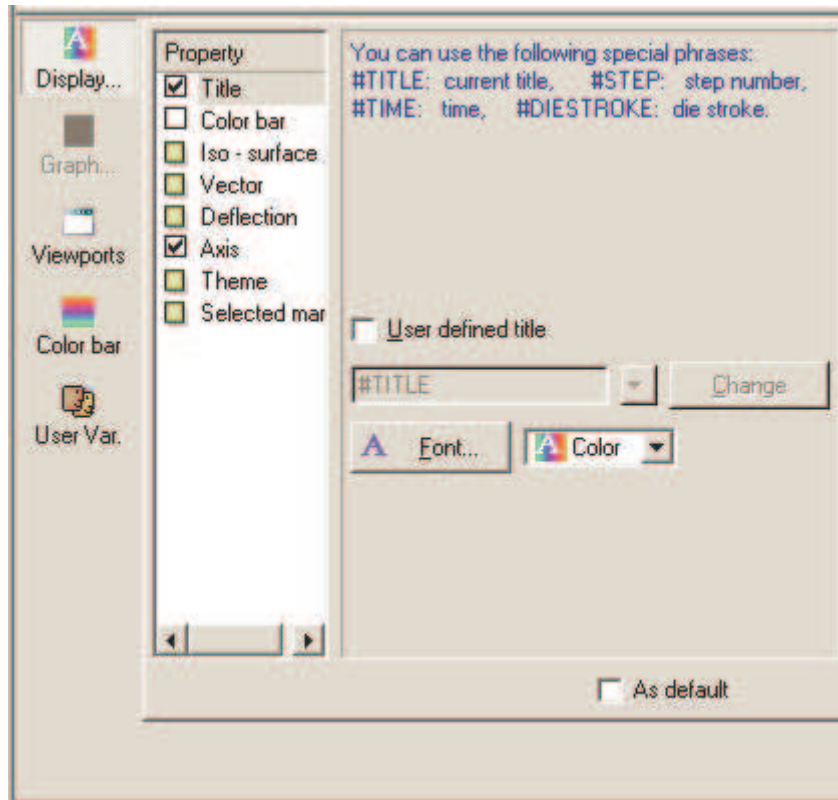


Figure 86: Viewport options window.

4.5.13. Data Extraction

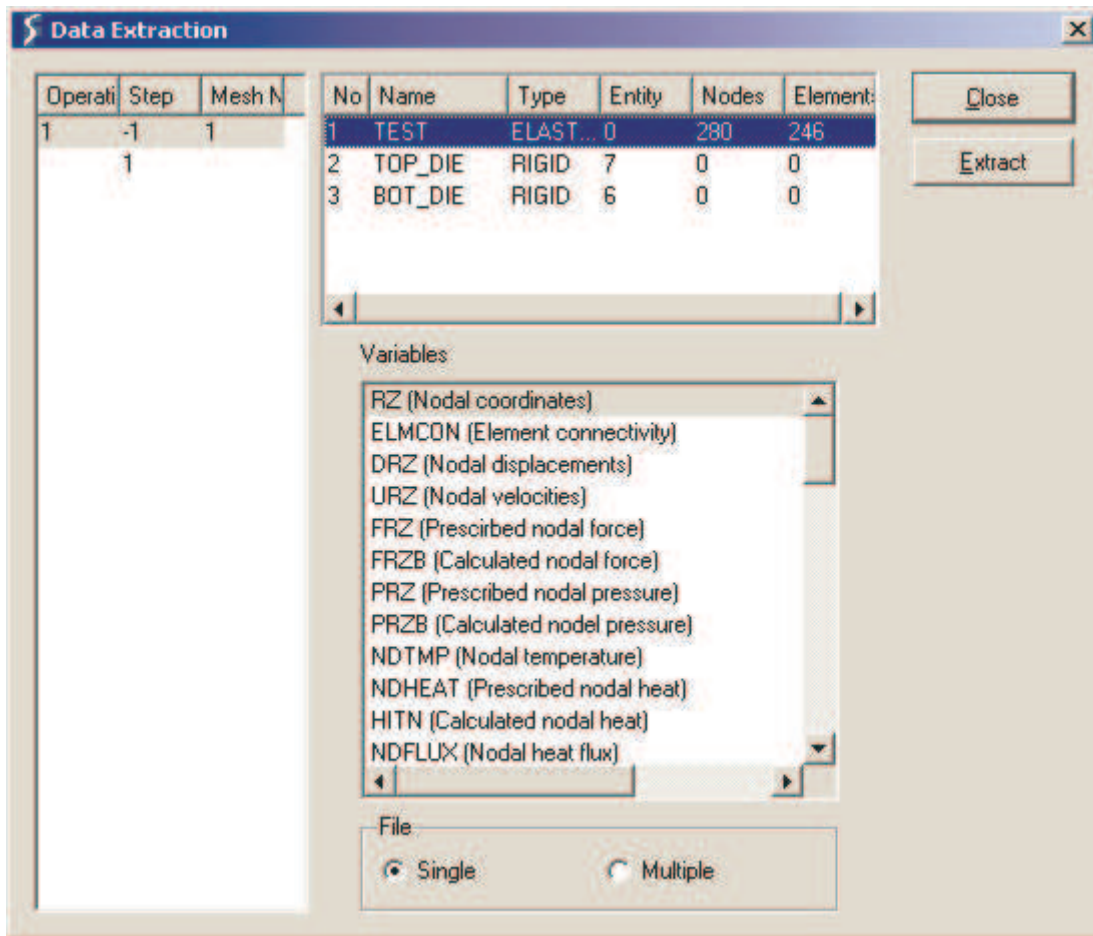


Figure 87: Data extraction window.

Data Type

Information can be written out in a keyword format to an output file. The following items listed below can be extracted.

Simulation Controls

Object Data

User Defined Variables

Material properties

Inter 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 the terminal of a file.

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.

4.5.14. State variable distribution between 2 points.

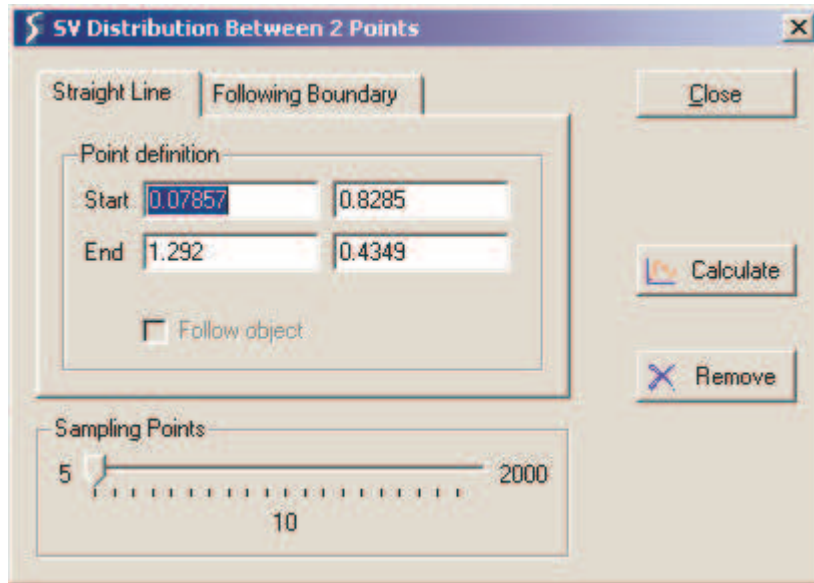


Figure 88: State variable distribution between 2 points.

This feature allows the user to specify two points and to plot a given state variable distribution between those two points. The state variable distribution can either linearly interpolate between the two points or follow the boundary of the object.

Step 1:

Click on the SV distribution button in the Post-Processor.

Step 2:

Select whether to interpolate along a straight line or to follow the boundary.

Step 3:

Pick two points on the object to interpolate along.

Step 4:

Click the Calculate button.

If no state variable is currently selected, only a line of points will be shown. After a state variable is picked from the state variable window, reclicking the Calculate button will show a graph of the distribution between the two points.

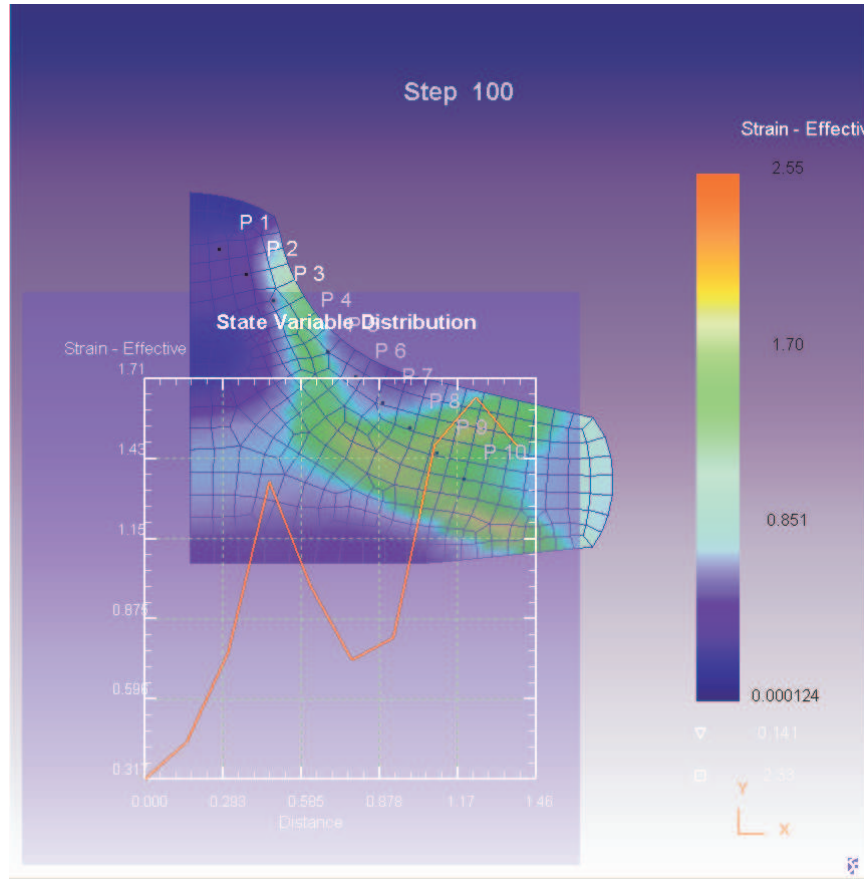


Figure 89: State variable distribution between two points.

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 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.

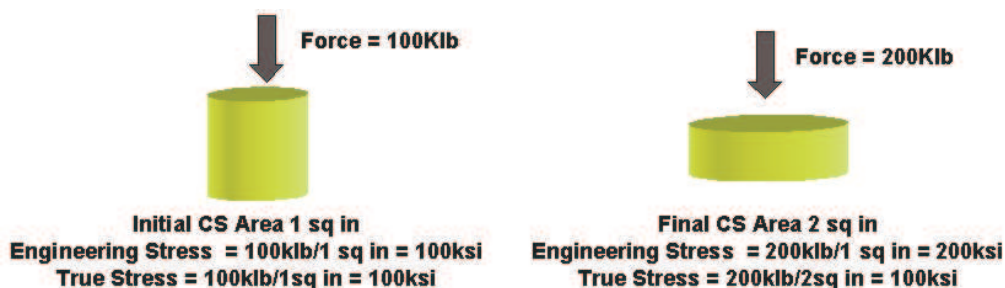


Figure 90: Demonstration of concept of stress.

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 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.

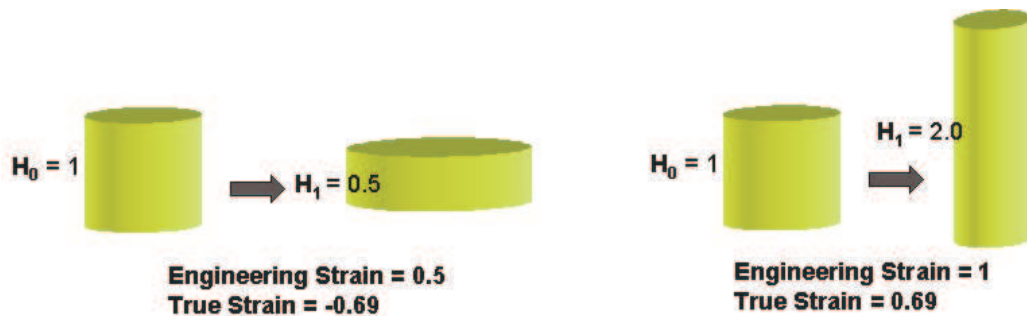


Figure 91: Demonstration of concept of strain.

In Figure 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.

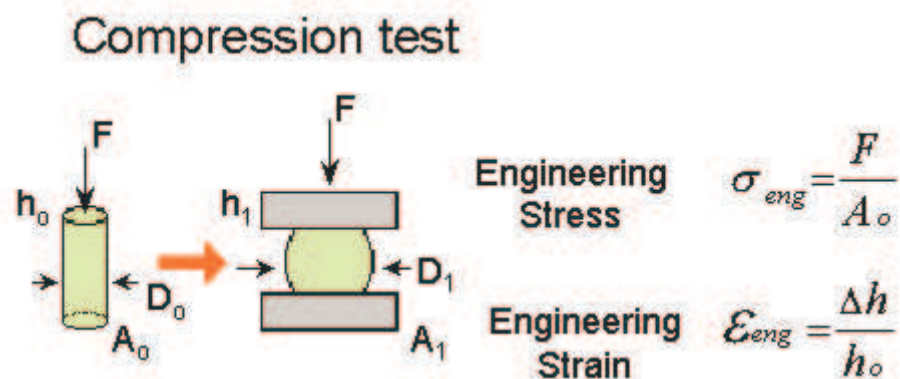


Figure 92: Engineering stress and strain for a compression test.

In Figure 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.

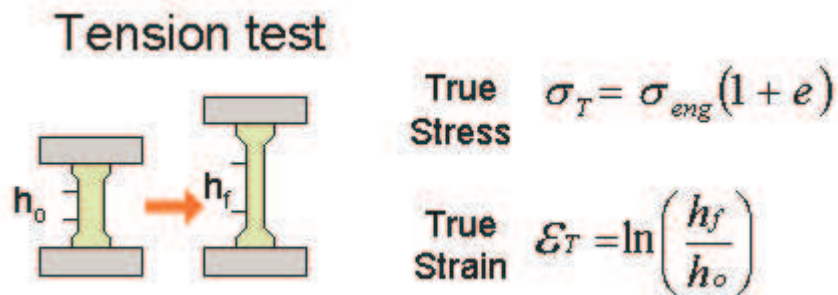


Figure 93: Stress and strain defined for tension test.

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. 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.

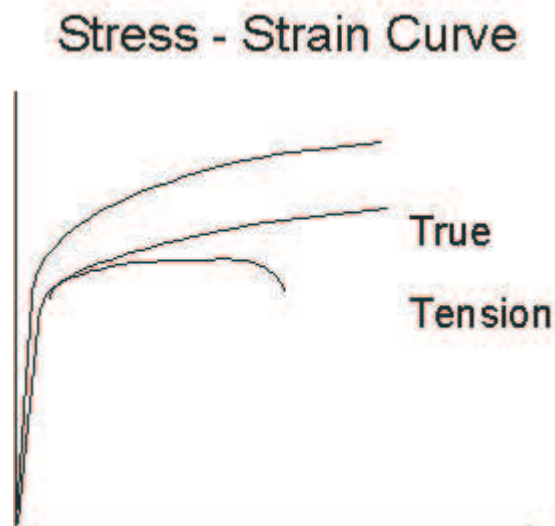


Figure 94: Stress-strain curve.

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 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.

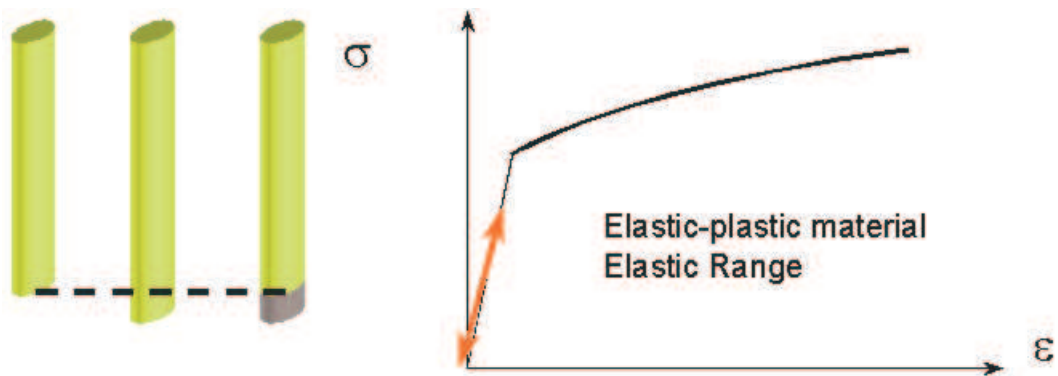


Figure 6: Diagram of elastic deformation.

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 7, a specimen under tension deforms first elastically and then plastically. The loading curve in figure 7 first 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.

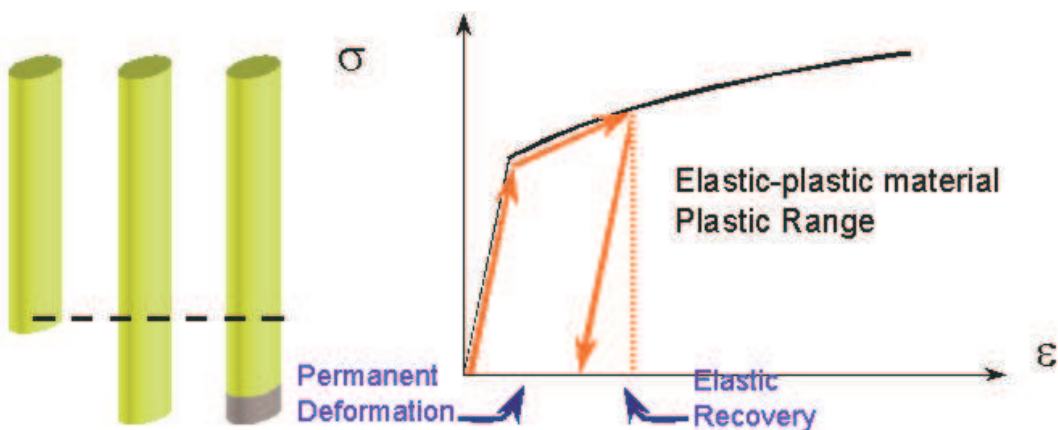


Figure 95: Diagram of plastic deformation.

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 8.

$$\text{Effective (von Mises) Stress } \bar{\sigma} = \frac{1}{\sqrt{2}} \sqrt{(\sigma_x - \sigma_y)^2 + (\sigma_y - \sigma_z)^2 + (\sigma_z - \sigma_x)^2 + 6(\tau_{xy}^2 + \tau_{yz}^2 + \tau_{zx}^2)}$$

$$\text{Effective Strain } \bar{\epsilon} = \frac{\sqrt{2}}{3} \sqrt{(\epsilon_1 - \epsilon_2)^2 + (\epsilon_2 - \epsilon_3)^2 + (\epsilon_3 - \epsilon_1)^2}$$

Figure 96: 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 9, the concept is shown visually.

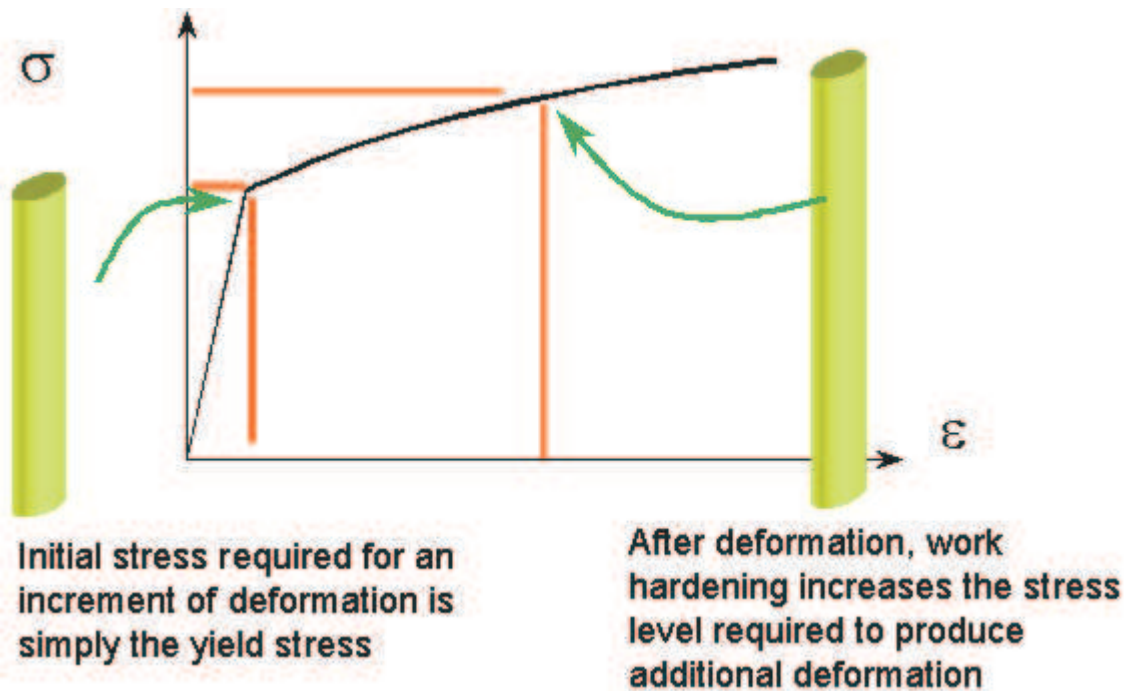


Figure 97: Introduction to flow stress concept.

Flow stress is strongly dependent on several state variables, among these are accumulated strain, instantaneous strain rate and current temperature. As seen in Figure 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.

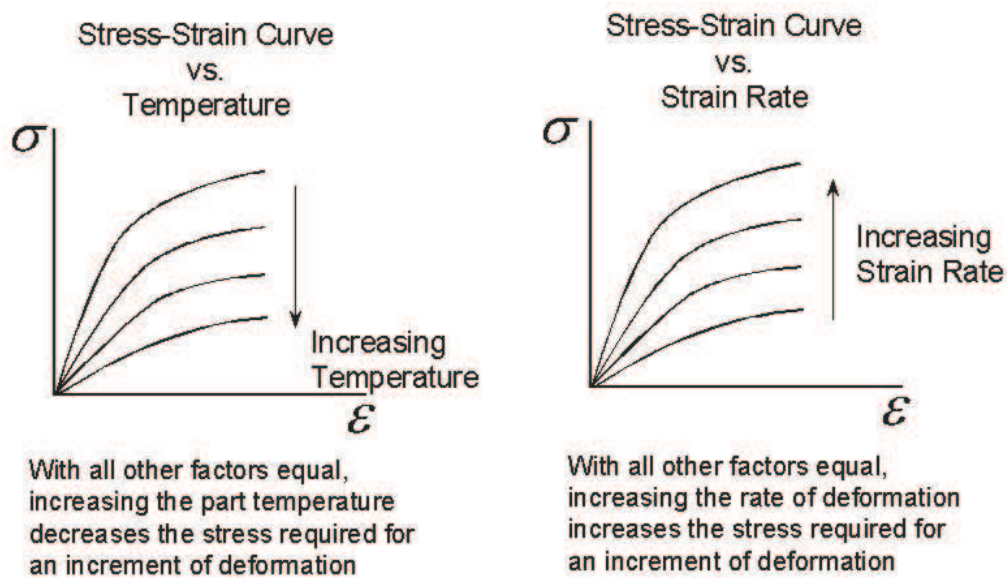


Figure 98: Variation of flow stress with strain rate and temperature.

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 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.

$$\sigma_{ij} = \frac{1}{\lambda} \dot{\epsilon}_{ij}$$

Figure 99: 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 12 where there are three different upset cases and the amount of friction between the workpiece 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.

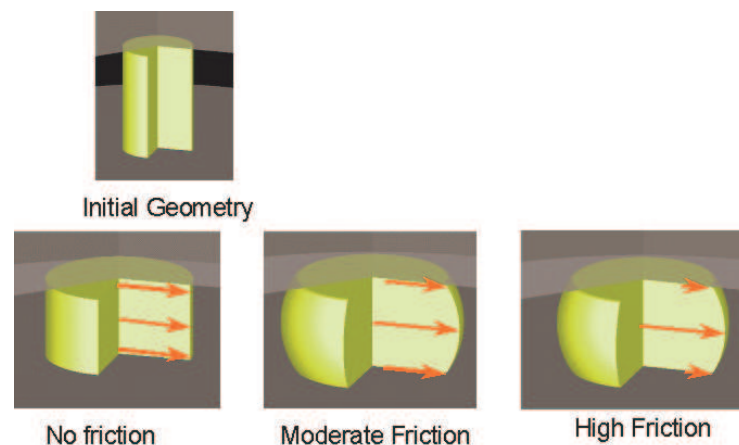


Figure 100: Minimum work rate principle. Note for each case, the actual velocity is the one which incurs the lowest work rate for the workpiece.

This minimum work rate principle can be expressed mathematically as the following functional form seen in Figure 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.

$$\pi = \int_V \bar{\sigma} \dot{\epsilon} dV - \int_S F_i u_i dS$$

$$\delta\pi = \int_V \bar{\sigma} \delta \dot{\epsilon} dV - \int_S F_i \delta u_i dS + K \int_V \dot{\epsilon}_V \delta \dot{\epsilon}_V dV = 0$$

plastic work *surface traction work* *change in volume (multiplied by a large "penalty" constant K)*

Figure 101: 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 14, an axisymmetric body is being upset between two flat dies. There is a grid that has been superimposed on the figure of the workpiece. 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 13 can 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 13 are the velocity at each node, which are shown as vector arrows the right side of Figure 14. In additions to the bottom equation of Figure 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 workpiece and the die. The boundary conditions 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.

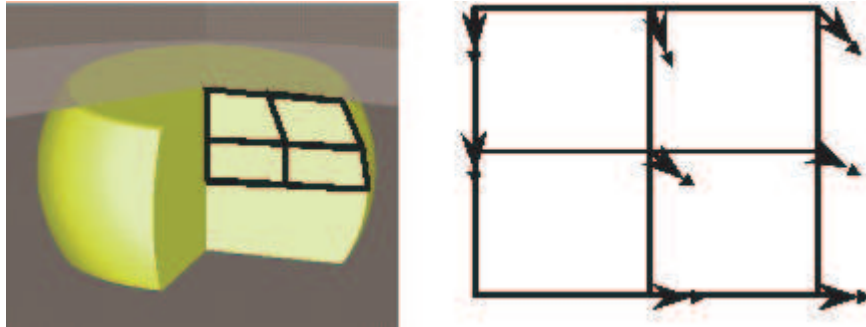


Figure 102: 2D mesh of upset test. (Note: This mesh is extremely coarse for the sake of clarity).

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 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.

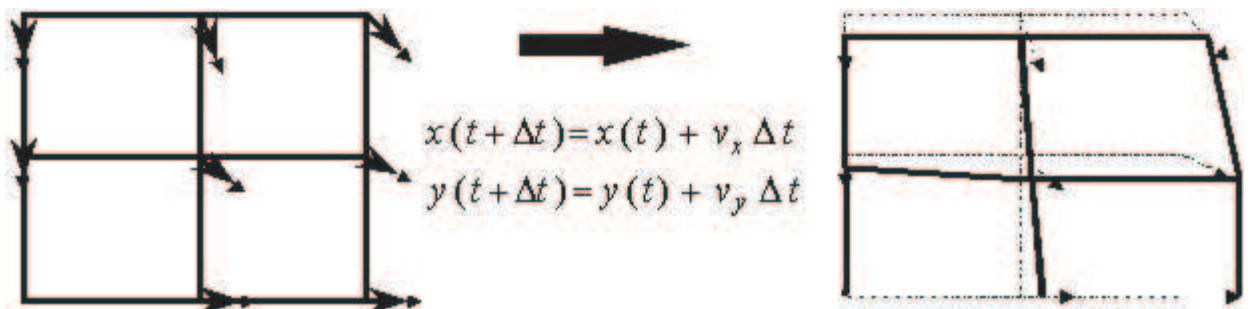
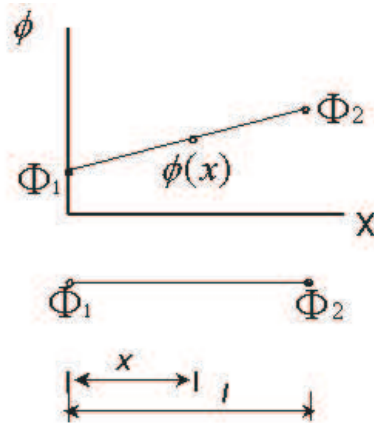


Figure 103: Updating nodal coordinates after a completed calculation.

The problem that now must be addressed is how to solve the equation in Figure 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 15 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 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:

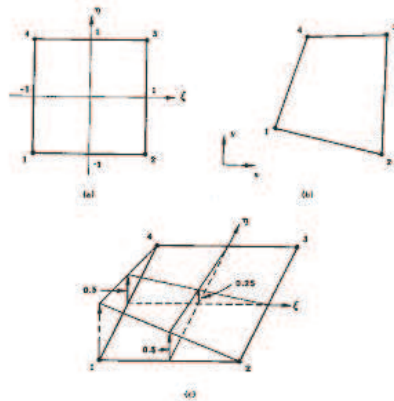
$$\phi = \sum N_i \Phi_i$$

Figure 104: Description of a shape function.

Figure 16 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 105: 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 17. 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.

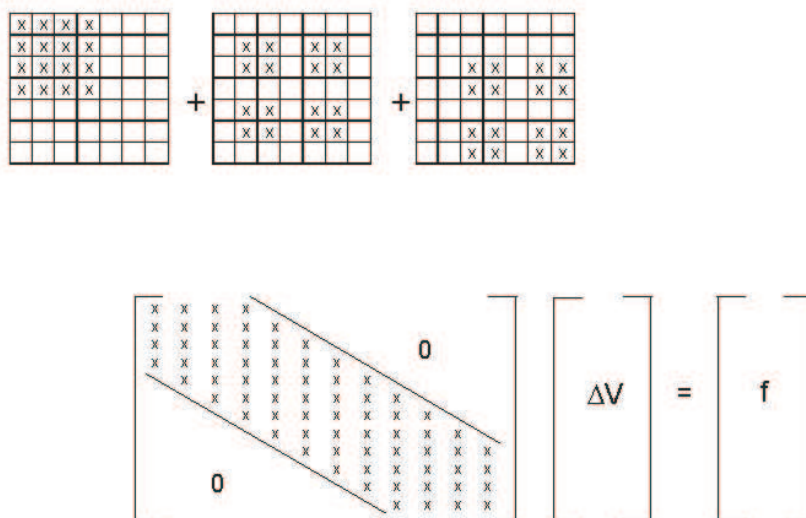


Figure 106: Construction of a stiffness matrix to solve for velocities.

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 Analysis*. 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 in the FEM (Finite Element Method) 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.

Prerequisites:

The user should have at least a rudimentary understanding of the FORTRAN language. This section is not meant to be an introduction into the FORTRAN language rather it is meant to explain how to build the user routines and how to implement them within DEFORM. For more information on the FORTRAN language, please refer to many books or the documentation that was included with your compiler.

Overview:

There are two different types of user routines available:

- User-Defined FEM Routines
- User-Defined Post-Processing Routines

The two different types of User Routines will be outlined below as to their purpose and how they are implemented. Later in this chapter, each one of these routines is explained in greater detail on how to implement each subroutine. To implement the user routines you must have a FORTRAN compiler installed on your system or you must use the DEFORM Support website (Windows only).

Note: The manner in which the User-Routines are compiled and linked is slightly different between Windows and UNIX/Linux platforms. These differences are outlined in detail later in this chapter.

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 provided object code to generate a custom built FEM engine. All the available subroutines are contained in a file named DEF_USR.FOR (UNIX) or def_usr.f (Windows). This file is a text format file containing all the available FORTRAN subroutines. To compile this file, run the script file DEF_INS.COM

(Unix) or follow the instructions below (Windows). At this point, the FORTRAN file will be compiled and linked to the object code named DEF_SIM.OBJ (UNIX) and DEF_SIM_LIB.lib (Windows). This will then generate a new FEM engine, named DEF_SIM.EXE. In the case of UNIX platforms, this whole process is shown in Figure 107.

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 workpiece. This can be performed using user-routines since these variables are available when the movement of the rigid die is calculated.

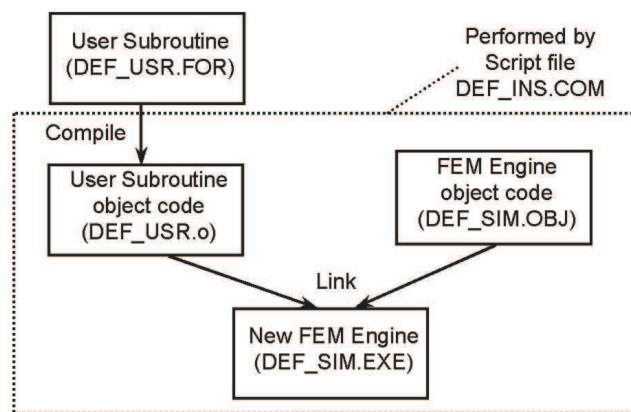


Figure 107: Description on how to compile/link a new FEM engine.

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. USRMTR

Description: This routine allows the user to calculate flow stress of a material. This routine is called at the beginning of each iteration.

2. 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.

3. **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.

4. **USRDMG**

Description: This routine allows the user to calculate damage of a material. This routine is called at once every step.

5. **USRZRT**

Description: This routine is added for convenience to complement USRRAT. This routine is called whenever INCUBT is called.

6. **USPM**

This subroutine allows the user to specify parameters for densification of a porous material model. This routine is called before stiffness matrix generation at the beginning of each iteration.

7. **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.

8. **USRBCC**

Description: This routine is used to allow the user to specify special boundary condition settings to a given object. This is called upon the beginning of each iteration.

9. **USRMAT**

Description: This routine is used to allow the user to specify a unique material property for a given object. This is called upon the beginning of each iteration.

9. **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.

10. **USRBCC2**

Description: This routine is used to allow the user to specify special boundary condition settings to a given object. This is called upon the beginning of each iteration.

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,

1. The file in the USR subdirectory of DEFORM-2D should be copied to a local directory and the file PSTUSR.FOR (UNIX) or pstusr.f (Windows) can be edited. The manner in which to edit this file is discussed in a later section of this document.
2. After editing, this file can be compiled and linked as a shared object file (UNIX/LINUX) or a dynamically-linked library (Windows).
3. 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.
4. 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.

The only difficulty is in editing the FORTRAN file. This requires a slight knowledge of the FORTRAN language and a compiler available on the computer one is working on. Please refer to Appendix N if using a Windows machine and a compiler is not available. *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-2D. The skeletal code for user routines is stored in DEF_USR.FOR or def_usr.f that has the FORTRAN functions that the FEM engine calls if a user routine is to be used. The user routines calculate 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 store data that can be used to specify parameters for the user-routines. The purpose for importing data for user routines through this method is that this data can be stored within a keyword file or a database and can be made unique to an individual simulation. This data can be accessed in the *Simulation Controls, Advanced Controls* menu. Ten lines are assigned for user defined data. Each

data should be separated by space and user can input as many of data in each lines unless they don't exceed 80 columns. In the user-routines the following code lets the user access the USRDEF values common block through the variable IUSRVL.

```
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
C READ(IUSRVL(LINE NUMBER),*) DATA1,DATA2,DATA3...
C
C TO WRITE DATA (10 RESERVED LINES)
C
C WRITE(IUSRVL(LINE NUMBER),*) NEWDATA1, NEWDATA2, NEWDATA3 ...
```

The screen where this information can be set is seen in the figure below.

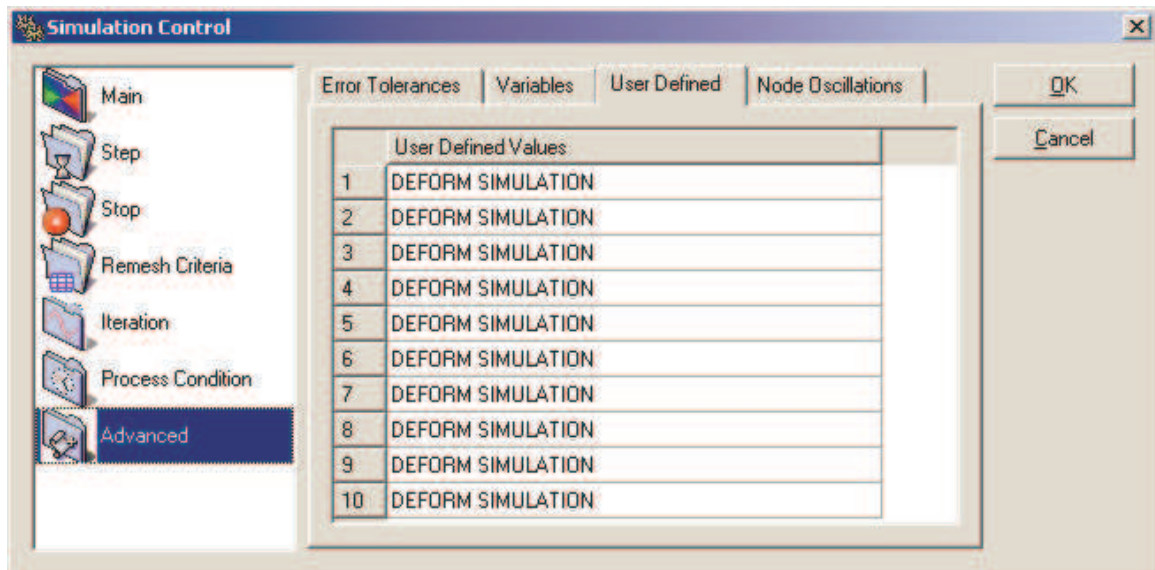


Figure 108: User-defined variables dialog.

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 :

1. The flowstress of the element at the current state condition.
2. The derivative of the flowstress with respect to total effective plastic strain.

3. The derivative of the flowstress with respect to total effective plastic 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 as *User routine* (as seen in Figure 109) and a routine number can be specified by selecting the icon to the right of the *User routine* line. This routine number (NPTRTN) is passed to the user defined flow stress subroutine to control branching to the specified UFLOW module.

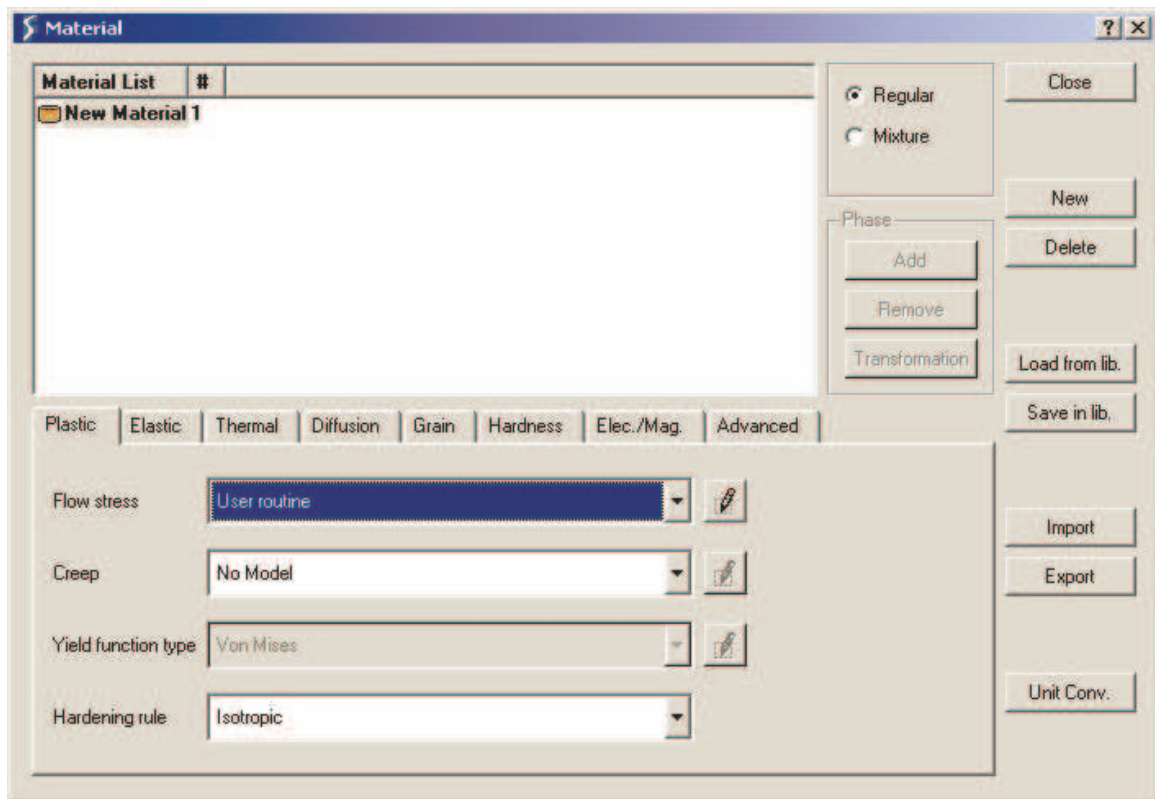


Figure 109: Specifying flowstress as a user-defined method.

Examples of using the user defined flow stress subroutine are given below. Note that only the additional code to the skeleton of the routines is shown in order to save space in the text (except for example 1). Recall that only three values need to be returned as mentioned above, YS (yield stress), FIP (derivative of flowstress as a function of strain rate) and YPS (derivative of flowstress as a function of strain):

1. The flow stress depends on the strain rate sensitivity index (PEM) and on the effective strain rate (EFEPS).

```
C*****
SUBROUTINE UFLOW1(YS,YPS,FIP,TEPS,EFEPS,TEMP)
```

```

C*****
C
C
C
C
C*****
C      IMPLICIT REAL*8 (A-H,O-Z), INTEGER*4 (I-N)
C
C ****  USER DEFINED VARIABLES ****
C
C      CHARACTER*80 IUSRVL
C      COMMON /IUSR/ IUSRVL(10)
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 ...
C
C ****  END  ****
C
C      DESCRIPTION
C
C      THIS ROUTINE IS USED TO DEMONSTRATE THE IMPLEMENTATION OF
C      MATERIAL ROUTINE. ALL THE REAL VARIABLES SHOULD BE DOUBLE
C      PRECISION. THE DEFINITION OF ARGUMENTS ARE DESCRIBED AS FOLLOWS:
C
C      INPUT :
C
C      TEPS  = EFFECTIVE STRAIN
C      EFEPS = EFFECTIVE STRAIN RATE
C      TEMP  = TEMPERATURE
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
C
C      COMMON /ELMCOM/ RZE(2,4),URZE(2,4),STSE(4),EPSE(4),EFEPSE,EFSTSE,
+      TEPSE,RDTYE,TEMPE(4),DTMPE(4),DAMAGE,

```

```

+      USRE1(1500),USRE2(1500),
+      USRNE(1500,4),NODEE(4),KELE,KELEL,KGROUP

```

C This code added for example 1

```

  PEM = 0.1
  YS = 10. * (EFEPS)**PEM
  FIP = 10. * PEM * (EFEPS)**(PEM-1.)

```

C

```

  RETURN
  END

```

Discussion:

In this very simple model, the flowstress is merely a function of strain rate. The three lines of added code can be seen using the comments in the code. This creates a very simple equation for the derivative with respect to strain rate and the derivative with respect to strain is zero. Every time a flowstress value for an element is required, this routine is called. It then computes the flowstress for the element and the derivatives and returns the values as PEM, YS and FIP.

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 that 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

```

User defined movement control (USRDSP)

DEFORM supports control of the die movement control for machines which cannot be controlled using the movement mechanisms given in the DEFORM system. The die movement can be a function of the following variables:

```

C INPUT :
C
C   NSPD = USER SUPPLIED ROUTINE NUMBER

```

C TIME = THE SIMULATED PROCESS TIME
C PDIS = PRIMARY DIE DISPLACEMENT
C VX = DIE SPEED IN X DIRECTION
C VY = DIE SPEED IN Y DIRECTION
C STRKX = THE CURRENT DIE STROKE IN X DIRECTION
C STRKY = THE CURRENT DIE STROKE IN Y DIRECTION
C FRZX = DIE FORCE IN X DIRECTION
C FRZY = DIE FORCE IN Y DIRECTION
C AVGSRT = AVERAGE STRAIN RATE
C SRTMX = MAXIMUM STRAIN RATE
C TMPMX = MAXIMUM TEMPERATURE
C DTIME = CURRENT TIME STEP (I/O)

The output that the user has to provide are:

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 = DESIRED TIME STEP (I/O)

It is important to note that either UPDV or UPDF need to be defined but both cannot be defined at the same instance. The reason for this is that defining the force determines the die speed through how fast the moving die can press the workpiece. Conversely, defining the die speed determines the amount of force required to push the workpiece a given distance at a given speed.

The die speed routines are functions which are called from the USRDSP subroutine based on the function number specified in the *Object, Movement controls* window.

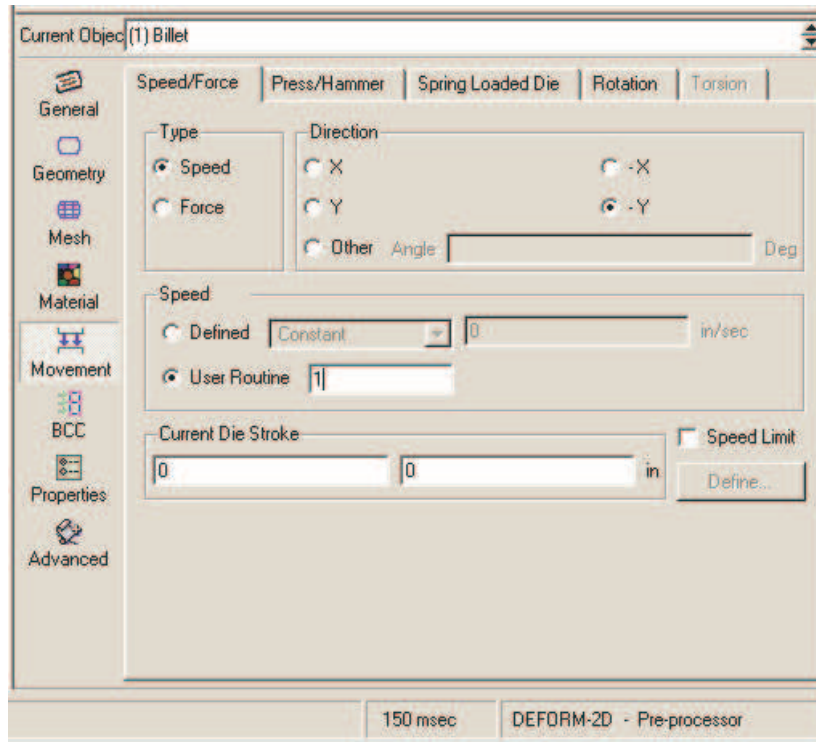


Figure 110: Specifying speed using a user routine.

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. The reason for this is that many different object can be deforming at different rates at a given step. This is the keyword PDIE(2) in *Simulation Controls, Advanced Controls* in the pre-processor.

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:

$$V = \dot{\epsilon} (H_{initial} - S)$$

where:

V = Output die velocity
 $\dot{\epsilon}$ = 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 workpiece. The total energy is an initial 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 translational 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 that can calculate 100 nodal and 100 elemental values 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 subroutine USRUPD is called to calculate the nodal and element values, which in turn calls USRSV1, USRSV2, etc based on the value of NPRTRN the material group number of the current element. If nodal values are being calculated the branching based on the material group number will not work if an object is composed of more than one material group.

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 :

Common block USRCTL
KOBJ : Object number
KSTEP : Step Number (N)
ISTATUS : 0 - the beginning of the step
 1 - the end of the step
Common block ELMCOM
RZE : Four corner coordinates
URZE : Velocity
STSE : Stress
EPSE : Strain rate
EFEPSE : effective strain rate
EFSTSE : Effective stress
TEPSE : Total effective strain
RDTYE : Density
TEMPE : Temperature
DAMAGE : Damage value
DTMPE : Temperature rate
USRE1 : Element user state variables (Input : At the beginning of Step N)
USRE2 : Element user state variables (Output: At the end of Step N)
USRNE : Nodal user state variables 1,2 at 4 nodes
NODEE : Connectivity
KELE : Global element number
KELEL : Local element number
KGROUP : Material group number
Common block NODCOM
RZN : Nodal point coordinates
URZN : Nodal point velocities
DRZN : Nodal point displacement
TEMPN : Nodal point temperature
USRN1 : User defined state variables (Input : At the beginning of Step N)
USRN2 : User defined state variables (Output: At the end of Step N)
KNODE : Node number

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 first user element value (USRE2(1)) and the second element variable is not used in this example.

```
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 approximately calculated and the result stored in the second user nodal variable (USRN2(2)). At first starting time is marked when TEMPN is between $AMAX_{TEMP}$ and $AMIN_{TEMP}$. When TEMPN is less than $AMIN_{TEMP}$, ending time is marked and elapsed time is calculated by difference of them. The cooling rate is calculated as the temperature difference divided by the elapsed time. 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.TEMP.NGT.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.

The damage routines are functions USRDM1 onwards with the inputs being as follows :

```

STS   = STRESS
EFSTS = EFFECTIVE STRESS
EFEPS = EFFECTIVE STRAIN RATE
DAMAG = PREVIOUS ACCUMULATED DAMAGE
STRLMT = STRAIN LIMIT
DTIME = TIME INCREMENT

```

The output is the new value of damage

```
DAMAG = NEW VALUE OF ACCUMULATED DAMAGE
```

The routine USRDM1 has an example on how the above routine is to be used.

1.

The default damage model in DEFORM is the normalized Cockroft and Latham model in which damage is computed as given below.

```

DNT = DSQRT((STS(1)-STS(2))**2 + 4. * STS(4)**2)
S1 = ((STS(1) + STS(2)) + DNT) * 0.5
S2 = ((STS(1) + STS(2)) - DNT) * 0.5
PRNSTS = S1
IF(PRNSTS .LT. S2) PRNSTS = S2
DAMAG = DAMAG + (PRNSTS*EFEPS*DTIME)

```

User defined material models (USRMAT)

The user defined material model(USRMAT) can be used to define a unique constitutive model. This will provide flexibility for models which require constitutive equations different than those provided by the default material

models. For example, in the case of molten glass or polymers, the constitutive models for rigid-plastic or elasto-plastic models are not sufficient in certain process conditions. This routine allows the user the flexibility to apply a constitutive model to capture instances where the built-in material models don't capture the correct behavior of the process. To select a user-defined material model, the user needs to select User under the Material properties window for a specific material group and then define a routine number for that material group. This routine number will determine which subroutine is called for that material group.

The inputs to the USRMAT subroutine are the stress components at the previous step, strain components increment, time increment and other elemental variables including deformation gradient and rotation tensor. The user may also use user-defined elemental values for this constitutive routine.

INPUT

DSTRAN(4) : Total strain increment (at $t=n+1/2$ and rotated by ROT1)
DFGR0(2,2) : Deformation gradient $F(t=n)$ w.r.t. $t=0$
DFGR1(2,2) : Deformation gradient $F(t=n+1)$ w.r.t. $t=0$
ROT1(2,2) : Rotation tensor at time $n+1/2$
STRESS(4) : Stress at the start of increment
EPX : Effective plastic strain at the start of increment
TOLDG : Temperature at the previous increment
TNEWG : Temperature at the current increment
TREF : Reference temperature

The user-material routine is responsible to compute the stress components and tangent modulus at the end of the time increment for the Newton-Raphson solution procedure. If the user changes the effective plastic strain as an internal variable, it can be updated using variable EPX.

STRESS(4) : Stress at the end increment
DSAVE(4,4) : Tangent modulus at the end of increment
EPX : Effective plastic strain at the end of increment

The routine USRMAT has an example on how the above routine is to be used.

1.

This example shows the user-material routine applied to a elastic constitutive model. Notice that tangent modulus, stress components and plastic strain were updated in this subroutine.

```

YM=1000
PV=0.3
THIRD=-1./3.
C
C      Elastic tangent modulus
C      S1=(1.+PV)*(1.0-2.0*PV)
S2=(1.-PV)/S1
S3=YM/(1.+PV)
S4=S3/2.
S5=S3/(1.-2.*PV)
S6=S5*PV
S7=S5-S6
C

```

```

C      Store elastic tangent modulus in DSAVE
C
DSAVE(1,1)=S7
DSAVE(1,2)=S6
DSAVE(1,3)=S6
DSAVE(2,1)=S6
DSAVE(2,2)=S7
DSAVE(2,3)=S6
DSAVE(4,4)=S4
DSAVE(3,1)=S6
DSAVE(3,2)=S6
DSAVE(3,3)=S7
DSAVE(4,1)=0.
DSAVE(1,4)=0.
DSAVE(4,2)=0.
DSAVE(2,4)=0.
DSAVE(4,3)=0.
DSAVE(3,4)=0.
C      C      Total stain increment
C      H1=DSTRAN(1)
H2=DSTRAN(2)
H3=DSTRAN(3)
H4=DSTRAN(4)
C
C      Calculate elastic stress
C      T1=STRESS(1)+S7*H1+S6*H2+S6*H3
T2=STRESS(2)+S6*H1+S7*H2+S6*H3
T3=STRESS(3)+S6*H1+S6*H2+S7*H3
T4=STRESS(4)+S4*H4
C
C      new stress at the end of increment
C      STRESS(1)=T1
STRESS(2)=T2
STRESS(3)=T3
STRESS(4)=T4
C
C      No effective plastic strain
C
EPX=0.0

```

User defined boundary conditions (USRBCC)

User defined boundary conditions (USRBCC) can be implemented to model special boundary conditions not available in standard DEFORM. *Currently, user-defined boundary conditions are only supported for the case where there is no contact and the edge is free.* The inputs to the routine are seen below which allow the user to make use of various quantities including nodal temperature and user-defined node variables. If any quantity is not provided in this group, the user can always compute it in user-defined nodal variables and import it into this subroutine.

Input :

- NODE1, NODE2 : edge node number
- IELEM : element number of current edge
- NBCD(2,2) : boundary condition code
- NDIE : contact condition
- 0 : non-contact
- n : contact to n object

RZ(2,2) : coordinate of nodes
 DRZ(2,2) : displacement of nodes USRNOD(NUSRN,I) : user defined node variables
 I : node number, user node1 and node2 to get the values
 NUSRN : number of user defined node variables
 ENVTEM : environmental temperature or film temperature
 TEMO(2) : previous step temperature
 TEMC(2) : current step temperature CURTIM : current time

As in the other user-defined routines, the user can write an arbitrary number of subroutines so long as the subroutines are defined and the call statements are defined. In each boundary condition window in DEFORM, the user can define a function number which allows the user to use a special subroutine for any type of boundary condition (the only exception is in the case where contact would exist). For example, if the user would want to write a routine to define the heat flux over a certain portion of the boundary, the user would select a portion of the boundary to have heat flux condition. The user would give the heat flux a function number and when the heat flux is computed for that section of the boundary, this routine would be called and the user would need to provide a heat flux value. As seen below, when the heat flux is computed, the subroutine number is found and when heat flux is computed (ITYPE=4), the user can compute heat flux. All the values below can be computed except for friction coefficient.

MODEL : subroutine number (function number)
 IMODE : simulation type
 1 : deformation
 2 : heat transfer
 3 : diffusion
 ITYPE : output type
 IMODE = 1
 1 : pressure
 2 : friction coefficient IMODE =2
 1 : environmental temperature
 2 : convection coefficient for free surface
 or lubricant coefficient for contact surface
 3 : radiation coefficient
 4 : heat flux
 IMODE = 3
 1 : environmental atom content
 2 : surface reaction coefficient
 3 : atom flux

Output :
 Variable : user defined boundary coefficient or value

The user needs to specify the function number as a negative value anytime a user-defined boundary condition routine is used. The output variable for each different combination of output type and mode corresponds to the list above. For example, if heat exchange with the environment is defined (IMODE=2, ITYPE=1), the user would need to supply environmental temperature as the output variable.

Windows building procedure for FEM routines

Requirements

ABSOFTE Pro Fortran 7.0 or above (version 7.0, 7.5 or 8.0)

Files provided by SFTC

(1) Libraries

DEF_SIM_USR.lib

(2) Project files

DEF_SIM_USR.gui (for Absoft 7.5 or above)
DEF_SIM_USR_Absoft70.gui (for Absoft 7.0)

(3) Sample fortran source file

def_usr.f

PROCEDURE

Compile def_usr.f and link this library with the DEF_SIM_USR_LIB.lib provided by SFTC. As result, an executable file DEF_SIM.exe will be produced. SFTC provides two project files and one FORTRAN file that can be used as templates.

Step-by-Step:

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 to the folder where DEFORM-2D is installed.

To build DEF_SIM.exe manually, follow these steps:

- (1) Double click DEF_SIM_USR.gui (DEF_SIM_USR_Absoft70.gui if you are using Absoft 7.0), 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 DEFORM2D/V8_0 directory (do not forget to make a backup copy of the original DEF_SIM.exe).

Note: An alternative to compiling user routines on a local compiler is to use the DEFORM webpage. This is available to all current DEFORM customers. Please refer to Appendix N for more details.

Windows building procedure for FEM routines

Requirements

ABSOF7 Pro Fortran 7.0 or above

Files provided by SFTC

pstusr2.f
PC_pstusr2.f
PC_pstusr2.als
PC_pstusr2.xps

Project file

USR_DEF_PST2.gui (USR_DEF_PST2_Abssoft70.gui for Abssoft
Version 7.0)


PROCEDURE

Compile USR_DEF_PST2.gui to generate a Dynamic-Link Library ---
USR_DEF_PST2.dll

Step-by-Step

If you can find Compile_DEF_PST_USR.bat in the current directory, you can compile the user routine by simply click on that batch file, and copy the USR_DEF_PST2.dll to DEFORM2D/8_0/Usr directory.

To generate USR_DEF_PST2.dll follow these steps

1. Double click USR_DEF_PST2.gui (USR_DEF_PST2_Abssoft70.gui if using Abssoft 7.0 compiler), Abssoft Pro Fortran compiler will open automatically.
2. Click on icon  or in the menu bar Click "ToolsàBuild", to build USR_DEF_PST2.dll.
3. After finishing with the set up of the project, customize pstusr2.f and rebuild USR_DEF_PST2.gui.
4. Copy USR_DEF_PST2.dll to the DEFORM2D/8_0/Usr directory.

Compiling user routines for UNIX platforms

The user routines are called for each node/element for every step during the simulation and efficient coding practices should be followed to minimize simulation time. Do not open or close data files during each subroutine call as this can degrade performance significantly. The USRDEF field in the pre-processor can be used to store a limited number of parameters. If you require more space than what is available in USRDEF then open the data files in the user subroutines and store a static variable in a COMMON block so that the file does not have to read each time the user subroutine is called.

After the FORTRAN code has been modified a new FEM engine can be build using the INSTALL2D script which is located in the \$DEFORM_DIR or using the script build_fem which is in the \$DEFORM_DIR/USR directory. If the INSTALL2D program is being used, the code in \$DEFORM_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 \$DEFORM_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.

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

```
$DEFORM_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 \$DEFORM_DIR/CONFIG.COM and a line in the .cshrc after the source \$DEFORM_DIR/CONFIG.COM redefining the alias should work. In the .cshrc file the following modifications can be made.

```

setenv DEFORM_DIR '/disk1/deform/2d/v60'
source $DEFORM_DIR/CONFIG.COM
#
# Old alias
#
alias DEF_ARM $DEFORM_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.

```
$DEFORM_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 will be located with the current installation of DEFORM-2D usually in a directory such as C:\DEFORM2D\V8_1. If it is not there, look for a file named DEF_SIM.EXE.

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 customers of DEFORM:

<http://support.deform.com/2d/support/fortran/>

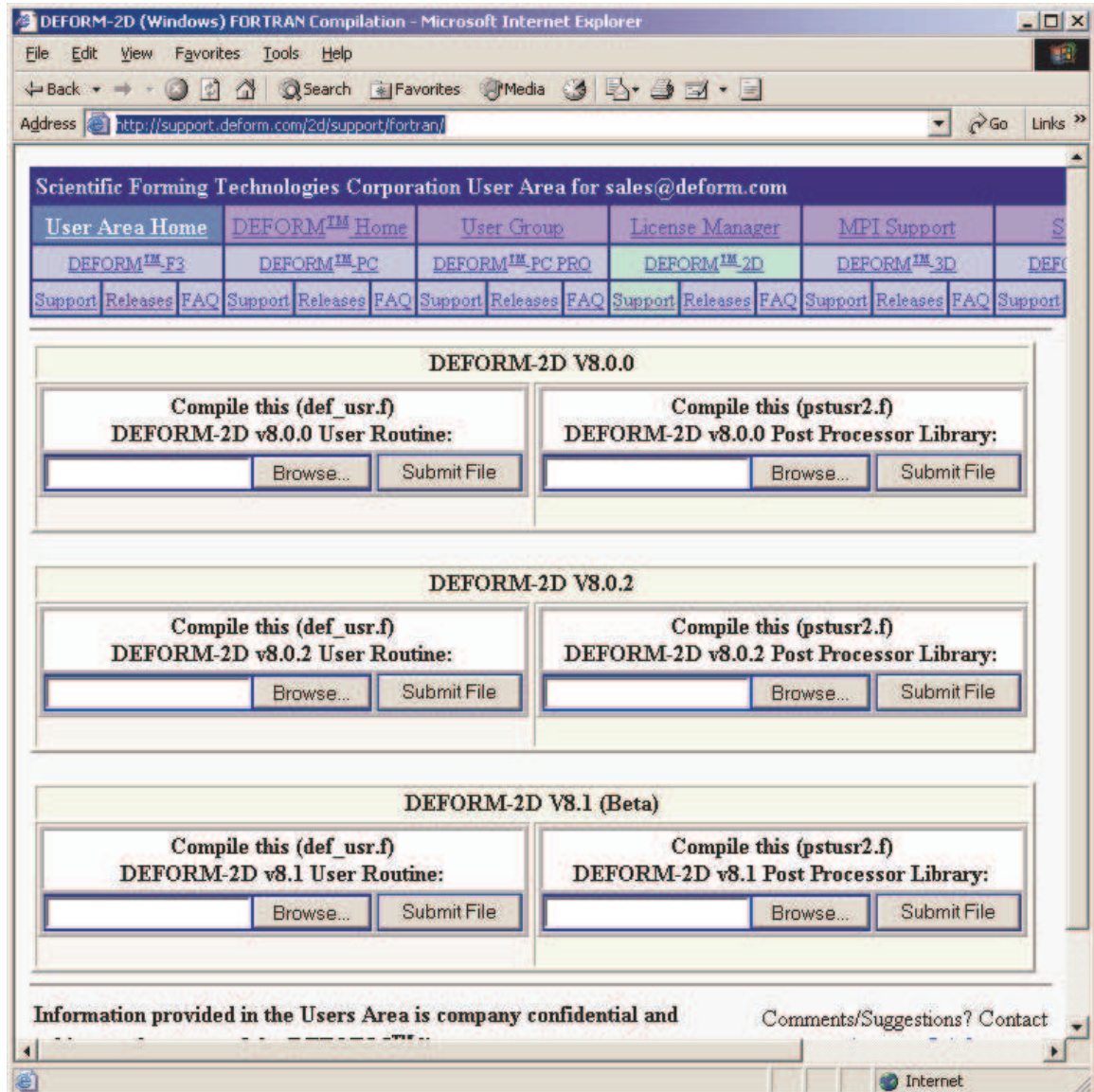


Figure 111: The DEFORM webpage for compiling user routines.

Figure 129 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 are 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 which is stored in the \$DEFORM_DIR/USR directory. 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 linked to the TEXT and GUI post-processors using the INSTALL2D program or running the script build_fem in the \$DEFORM_DIR/USR directory. The script compiles the program PSTUSR.FOR and also links this with the TEXT post-processor and 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. 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 "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 then 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 :

```
TNOW      : CURRENT Time
RZ        : Element center Coord. (Dimension : 2 )
TEMP      : Temperature
EFEPS     : Effective Strain Rate
TEPS      : Total accumulated Strain
EFSTS     : Effective Stress
DAMG      : Damge Factor
RDTY      : Relative Density
STS       : Stress Tensor (Dimension : 4 )
EPS       : Strain Rate Tensor (Dimension : 4 )
TSR       : Strain tensor
VAR1(1-20) : Initial State Variables
IOBJ      : Object number
NUMNP     : Total number of nodes
NUMEL     : Total number of elements
ICURNE    : Current node/element (when INIT=2)
NOUTINE   : Routine to be used USRPS1 .. USRPS20
```

The output from the functions to be given by the user :

```
VAR2(1-20) : Updated State Variables
```

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 user routines

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 INSTALL2D program which is located in the \$DEFORM_DIR or using the script build_pst which is in the \$DEFORM_DIR/USR directory. If the INSTALL2D program is being used, the code in the \$DEFORM_DIR will be altered. If you wish to make a local copy of the code then copy all files from the \$DEFORM_DIR/USR directory to your local directory. Then run the script build_pst using the following command:

```
> build_pst
```


This builds a new copy of the post-processor DEF_PST.EXE in the local directory and also builds DEF_PST2.SL, 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 text-based and 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

The GUI post-processor looks for the shared library DEF_PST2.SL first in the current working directory then in the users HOME directory under the subdirectory DEFORM2 and then in the DEFROM_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.

Release Notes for Version 8.1

Overview:

These release notes are organized as follows:

- Overview of available interfaces
- Additional capabilities
- Improvements in the interface

If there are any questions about the information contained in this document, please contact Scientific Forming Technologies Corporation. Please review this document, as it is a general overview of the improvements and exciting new features contained within this new release.

Overview of available interfaces

There are many new interfaces for the DEFORM system. A description of them is as follows:

1. The standard DEFORM-2D interface of Pre-Processor, Post-Processor and the Main menu.
2. DEFORM-F2: An open, hybrid interface to allow the user to either follow an ordered list to create simulations or to skip through the settings. This interface is extremely streamlined to construct a large variety of simulations in a very brief time.
3. DEFORM-MO2: An interface to allow the user to construct many successive operations at the initial setup.
4. Heat-treatment interface: This allows the user to construct many successive heat-treatment simulations with minimal effort.
5. Inverse heat transfer interface: This allows the to determine the heat transfer coefficient at the boundary of an object by using actual thermocouple data coupled with iterative simulation results.
6. Optimization interface: This allows the user to perform optimization simulations of either preform shape or strain distribution.
7. Machining interface: This allows the user to perform two-dimensional cutting simulations operations with ease.

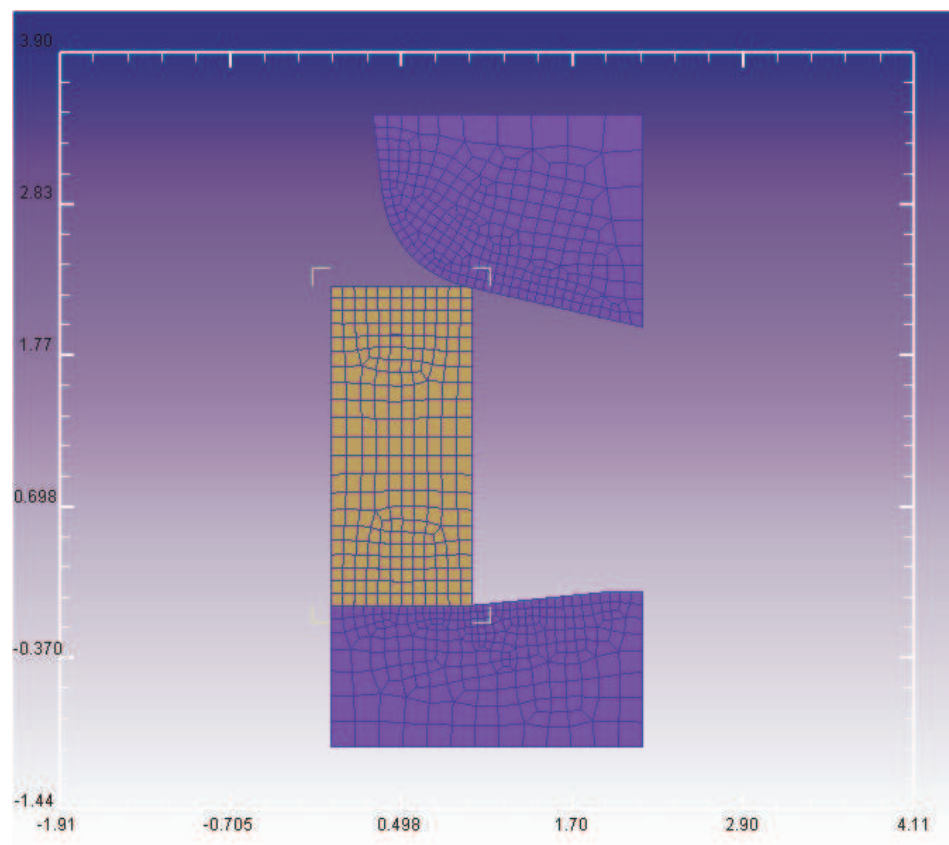
Additional capabilities

1. Simulation graphics are now available through the interface.
2. There is now a three-dimensional viewer for post-processing that can be utilized for all DEFORM-TOOLS users.

Improvements in the interface

Geometry and Display

1. A new geometry-editing dialog with many new options has been added.
2. Many points can be selected at once by clicking and dragging on the displayed points in the geometry window.
3. A scale has been added to the display window as seen in the figure below.



4. A centerline has been added to the display window for axisymmetric simulations.
5. Geometry now has a shading near the border in order to show the inside versus the outside. Shading displayed on the inside of the geometry indicates correct (counterclockwise) orientation for closed bodies.

6. There are now primitive geometry definitions available in the geometry definition.
7. Geometry can now be created through a series of boolean operations.
8. Fixed the problem in transformation data initialization/display.
9. Improved the picking in element and edge initialization.

Object Positioning:

10. When positioning objects, the window definitions (such as mesh density windows) will be positioned accordingly with the body.
11. Multiple objects can be coupled together when positioning. This will allow the user to position many objects simultaneously.
12. Mouse-driven positioning is available. Objects can be dragged using the mouse.

Object Movement:

13. Spring-loaded dies are now available.
14. The screw press model now has a template for multiple blows.
15. The problem of not showing asterisk for the picked axis in object movement has been fixed.

Object Meshing:

16. Mesh density windows can now be reordered manually.
17. When saving point tracking data directly to file, the format is now all on a single line in order to be excel-friendly.
18. Absolute mesh density is available now by setting the number of elements to zero.
19. Thickness elements are now saved to keyword and database files.
20. DEFORM-2D can now read and border extract meshes with degenerate elements (triangular).
21. Picking nodal and element information after interpolation of state variables now works correctly.

Object Boundary Conditions

22. Edge BCC now works correctly in the case of multiple mesh boundaries.

Material Properties

23. The pasting of numbers to the flow stress table has been improved to have higher precision.

24. The phase transformation kinetics dialog has been consolidated into the phase transformation dialog.

Inter-Object Properties

25. Fixed problem of failing to delete self-contact relation in inter-object dialog.

26. The friction user routine number of inter-object is now saved into the database.

27. Heat transfer coefficient as a user routine is now available.

28. Inter-object relations for one non-rigid object is now allowed.

29. Friction window re-ordering is now allowed.

Database Generation

30. The warning for the 0.005 force convergence for elasto-plastic simulations has been removed.

31. When defining either distance per step or time per step, the other is now set to zero.

Simulation

32. Simulations can now be submitted to a remote license server.

33. Floating licenses can be monitored remotely now.

34. Fixed bug of user lost MPI processor information without clicking enter or tab on table in environment dialog.

35. Spaces in the directory name are now legal for the batch queue.

Post-Processing

36. Unit conversion is now available in the post-processor.

37. Graph smoothing is now available.

38. Shear angle can now be plotted from the Simulation Summary window.

39. Point tracking and flow-net have more options for their display.

40. Micro-structural variables and user variable can now be point tracked.

41. The flow-net capability has been improved to handle concave shapes.

42. Added 2D torque, angular velocity, and angle rotated to load stroke graphing.

43. Add 6 more levels to user defined color editor

44. Stroke can now be placed on the y-axis for load/stroke curves.

45. New viewport options have been added.

46. Increase step combobox pull down list length to display larger number of entries.

47. Implemented vector plot for principal stress/strain/strain rate

- 48. Line contour - show geometry for object without mesh
- 49. User specified range for state variables is stored independently for each variable.

Miscellaneous:

- 50. Multiple resolutions now available for printing

Additions and Enhancements to the Simulation Engine

- 51. Improved phase transformation kinetics models
 - a. Allow simultaneous modeling of competing A->B, A->C phase transformations. The model utilizes an Avrami formulation augmented with physics-based factors such as interfacial area between mother and child phases.
 - b. Equilibrium volume fraction. This can be used to compute volume fraction of precipitation, as well as transformation like Austenite -> Ferrite, where equilibrium volume fraction depends on carbon content.
 - c. State variables are simplified. Incubation time (TICF), starting volume fraction (VOLFN) not used anymore.

Additions and Enhancements to the Inverse Heat Transfer Template

- 52. Diffusion is added.
- 53. Symmetrical plane definitions.
- 54. Allow heat treatment medium temperature specified as function of time.
- 55. Allow to deactivate certain transformation for particular operations.

Additions and Enhancements to the Inverse Heat Transfer Template

- 56. Allow one heat transfer zone to contain two or more segments on the boundary.
- 57. On-line checking of currently best heat transfer coefficients.
- 58. Convenient restart using results from previous optimization.
- 59. Final simulated thermal history is compared with experimental measurements.

System Installation: UNIX and Linux

Warning: In this release, we have changed the name of the password file from DEFORM.IDF to DEFORM.PWD in order to be consistent with the PC version. If you install the password file as DEFORM.IDF, the installation will not work!!!

OVERVIEW

In order to load the DEFORM-2D system on your machine, you need the version of the operating system given in Table B.1 and a FORTRAN compiler if you need to use user routines (optional).

Approximately 100-120 MB of disk space will be required for this installation. In this release each platform's installation requires a platform independent file DEF2DV80.TAR.Z and a platform specific file (Table 1). The installation script INSTALL located on the CD-ROM will install the required files on your system. The platform independent file contains the manuals, labs, examples and other other utilities. The platform specific file contains the DEFORM executables and Python.

HP USERS*

The two HP tar files are compiled for the same operating system (HP-UX) but are for different operating system releases.

Computer	Operating System	File Name
HP (IA64)	HP-UX 11.22	2DV80_HP-UX_B.11.22.TAR.Z
HP (PA-RSIC)	HP-UX 10.20	2DV80_HP-UX_B.10.20.TAR.Z
HP (PA-RSIC)	HP-UX 11.00	2DV80_HP-UX_B.11.00.TAR.Z
HP Tru64 (Alpha)	OSF1 V4.0 F+	2DV80_OSF1_V4.0.TAR.Z
IBM (PPC)	AIX 4.3+	2DV80_AIX_3.TAR.Z
Linux (x86)	Red Hat 7.2	2DV80_LINUX_2.4.20-20.7.TAR.Z
SGI (MIPS)	IRIX64 6.5+	2DV80_IRIX64_6.5.TAR.Z

Sun (Sparc)	Solaris 2.6+	2DV80_SUNOS_5.6.TAR.Z
-------------	--------------	-----------------------

Table 1: Files for different platforms.

Installing DEFORM-2D

The first step in the installation is to mount the installation CD-ROM. Mounting a CD-ROM makes the files from the CD available for use in the file structure of the system. On some systems (SGI, Sun...) this is accomplished automatically by an auto mount daemon. On other systems this task must be performed manually. In most cases administrative privileges (root) are required to mount a disk.

Note: The CD must be mounted in a manner such that long filenames are available, otherwise the installation script will not work properly. The DEFORM-2D installation disk is created using the ISO9660 format with both Rock Ridge and Joliet extensions.

Mounting the CD

Note: With few exceptions, these steps must be performed as root. The devices and mount points used in the examples may vary from system to system. Consult a System Administrator for details about a specific system.

If the directory where the CD is to be mounted does not exist create it. A typical mount point for a CD is /cdrom. This directory can be created using the command:

```
# mkdir /cdrom
```

The CD directory will be referred to as /cdrom in this installation procedure. If it is different, substitute the site dependent directory name.

Load the release CD-ROM in the CD-ROM drive and mount the CD-ROM using the following commands (to use this you must be logged in as root).

Platform Specific Mounting Instructions

Compaq TRU64

```
% mount -r /dev/device /cdrom
```

or

```
% mount -t cdfs /dev/device /cdrom
```

as needed to mount the CD with long filename support.

HP

To properly mount a DEFORM CD on HP-UX the following procedure must be followed:

This is a summary of instructions detailed at <http://www.faqs.org/faqs/hp/hpux-faq/> Section 5.3.8.

Create (or alter) /etc/pfs_fstab to include a line as follows:

```
/dev/dsk/c0t0d0 /CDROM pfs-rrip xlat=rrip 0 0
```

Where /dev/dsk/c0t0d0 is the device and /CDROM is an existing directory where you will be mounting the device.

Start the pfs daemons with the following commands:

```
# nohup pfs_mountd &
```

```
# nohup pfsd 4 &
```

Mount the drive:

```
# pfs_mount /dev/dsk/c0t0d0 /CDROM
```

Note: You must use pfs_umount to unmount a disk mounted with pfs_mount.

IBM

mount -v 'cdrfs' -r /dev/device /cdrom

SGI

IRIX systems automatically mount the CD-ROM after it is inserted in the drive

SUN

Sun Solaris systems mount the CD-ROM after it is inserted in the CD-ROM drive

Starting the INSTALL Script

Note: The installation is generally performed as an ordinary user, not as root. Once the CD-ROM is mounted, DEFORM may be installed using the following commands:

Change to the CDROM directory:

```
% cd /cdrom
```

Run the INSTALL program:

```
% ./INSTALL
```

The INSTALL program will check the computer for a supported operating system. If the platform is supported the script will prompt for a directory where DEFORM will be installed. If the installation directory does not exist, a directory will be created. The appropriate platform dependent and platform independent files will be installed from the CD to the installation directory. An example of a sample session is given below :

```
-----  
SCIENTIFIC FORMING TECHNOLOGIES CORPORATION
```

```
DEFORM-2D V8.0 INSTALL SCRIPT
```

```
Support Telephone : 614 - 451 - 8313
```

```
Fax: 614 - 451 - 8325
```

```
E Mail: support@deform.com
```

```
Web site: http://www.deform.com
```

```
NOTE : If at any time you want to stop the installation  
process, press Ctrl-C to abort the process
```

```
-----  
...Current directory = /home/testinst
```

```
...DEFORM_DIR = /home/testinst/2D/V80
```

```
...Operating system name = GenericUnixOS
```

```
...Release version = 17
```

```
Install default configuration, DEFORM-2D 8.0 for GenericUnixOS? [y]
```

If the installation script does not work DEFORM can be installed by performing the tasks:

1. Create the installation directory:

```
mkdir /usr/deform/2D/V80
```

2. Switch to the installation directory:
`cd /usr/deform/2D/V80`
3. Uncompress and untar the installation files from the CD:
`zcat /cdrom/2DV80_GenericUnixOS.TAR.Z | tar xvf -`
`zcat /cdrom/DEF2DV80.TAR.Z | tar xvf -`

Installing the password file

The password for running DEFORM-2D on the current system has to be setup in the file DEFORM.PWD. This password file can be obtained from your distributor.

Installing DEFORM-2D components (Optional)

1. Go to the DEFORM-2D directory.
`% cd /usr/deform/2d/v80`
2. **Optional**
 Modify the command procedure file, DEF_MSH.COM which controls the activation of other FEM pre- or post- processors, if necessary.
 In DEFORM-2D System, several commercially available FEM pre- and post-processors (i.e. PATRAN, I-DEAS) can be directly activated. This is done by DEF_MSH.COM. Since the activation of these processors is site-dependent, modifications may be necessary. Please check with your system administrator for the procedure to activate these processors and follow instructions given in DEF_MSH.COM to make necessary changes.
Note : The GUI version does not use the above features
3. **Optional**
 Do not copy DEF_USR.FOR from an old version of DEFORM. Update DEF_USR.FOR provided with this version. Changes are documented in the file DEF_USR.FOR.
 Modify the FORTRAN source, DEF_USR.FOR which contains user defined material properties routines and die velocity routines, if necessary. The file is in the USR directory.
 10 dummy material property routines, from UFLOW1 - UFLOW10, are included in DEF_USR.FOR. To replace these routines, refer to subroutine UFLOW1 for instructions.
 10 dummy die speed routines, from DIESP1 - DIESP10, are included in DEF_USR.FOR. To replace these routines, refer to subroutine.
 The user defined variable subroutine USRUPD may be used to specify the USRNOD and USRELM variable values. As an example USRELM(1) stores the maximum principal stress. Instructions on what is to be done is given inside this subroutine.
 A user defined damage criteria also exists in the DEF_USR.FOR. To use these subroutines refer to the routines.
4. **Optional**
 Do not copy PSTUSR.FOR from an older version of DEFORM. Update PSTUSR.FOR provided with this version. Changes are documented in the file PSTUSR.FOR.
 Modify the Fortran source, PSTUSR.FOR which contains user defined

routines for state variables tracking, if necessary. The file is in the USR directory.

10 dummy state variable routines, from USRPS1 - USRPS10, are included in PSTUSR.FOR. To replace these routines, refer to subroutine USRPS1 for instructions.

5. If no change is made to DEF_USR.FOR goto step 9.
6. To link DEF_USR.FOR with the DEFORM simulation engine, run
INSTALL2D.
% INSTALL2D

Select Option 3

The updated FEM simulation execution image "DEF_SIM.EXE" will be generated in the DEFORM-2D EXE directory.

7. If no change is made to PSTUSR.FOR goto step 9.
8. To link PSTUSR.FOR with the DEFORM-2D post processor, run INSTALL2D.
% INSTALL2D

Select Option 4

The updated post processor execution image "DEF_PST.EXE" will be generated in the DEFORM-2D EXE directory. For user variable tracking in the GUI post processor a shared library "DEF_PST2.SL" will be generated in the DEFORM-2D USR directory.

9. The installation of the optional modules is complete. To run the program you must create a valid password file in the current directory. See section for more information about the password file.

Installing Java for the 2D cutting template (HP-UX 11.0 only)

1. Before install java environment, go to website:

<http://www.hp.com/products1/unix/java/patches/index.html>

This webpage provides information on how to detect an existing java environment on each HP system, and which patch to be applied before install new java.

Cutting template recommends using Java 1.3. If your system already has java installed, check the version by typing:

```
java -version
```

Since java 1.4 may not be compatible with java 1.3. If you already has java 1.4 installed and have difficulty running template. Please try to install java 1.3

2. To install java 1.3

a). Go to the website :

http://www.hp.com/products1/unix/java/java2/sdkrte1_3/downloads/license_rte_1-3-1-10_pa-risc.html

b). Read the terms, check "ACCEPT ALL OF THE ABOVE TERMS."

Fill in the form. After this, click "submit" button, you can see the download page.

c). Click "download hp-ux runtime environment, for the Java 2 platform version 1.3.1.10 PA-RISC " to download installation file to your hard disk

d). Follow the instruction shown to install

3. Check PATH by typing: env

If it doesn't include java path, add it by:

```
setenv PATH (path to java, ie: /opt/jre1.3/bin) $PATH
```

----Cutting template configuration

1. After installed cutting template, go to 2d_cutting directory

2. cd CONFIG, modify "directory.ini" file, it requires 2 values for DEF_DIR, and USR_PROB_DIR

DEF_DIR: the absolute path of 2d_cutting.

ie: /home/deform/2d/v80/TOOLS/2d_cutting

USR_PROB_DIR: the absolute path of PROBLEM directory of user (usually under your account)

ie: /home/myAccount/PROBLEM

3. cd ..

4. type:

```
Machining_Template.bat
```

to run template

Setup a user account

1. Make sure the default login shell is "C" shell (csh). We will assume the

DEFORM-2D user name is "user1" and the login directory is "/disk1/user1".

2. Log into the user account and edit 'C' shell login command file, ".cshrc", to

include the following two command lines:

```
setenv DEFORM_DIR '/usr/deform/2d/v80'  
source $DEFORM_DIR/CONFIG.COM
```

3. The first command line defines the global symbol "DEFORM_DIR" to be the DEFORM-2D directory. The second line executes the DEFORM configuration file "CONFIG.COM".

Note : The user can use any shell (csh,bsh,tcsh...) but the "C" shell file must have the above lines. Also the settings should be active for all shells, not just interactive shells.

4. You may log out the system at this time. Next time you log into the system, the above command lines will be executed automatically.
5. If you decide to stay in the system at this time, you need to execute ".cshrc" manually since its contents have changed. To do that, enter

```
% source .cshrc
```

6. To verify the definition of DEFORM_DIR, you may enter "printenv" or "setenv" from the keyboard. The system will display the definition of DEFORM_DIR and

```
DEFORM_USER:  
DEFORM_DIR=/usr/deform/2d/v80  
DEFORM_USER=/disk1/user1
```

System execution

The following procedure may be used to run the DEFORM-2D system :

1. Log into the DEFORM-2D user account from an X-terminal
2. Invoke the DEFORM-2D System by issuing:
% DEFORM (Text based interface)
% DEFORM2 (Graphical based system)
3. Program modules in DEFORM-2D System can also be invoked independently by issuing:

```
% DEF_AMG ( Automatic Mesh Generation Module )  
% DEF_PRE ( Input Preparation Module )  
% DEF_SIM ( DEFORM-2D Simulation )  
% DEF_PST ( Simulation Results Post-processor )  
% DEF_MO2 ( Multiple Operation system )  
% DEF_BQ2 ( Simulation Queue )
```

Installation related problems

Simulation aborted by the user

Simulation stops after one step with the message SIMULATION ABORTED BY USER.

If the simulation status file protection is not set properly then the message SIMULATION ABORTED BY USER will appear when trying to run a simulation. If the status file is corrupted for any reason, a new status file can be generated using the INSTALL2D script:

1. Go to the DEFORM-2D directory

```
% cd /usr/deform/2d/v80
```

2. Run the script INSTALL2D and select the option 1 to create a new status file
% INSTALL2D

Invalid program copy

In order to run DEFORM on your computer a valid password file for the computer is required. This file ensures that DEFORM will work only on a single computer between the dates in the license agreement. If DEFORM is installed on a computer for which a valid password is not present then DEFORM will not work. The license file is stored in the DEFORM_DIR directory in the file DEFORM.IDF. This file is created when the system is installed or when a new password is entered. The file has the company name and also one line for each product that is licensed.

Possible reasons for getting an INVALID PROGRAM COPY message :

- ◆ The license has expired.
- ◆ The version of DEFORM that is installed is different from the one that is licensed.
- ◆ The computer on which DEFORM is installed is different from the one that DEFORM is licensed on.
- ◆ The environment variable DEFORM_DIR does not point to the proper directory.
- ◆ DEFORM.IDF exists but there is invalid data in this file.
- ◆ When copying this file from a PC the Ctrl-M characters were appended at the end of each line.
- ◆ Some of the above situations will not arise if DEFORM was working and suddenly stopped working. In most situations the license has probably expired and a new license has to be obtained from Scientific Forming Tech Corp.

Mesh generator does not work

One of the problems if DEFORM-2D is not installed properly is that in the GUI the mesh generator will not work and in the text based main program, the pre-

processor, and post-processor cannot be invoked. To confirm that this is the problem try to mesh a very simple geometry. In the terminal the following message will be displayed when trying to generate a mesh.

```
DEF_AMG : command not found.
```

On some systems the above message may not be printed but a dialog box with the following message can appear

```
MESH GENERATION ERROR
```

```
The KEY file generated by the mesh generator
could not be found. This implies mesh generation
failed at some state. Possible reasons could be
```

1. Incorrect geometry, check orientation.
2. Insufficient resources to run AMG
3. Insufficient memory array sizes.

This problem can occur if the environment variable DEFORM_DIR is not set in the proper place in the .cshrc file. The variable must be defined for interactive and non-interactive shells. If it is not defined for non-interactive shells, then the variable DEFORM_DIR is not defined when the mesh generator is executed. The shell returns with the message that the command is not found. Also it is very important that the .cshrc under the users HOME directory must have these definitions. If /etc/cshrc has the definitions then it may not work.

An example of a .cshrc file taken from an HP is given below with the incorrect definition of the DEFORM environment variables.

```
#
# Default user .cshrc file (/bin/csh initialization).
#
# Usage: Copy this file to a user's home directory and edit it to
# customize it to taste. It is run by csh each time it starts up.
#
# Set up default command search path:
#
# (For security, this default is a minimal set.)
#
set path=( . /usr/local/bin /usr/local/lib )
#
# skip remaining setup if not an interactive shell
#
if ($?USER == 0 || $?prompt == 0) exit
set history=100
set ignoreeof
set prompt=" hostname `{`whoami` } $cwd \!: "
date
#
# not in correct place
#
setenv DEFORM_DIR '/disk1/deform/2d/v60'
```

```
source $DEFORM_DIR/CONFIG.COM
```

In the above file the DEFORM_DIR variable is set for all interactive shells but there is an exit command before the variables can be set if the shell is a non-interactive shell. This will cause the problems with the mesh generator and simulation engine. To avoid this problem, move the statements above the if (...) exit statement

```
#  
# (For security, this default is a minimal set.)  
#  
set path=( . /usr/local/bin /usr/local/lib )  
#  
# new place for the DEFORM_DIR  
#  
setenv DEFORM_DIR '/disk1/deform/2d/v60'  
source $DEFORM_DIR/CONFIG.COM  
#  
# skip remaining setup if not an interactive shell  
#  
if ($?USER == 0 || $?prompt == 0) exit  
set history=100  
set ignoreeof  
set prompt="`hostname` { `whoami` } $cwd \!: "  
date
```

Final comments

Unix is type-case sensitive. Most of the Unix commands are in lower case. Please note that all the main DEFORM system files and commands are in upper case.

In this document, we have briefly explained how to load and execute DEFORM System. If you need further technical assistance, please contact us at the following address/numbers:

Scientific Forming Technologies Corp.

5038 Reed Road

Columbus, Ohio 43220-2514

USA

Phone : 614-451-8313

Fax : 614-451-8325

Email : support@deform.com

Web : www.deform.com

NT/2000/XP Installation: Installation Notes

INTRODUCTION

The following pages describe the installation steps that need to be performed in order to use DEFORM-2D. These steps include: ensuring the proper amount of memory, installing the Security Key device, installing the DEFORM-2D program files and troubleshooting, should you run into problems.

INSTALLATION REQUIREMENTS

Operating system: **Windows NT 4.0+, Windows 2000 or Windows XP**

Hard drive space required: **150 MB.**

INSTALLING THE SECURITY KEY DEVICE

- This Security Key allows the DEFORM-2D system to be highly portable while still allowing only one system to be operable at any time. The actual software can be installed on as many computers as you like, but each DEFORM system will be inoperable without the Security Key.
- Locate the 25-pin parallel port on the back of your PC. (This is sometimes called the LPT1 port)
- Plug the Security Key directly into this port.
- If an existing printer is also plugged into this port it must be plugged into the end of the Security Key.

WARNING

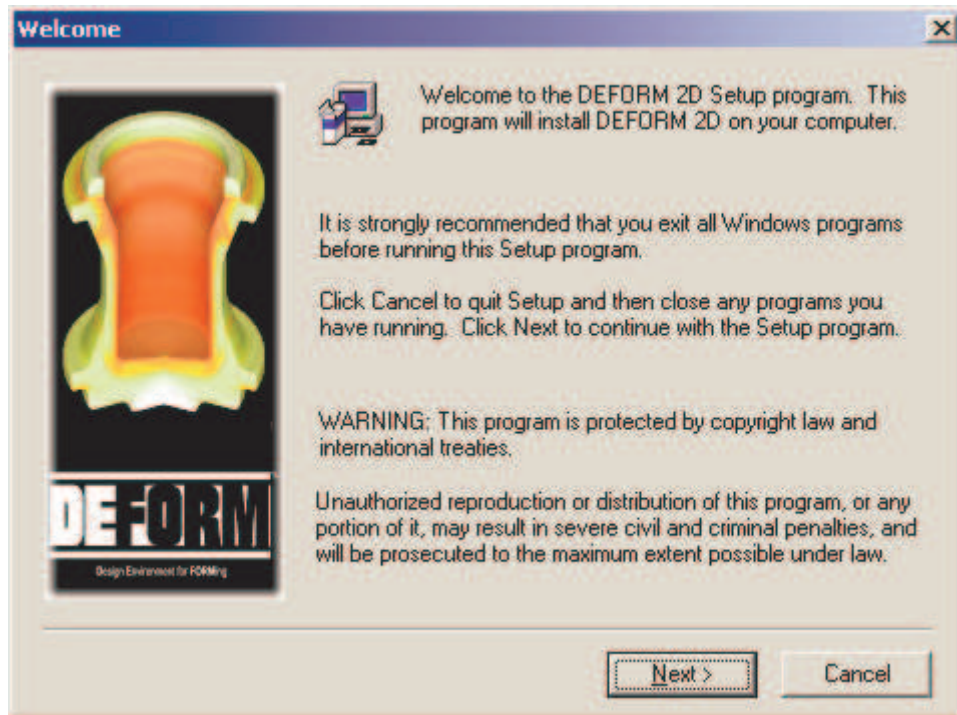
- The Security Key keeps track of the date on your PC. Make sure the date is correct before initially running DEFORM-2D. Changing the date backwards or forwards may make the DEFORM-2D system inoperable.

INSTALLING DEFORM[®] -2D IN WINDOWS

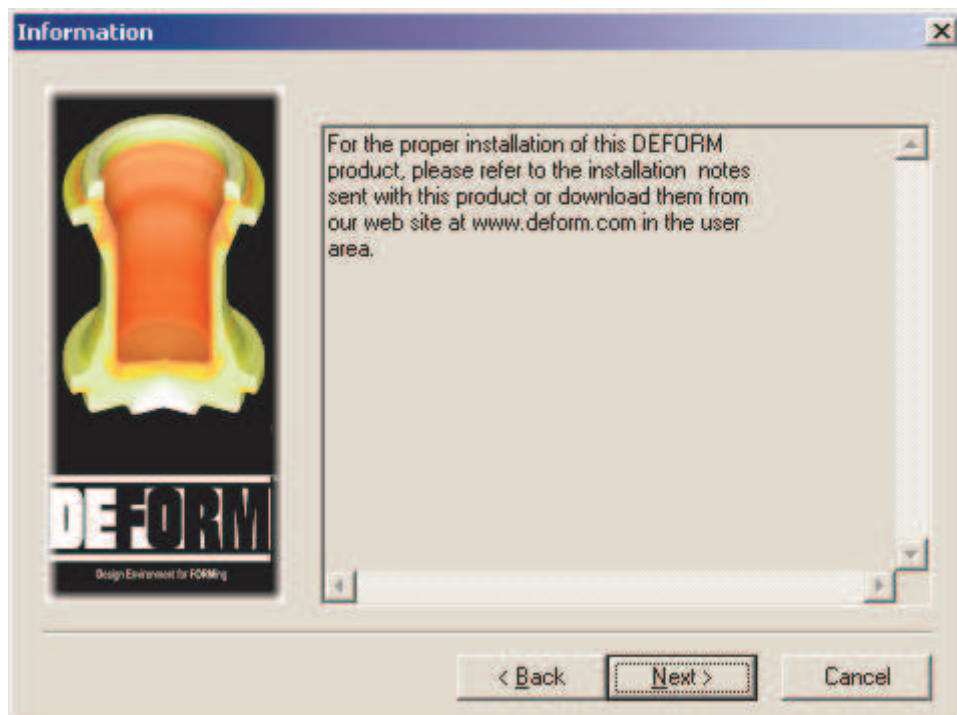
- DEFORM-2D comes on one CD with an automatic installation program. Installation can be done by following a few simple steps:

Important: Before beginning to install DEFORM-2D in Windows NT, it is necessary for the user be logged in as the **Administrator**.

- Start up Windows NT (Log in as the Administrator or with Administrator privileges).
- Insert the CD in the appropriate drive.
- The installation program should start automatically unless this version has been already installed. If it does not start automatically, simply start an NT Explorer window, go to the CD ROM drive and double-click the Setup.exe file.
- The Welcome window comes up. **Click Next.**

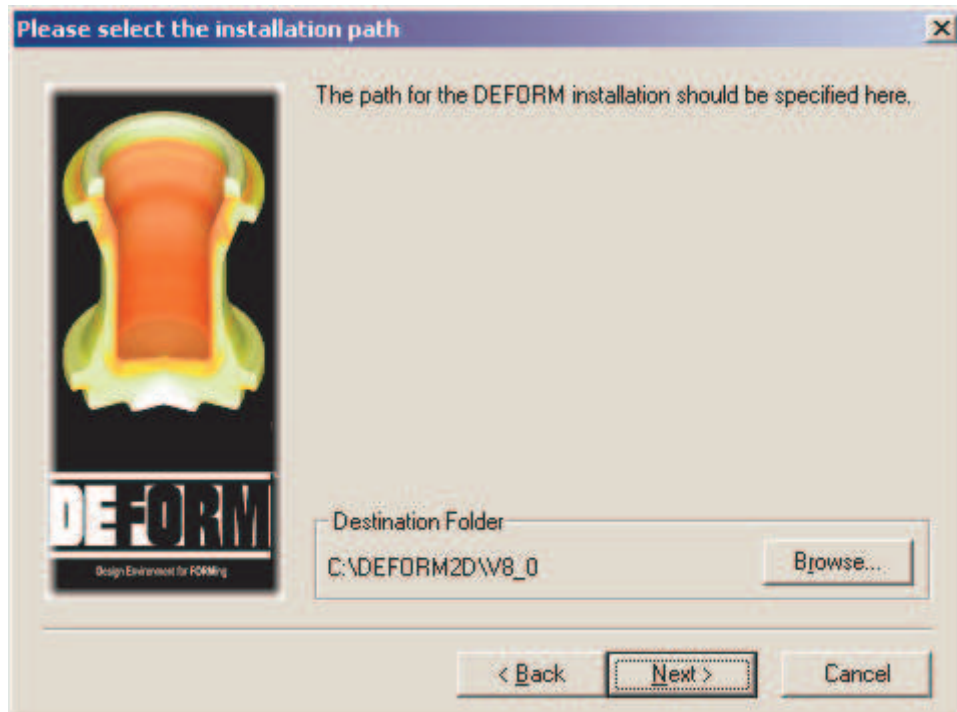


- An information window will appear asking the user to refer to this manual. **Click Next** when finished.

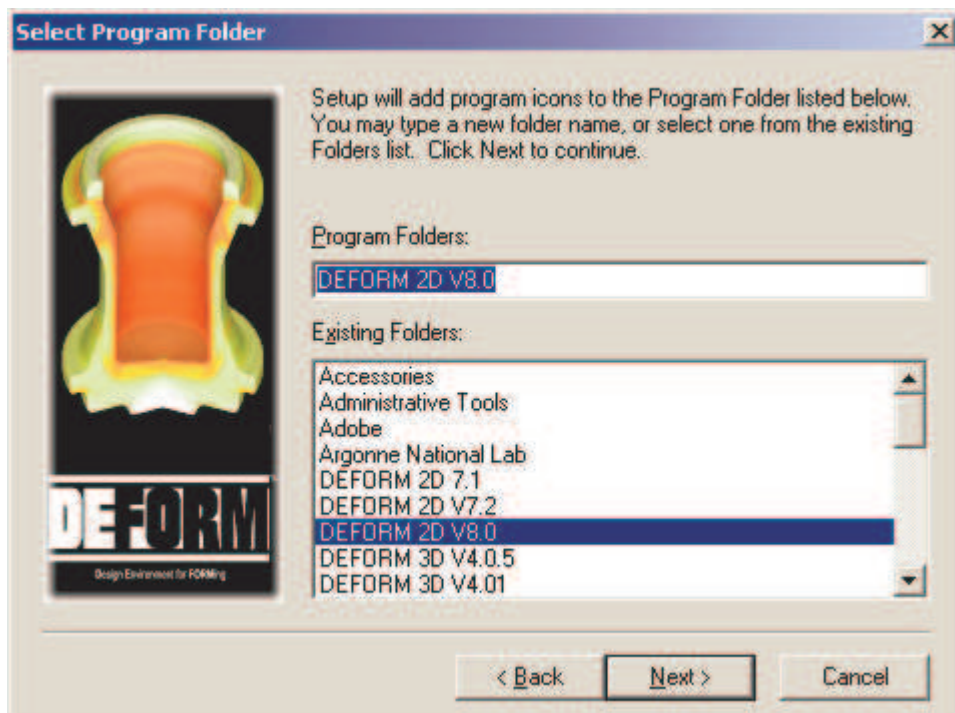


- Choose the Destination location where the files are to be installed or use the Browse button to install in a different folder. **Click Next.** (It is

recommended to choose a drive with ample space for all the **DEFORM™** files that may get generated over time).

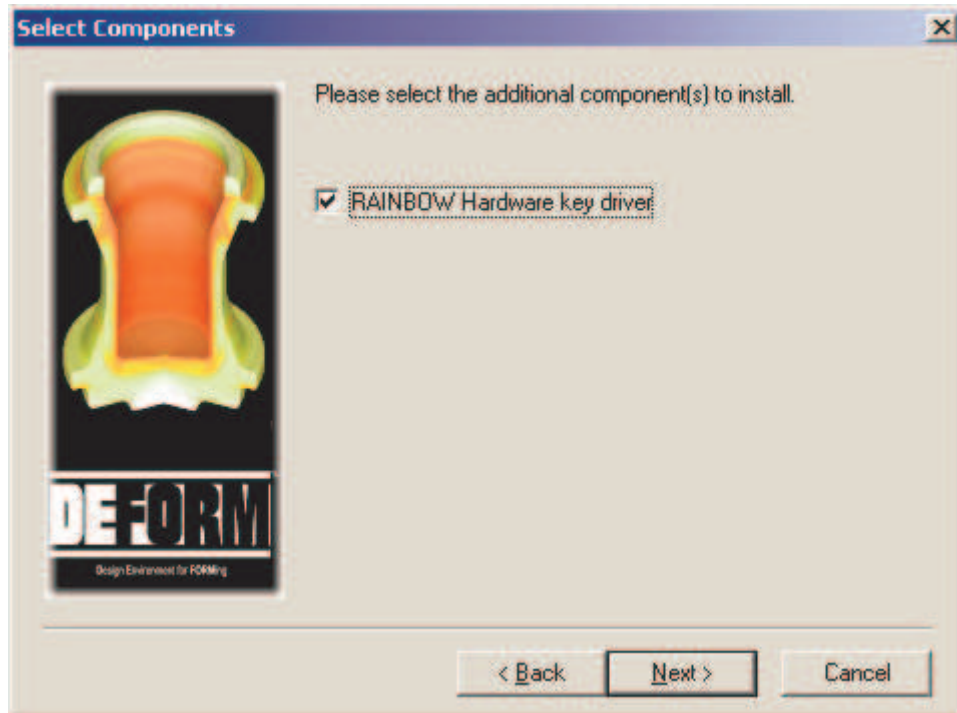


- Select the Program Folder (default recommended) and **Click Next**. The installation program copies program files. This can take a minute.

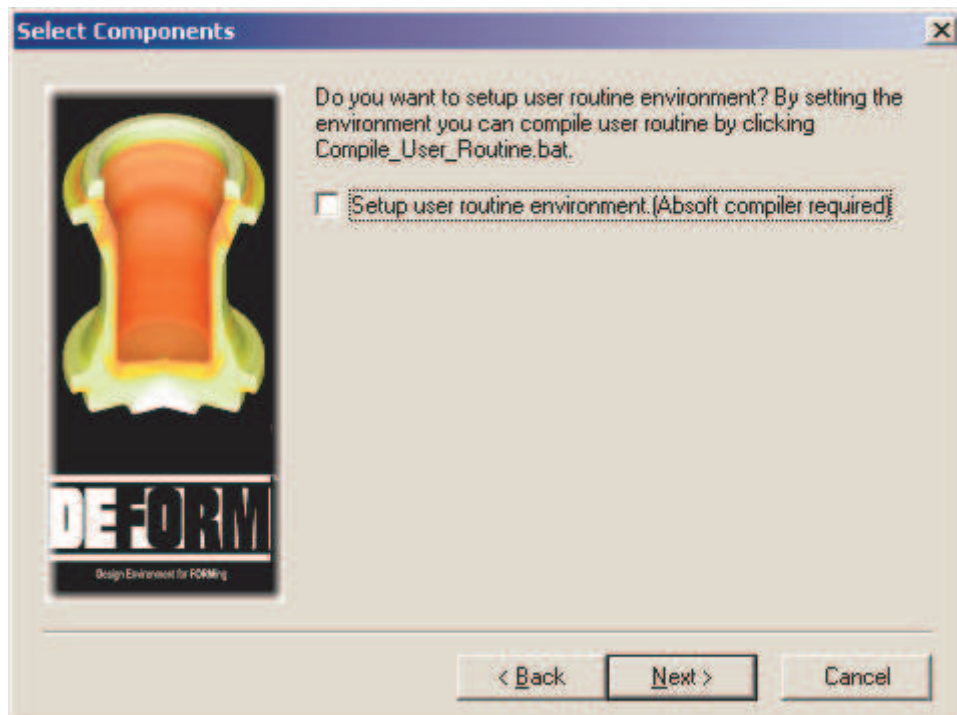


- Setup then asks you if you want to install the Hardware key driver. If this

to run with a local hardware key/dongle, leave the box checked and **Click Next**. If a network license is to be used for this installation, uncheck the box and **Click Next**.



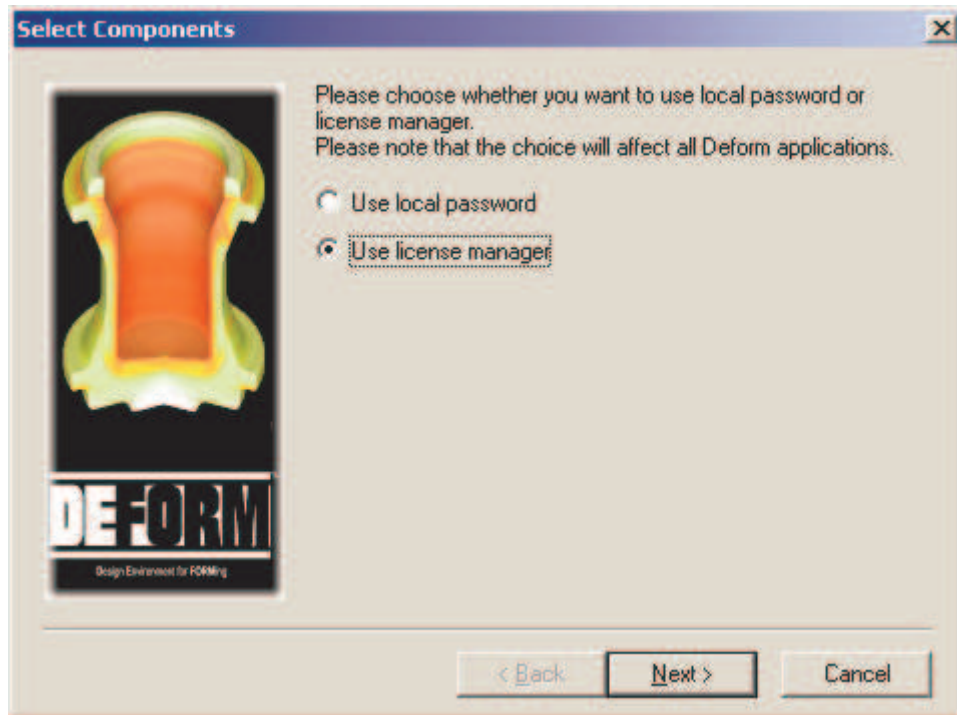
- If user routines are to be used, select the box and **Click Next**. Otherwise, simply **Click Next**.



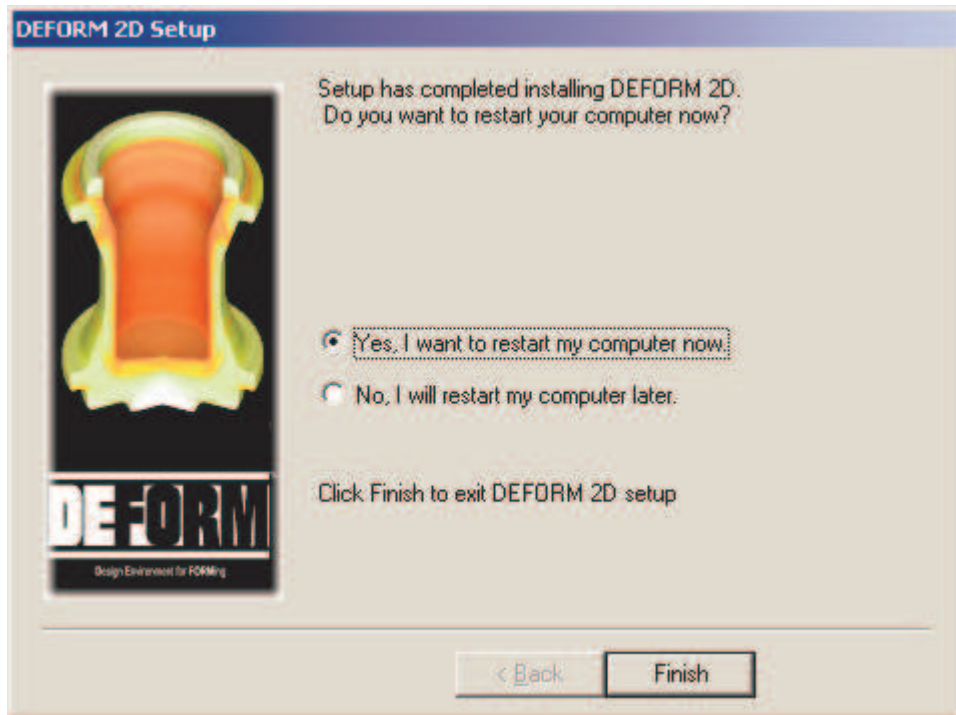
- DEFORM-2D will be installed. This may take a minute or two. If this

takes longer or appears to hang-up, please check the available disk space. Also, some anti-virus software has been known to slow incoming executables by scanning and can slow installation significantly.

- After installation, select the appropriate password scheme, either local password or license manager and **Click Next**.



- At this point, the installation should be finished and rebooting the machine will complete the installation. At this point **Click Finish**.
- The password will need to be installed in the case of a local password. Please copy the file DEFORM.PWD to the DEFORM2D/V8_0 directory where the installation was made and the program should be ready to run.



TROUBLESHOOTING

When you are troubleshooting the installation of DEFORM™, it is recommended to open an MS-DOS prompt from the Start Menu and change directory to where the DEFORM™ files are located. For testing 2D, type run the file DEFORM2.COM by typing the name of file in the DEFORM2D/V8_0 directory in the MS-DOS prompt. This will show the exact error message in the MS-DOS window.

Error:

The message

"Security Parameters do not match, please contact Scientific Forming Support Personnel"

is displayed.

Problem:

Your DEFORM-2D password has expired.

Solution:

Contact Scientific Forming Technologies Corp. for a new

password file (DEFORM.PWD).

Error:

The message
"Hardlock Key is not found"
is displayed.

Problem 1:

The Security Key cannot be located on the system by DEFORM-2D. This is probably a result of either the Security Key not plugged in to the parallel port tightly, or not on the system at all.

Solution 1:

Check the back of your PC. If the Security Key is not plugged in, locate the parallel port (usually LPT1) and plug the Security Key into this port. If the Security Key is plugged in, remove it and re-insert it making sure that it is completely inserted and try running DEFORM-2D again.

Error:

The message
"Computer clock is not correct"
is displayed.

Problem:

The computer's internal time and date have been changed since DEFORM-2D was last run.

Solution:

Check the time and date and make sure that they are correct. If they are both correct, then DEFORM-2D was run at least once with a newer time and date. Once DEFORM-2D have been run, the time and date cannot be set back. If they are not correct, and display an earlier time or date, update them to the current time and date and try running DEFORM-2D again.

QUESTIONS / PROBLEMS

If you have any questions, comments, or problems installing DEFORM-2D, contact:

Address:

Scientific Forming Technologies Corporation

5038 Reed Road

Columbus, OH 43220-2514 USA

Phone: (614) 451-8330

Fax: (614) 451-8325

E-mail: support@deform.com

Or visit our website at

<http://www.deform.com>

Chapter 7: Quick start tutorial

7.1. 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

1. Problem Title (optional)

Descriptive title for the problem – will be displayed on the screen during pre- and post-processing.

2. Unit System

Select English or SI units. This will change many default values and affect how material data is imported.

3. Select Axisymmetric or Plane Strain

An axisymmetric part can be completely defined by rotation of a 2 dimensional cross section, or significant details can be captured in a 2D section (for example – a bolt with a hex or Torx[®] head).

Buckling problems are not axisymmetric problems. An axisymmetric simulation will *NEVER* show buckling or bending, even if it will occur in production. Refer to the manual for more information.

4. 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

1. Define Object Type

For most simulations, plastic object type is suitable.

2. Define the Geometry

Geometry can be defined from a **2D** IGES OR .DXF file, or by entering table data.

Workpiece geometry requirements:

- The $x=0$ line is the centerline of the part. If the part is solid, it must be located on the centerline. If it is hollow, the left edge of the part should be positioned at the internal radius.
- Geometry must be defined counterclockwise
- Always check the geometry.

1. Mesh the Object

Typical progressions should use 1000 to 1500 elements. 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

1. Define Boundary Conditions

For *solid* parts, define the velocity in the X direction along the centerline to be 0. No velocity boundary condition is required for hollow parts.

1. Define the Tools

1. Assign Tool Names (optional)

For reference during pre- and post-processing.

2. 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.

1. 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 -90 degrees for punch on top). The stroke value should generally not be changed by the user.

1. Define Interobject Data

1. Inter-Object Relationships

The workpiece should be slave to both tools. For well lubricated cold forming, a friction factor of 0.08 is reasonable.

2. 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.

3. Generate Contact Boundary Conditions

The default value of tolerance is adequate.

2. Complete Simulation Controls

1. 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.

1. Select the Primary Die

Enter the object number for the punch(generally object #2)

2. 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.

3. 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.

4. Set substepping control

Under Advanced Step Controls, set **Strain per Step** to 0.025.

1. 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.

2. Running a Second Operation

1. 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.

2. Loading Simulation Results Into the Preprocessor.

Return to the preprocessor, and load the appropriate step from the database.

3. 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.

4. Positioning Objects

From Interobject reposition the tools against the workpiece using interference.

5. Generate Contact

Initialize and generate contact boundary conditions.

6. Reset Simulation Controls

Determine total steps and stroke per step as described above.

7. 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.

8. 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.

9. 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.

7.2. 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.

1. Problem Title (optional)

Descriptive title for the problem – will be displayed on the screen during pre- and post-processing.

2. Operation Name

Operation name

3. Unit System

Select English or SI units. This will change many default values and affect how material data is imported.

4. Select Axisymmetric or Plane Strain

An axisymmetric part can be completely defined by rotation of a 2 dimensional cross section, or significant details can be captured in a 2D section (for example – a pinion gear where fill in the teeth is not significant. An average pitch diameter which gives the same cavity volume should be used).

Buckling problems are not axisymmetric problems. An axisymmetric

simulation will *NEVER* show buckling or bending, even if it will occur in production. Refer to the manual for more information.

5. 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 FILE->LOAD->KEYWORD menu on the main preprocessor window.

5. Define the Workpiece

1. Define Object Type

For most simulations, plastic object type is suitable.

2. Define the Geometry

Geometry can be defined from a **2D** IGES OR .DXF file, or by entering table data.

Workpiece geometry requirements:

- The $x=0$ line is the centerline of the part. If the part is solid, it must be located on the centerline. If it is hollow, the left edge of the part should be positioned at the internal radius.
- Geometry must be defined counterclockwise
- Always check the geometry.

1. Mesh the Object

Typical progressions should use 1000 to 1500 elements. 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

1. Define Thermal Boundary Conditions

For *solid* parts, define heat exchange with the environment on all faces *except* the centerline. For hollow parts (rings), define heat exchange with environment conditions on all faces.

2. Initialize Object Temperature

Set the initial object temperature to the furnace soak or preheat temperature.

1. Complete Simulation Controls

1. Define Number of Steps

A typical transfer operation can be simulated in 10 to 20 steps.

2. Select the Primary Die

Enter object 1 (only object defined) as the primary die.

3. Calculate the Time per Step

Divide the total transfer time by the number of steps.

2. 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.

3. Loading Simulation Results Into the Preprocessor to Define Second Operation

Return to the preprocessor, and load the last step from the database.

4. Define Material Data for the tools

Specify thermal data for the tool material

5. Define the Tools

1. Assign Tool Names (optional)

For reference during pre- and post-processing.

2. 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.

1. 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.

2. Assign Tool Material

Be sure the proper material is assigned to both the tools and the workpiece

3. Assign Tool Temperature

Set the uniform tool temperature before forming begins.

1. Define Interobject Data

1. 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.

2. Object Positioning

Using interference positioning, position the workpiece on the bottom die. Leave the top die away from the workpiece during this operation.

3. Generate Contact Boundary Conditions

The default value of tolerance is adequate.

2. Complete Simulation Controls

1. Define Number of Steps

A typical die resting operation can be simulated in 10 to 20 steps.

2. Select the Primary Die

Enter object 1 as the primary die.

3. Calculate the Time per Step

Divide the total transfer time by the number of steps.

4. Define Operation Name (optional)

3. 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.

4. Loading Simulation Results Into the Preprocessor to Define Forming Operation

Return to the preprocessor, and load the last step from the database.

5. Set Simulation Controls

1. Operation Name (optional)

2. Select Simulation Mode

We will now simulate the forging operation, so turn deformation on (yes)

6. 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.

7. 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 -90 degrees). The stroke value should not be changed unless a mechanical press model is used.

8. Define Interobject Data

1. 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.

2. Object Positioning

Using interference, with the workpiece as the reference object, position both tools in contact with the workpiece.

3. Generate Contact Boundary Conditions

The default value of tolerance is generally adequate

9. Complete Simulation Controls

1. 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.

1. Assign the Primary Die

The top die (or punch) should be the primary die. This will generally be object #2.

2. 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

3. 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.

4. Set substepping control

Under Advanced Step Controls, set **Strain per Step** to 0.025

1. 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.

2. Running a Second Operation

1. 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.

2. Loading Simulation Results Into the Preprocessor.

Return to the preprocessor, and load the appropriate step from the database.

3. 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.

4. 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.

5. Positioning Objects

From Interobject reposition the tools against the workpiece using interference.

6. Generate Contact

Initialize and generate contact boundary conditions.

7. Reset Simulation Controls

Determine total steps and stroke per step as described above.

8. 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.

9. 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.

10. 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.

Appendix A: Running an inertia weld simulation in DEFORM

DEFORM™-2D includes special features to simulate uniform rotation and distortion about the Z axis. The torsional distortion mode uses a different element formulation, which considers out-of-plane shear strains (r, θ and z, θ) as well as the 3 normal + rz shear strains that are considered in a normal axisymmetric problem.

To simulate the inertia weld process, the following adjustments should be made:

- 1) Under **Simulation Controls**, set *Geometry* (GEOTYP) to Torsion. Be sure both *Heat Transfer* and *Deformation* are set to “Yes”
- 2) Enter the geometry for the weld zone for both deformable objects.
- 3) Each deformable object should be backed by a rigid object. One will be moving, the other will be stationary.
- 4) The rotation of the rotating part is controlled under the **Movement Controls->Rotation** tab. The rotation can be controlled by constant velocity, constant torque, or by flywheel inertia, for flywheel controlled weld equipment.
- 5) The translational (loading) force is assigned under translation controls as normal.
- 6) Contact between the deformable object and the rigid backing object must be defined manually, using “-n,-n” boundary conditions, which “lock” the nodes into position on the master surface.

This is done using **Advanced Deformation Boundary Conditions**



For each deformable/rigid contact interface, select the nodes on the deformable object along the contact zone with the rigid. Enter a BCC code ‘-n’ in the **Boundary Conditions** window (where n is the object number of the rigid contact

object). Set the direction to X and Generate BCC's



Then change the direction to Y and Generate BCC's on the same nodes.

Repeat this process for all other rigid/deformable contact pairs.

NOTE: The weld contact interface must NOT have -n,-n boundary conditions.

7) Define the rest of the preprocessor settings, including all interobject relationships, as you normally would.

Appendix B: 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 program

```
C:\deform2d\v7_2\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 substepping 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.

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:\DEFORM2D\7_2\DEF_ARM_CTL.COM DEFORM_DEMO B
```

The *B* indicates that the simulation should be run in *Batch* mode.

Appendix C: 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”]

Appendix D: Adding Gas Trap Calculation to a Simulation

In 2D, gas trapping can be predicted if the file DEF_GAS.DAT is added to the directory where a simulation is running. The contents of the file DEF_GAS.DAT is unimportant in that the file is merely a flag to indicate that gas trapping is to be performed.

The gas trapping is modeled as a perfect law for any time a boundary of a deformable object traps a region of a rigid object. When a volume is closed off by contact with a single body (See Figure 112), a pressure is applied to the surface of the body which increases as a perfect gas law. At the end, slight underfill can be noted in the workpiece by highlighting the contact nodes (Figure 113). In the case that the boundary of the deformable traps a split between two objects, there will be no trapping even if the two objects are flush or overlapped.

If a gas trap simulation is run, delete the files DEF_GAS.DAT and DEF_GAS.INI in the case of running a simulation that does not use gas trapping. The file DEF_GAS.DAT is a flag used activate gas trapping. The file DEF_GAS.INI is a file output by the program and should not be changed during a simulation.

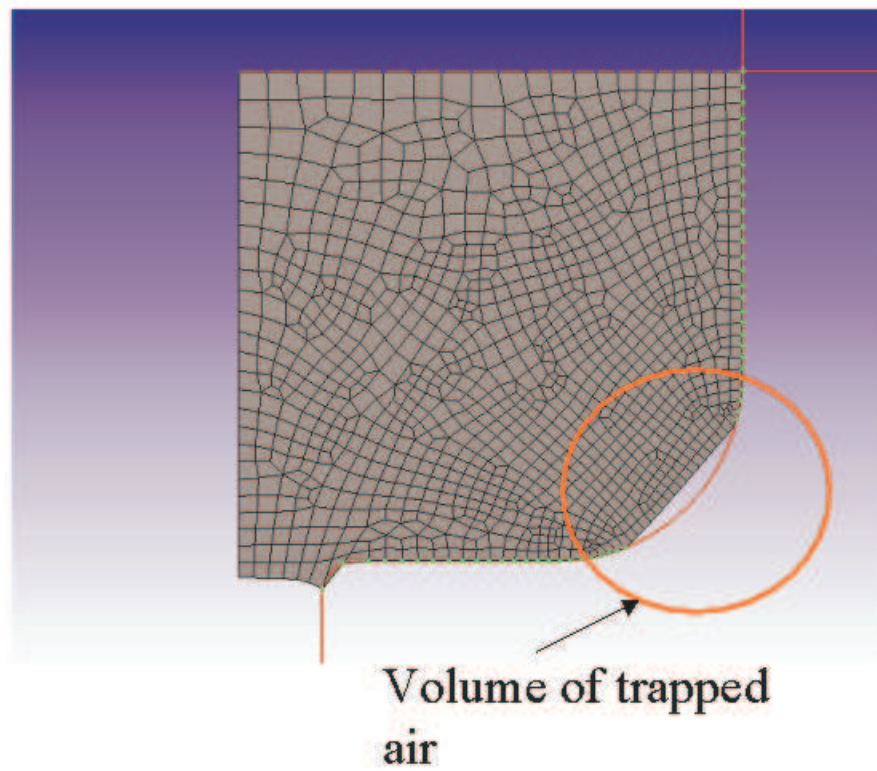


Figure 112: A trapped volume under compression. The red circle shows where the air is trapped as the workpiece is being extruded.

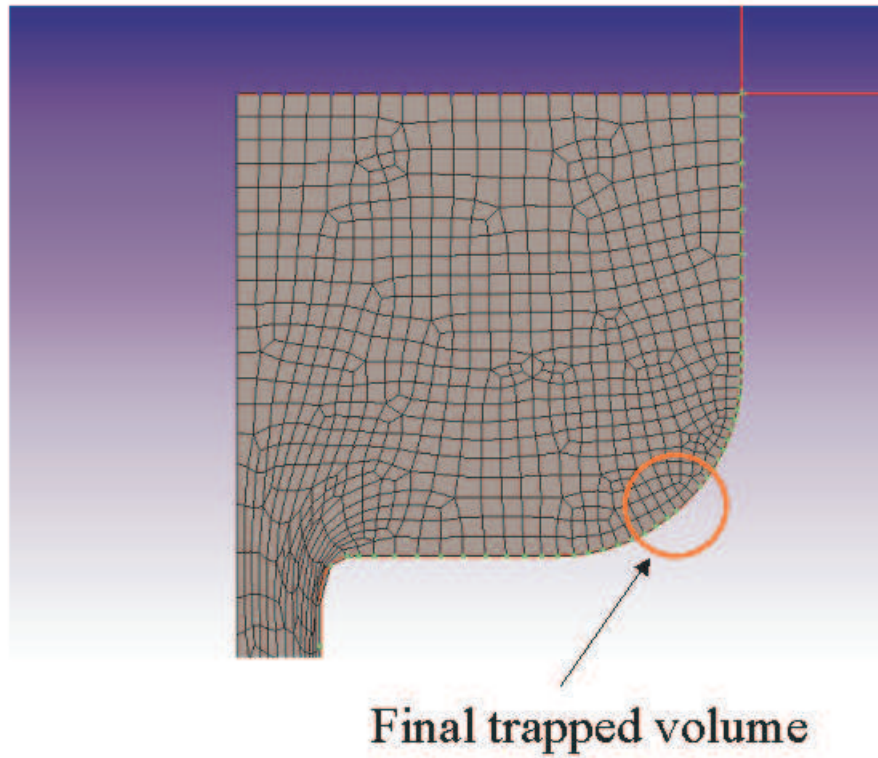


Figure 113: The final trapped volume. The final volume is small but it can be easily seen as slight underfill by the highlighted contact nodes.

Appendix E: Modeling of a Spring-Loaded Sliding Die Using Force Controlled Objects

Note: This method is no longer recommended since there is now a spring-loaded die movement control in DEFORM-2D.

Overview:

A simulation of a spring-loaded sliding die can be broken down into three typical phases of sliding die movement

- 1) Force on the sliding die is below the preload - die is stationary resting on a stop or another die
- 2) Force on die exceeds preload - die begins moving under spring load control
- 3) Die hits a stop - forming process completes with die stationary.

Note: Depending on the type of sliding die being modeled, this behavior may or may not be fully realized in any particular simulation. For example, some pneumatic springs don't require the modeling of a preload. In such a case, the first phase can be omitted.

Methods for modeling sliding dies in DEFORM™

Option 1: Make the sliding die the primary die, use load and/or stroke stopping controls to stop the simulation when die starts (Preload exceeded) or stops motion (spring goes solid), then change movement controls and restart.

Option 2: Use artificial elastic spacer(s) to allow the spring-loaded die to react against deformable body, and allow simulation to run through (More initial setup time but less intervention during run time) .

Option 1 described:

Phase 1: (See Figure 114)

- a. Set object 2 velocity control
- b. Object 4 primary die, no movement control
- c. Stop on load = preload on object 4
- d. Step control by time (primary die is not moving)

Phase 2: (see Figure 115)

- a. Object 4 load control. Direction should be direction load is applied, not direction of movement.
- b. If movement of object 4 is limited, stop on stroke can be set.

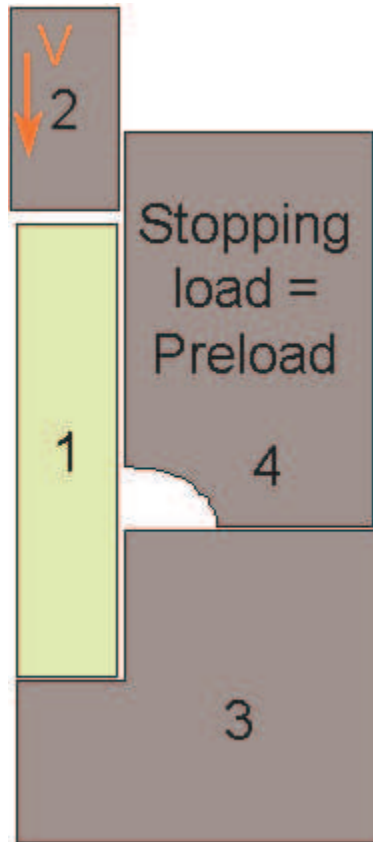


Figure 114: Sketch of spring-loaded die simulation with spring-loaded die set with a stopping criteria based on load.

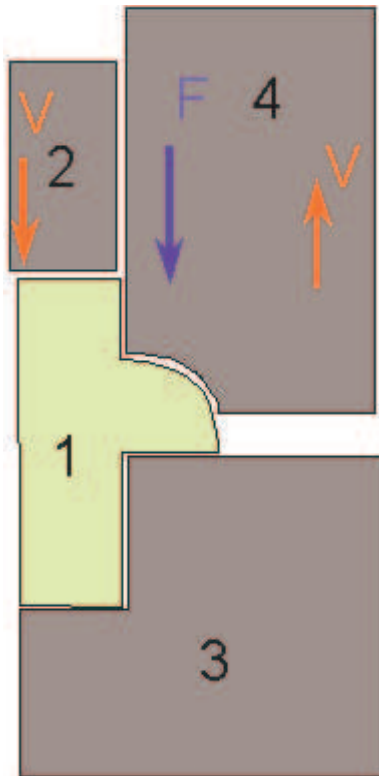


Figure 115: Sketch of spring-loaded die simulation with spring-loaded die set with movement type based on load.

Option 2 described: Using an artificial elastic “spacer” (See Figure 116)

- a. Create a thin elastic object (5 in this case) with about 10-20 elements
- b. Make object 5 slave to both objects 3 and 4
- c. Set movement on object 4 to load control, with the load as the preload.
- d. Run the simulation. When load exceeds preload, die will automatically move up.

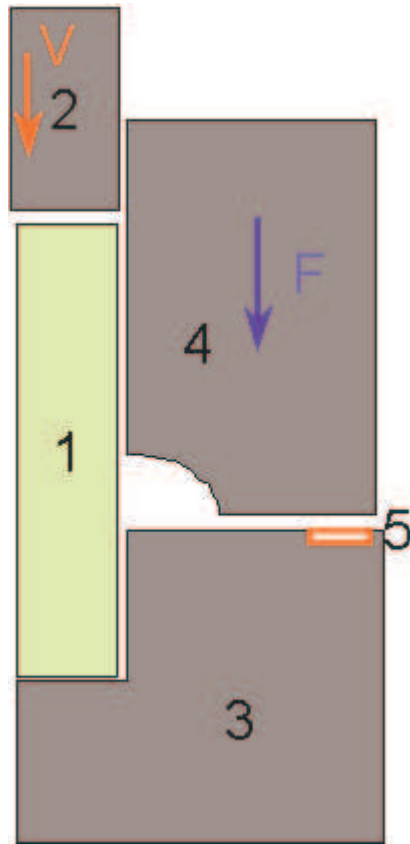


Figure 116: Sketch of spring-loaded die simulation with spring-loaded die set with movement type based on load and an additional elastic body (5) that provides a reaction force for the spring-loaded die (4).

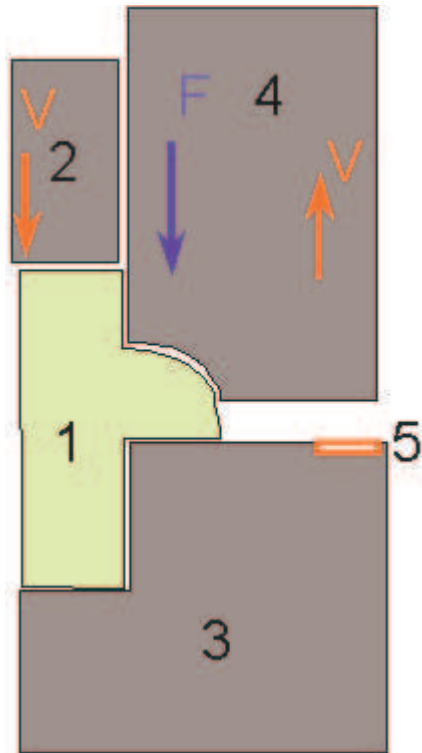


Figure 117: Sketch of spring-loaded die simulation with spring-loaded die set with movement type based on load and an additional elastic body. At this stage, the material has filled beneath the sliding die and is pushing the sliding die upward.

Appendix F: Using the Inverse Heat Transfer Template

Purpose: The purpose of this template is to aid in the estimation of heat transfer coefficients based on simulation results being optimized with experimental temperature data.

Procedure: Please refer to the heat transfer template lab available in the Htlabs.pdf file included with the documentation.

Appendix G: 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 118 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: $y = Ce^n e^m + y_0$, it is accurate for y_0 to be equal to the initial yield stress. If $y_0 = 0$, 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.

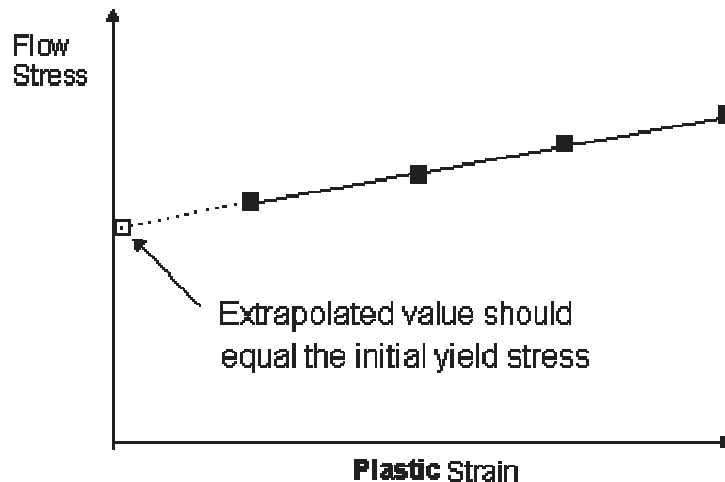


Figure 118: Extrapolation of flow stress data to determine the initial yield stress.

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 springback 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 Springback

- To calculate the residual stress or springback during unloading, one of two methods may be used.
- If the amount of springback or residual stress is the chief concern, remove the dies, put enough constraint on the workpiece 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.

Appendix H: Die Stress Analysis - Theory

(Still under construction)

Appendix I: Running creep simulations in DEFORM-2D

Creep is slow, continuous deformation with respect to time. The strain instead of depending only on the stress is a function of temperature and time. To run creep/stress relaxation simulations in DEFORM, the main issue is acquiring the material data. There are several common laws available for the creep modeling that can be seen in the material data section. If a specimen is under a constant tension and a given temperature, creep will give a changing strain over the total amount of time (See Figure 119). There are three different modes of creep: primary, secondary (steady-state), tertiary. In Figure 119, the scale is not correct in that the secondary creep generally has a much larger range of time compared to the other two modes. Secondary creep is the only mode that is modeled and fitting experimental data to a power law curve does this as below:

$$\dot{\varepsilon} = B\sigma^n$$

where:

$\dot{\varepsilon}$ = strain rate

B, n = material constants

σ = stress

Many of the available forms are just fancier versions of this basic form (known as the power law curve). It is easy to fit data to this curve by plotting log-log curve and fit the curve to a linear slope to obtain both B and n (or whatever form is being used).

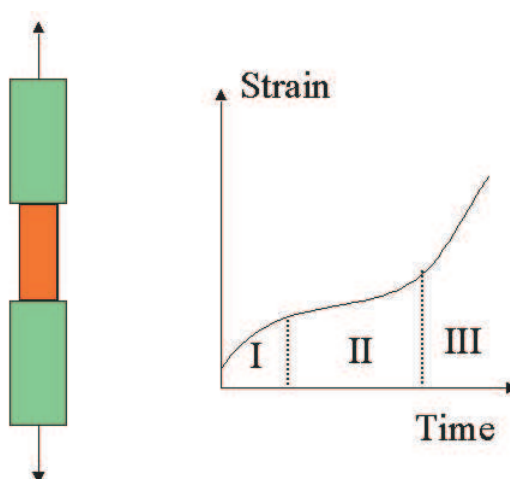


Figure 119: Strain versus time response for a specimen under constant load at a given temperature exhibiting creep behavior. Note the three different regions of response in the strain-time curve, I. Primary creep, II. Secondary (Steady State) creep and III. Tertiary creep.

To run a creep simulation, the following requirements must be fulfilled:

- The workpiece should be made elasto-plastic
- Creep has been activated in the object->properties->deformation window (See Figure 120).

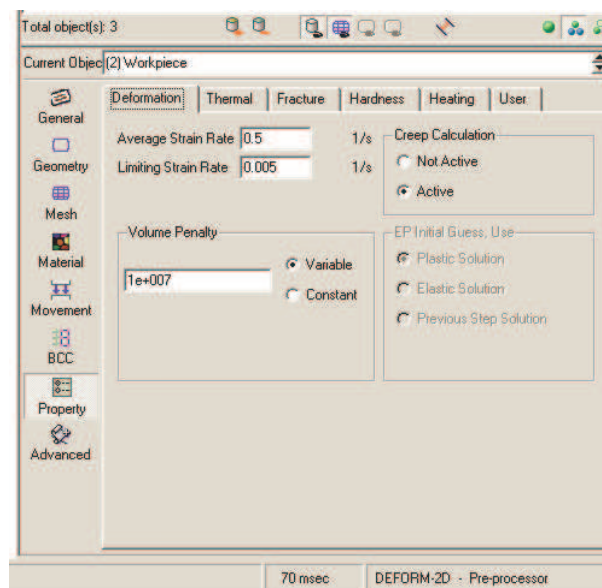


Figure 120: Object properties window where creep is activated.

- A creep model with non-trivial data has been defined (any of the given is sufficient). In the case of user routines, the required routine should have the routine compiled and the simulation should run the special FEM engine.
- There is either a non-zero stress state on the part (relaxation case) or an applied traction to the body (creep).
- Constrain the workpiece with velocity boundary conditions alone as the penalty method of contact can give some numerical error in the stress solution for creep cases.

Simple example:

To test the functionality of the creep model, a simple case can be constructed by running the UPSET.KEY in the DATA directory. A few changes are required to get this simulation to run better:

- Change the object type to elasto-plastic
- Change the total displacement to a small distance of 0.04 mm.
- Change the material law to something simple. For example, a simple case of linear hardening model gives very easy to hand verify stress values in the workpiece. In this case, the equation $\sigma = 2000\varepsilon + 10$ was used (See Figure 121).

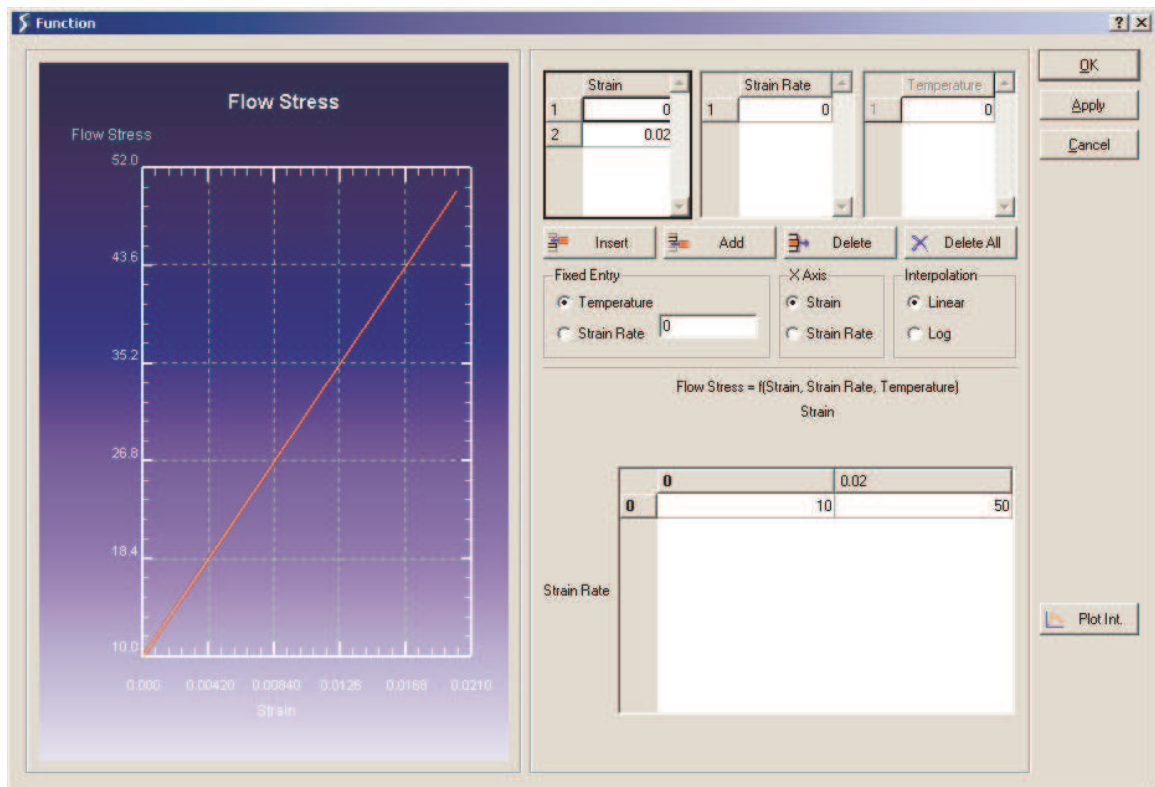


Figure 121: Flow stress data for example case.

- Make the friction between the tools and workpiece zero. This will yield a uniaxial stress condition.

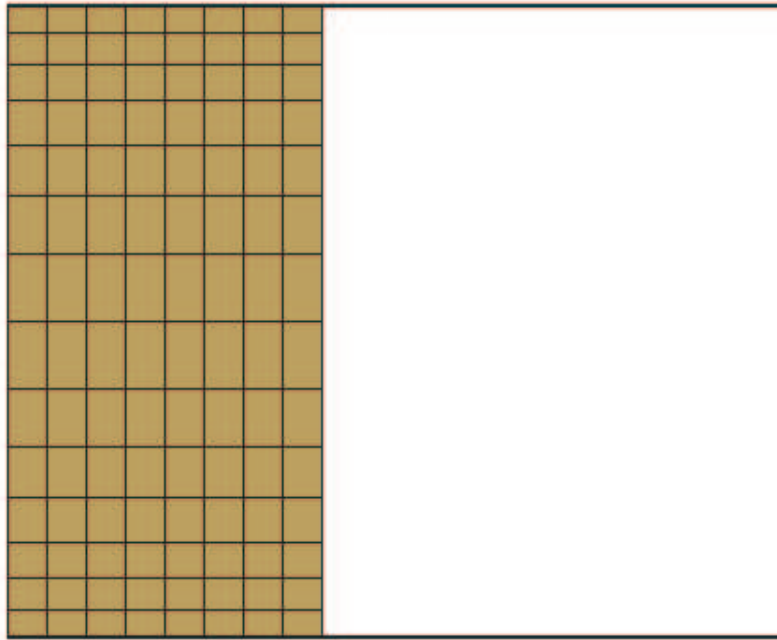


Figure 122: UPSET.KEY object view.

After the simulation is finished, a stress relaxation simulation can be performed. The following changes need to be made to the last step simulation:

- Add more steps to the simulation
- Increase the time step size
- Delete the rigid objects
- Add fixed velocity boundary conditions $V_y = 0$ for the top and bottom of the workpiece and $V_x = 0$ at the centerline.
- Add creep material data. This data is rather difficult to obtain since it is a pure material property and not obtainable without experimental testing.
- Turn on creep in the object properties window.

The simulation should run fairly quickly and the stress should relieve rather quickly. The stress results can be obtained by point tracking.

Appendix J: On Manually Importing Material to DEFORM.

(Still under construction)

Appendix K: Generating a Circular Geometry With a Hole.

This case involves the two features of generating a circular geometry and adding an internal hole. To create a circular geometry, go to the geometry window of the desired object.

1. Click the Edit tab and select the Line-Arc format radio button as seen in Figure 123. 2. Set the type to Arc.
3. The first point X_1, Y_1 is the start of the arc. The second point X_2, Y_2 is the center of the arc. The angle is the amount of sweep of the arc. Input the following values: (1,0) for X_1, Y_1 ; (0,0) for X_2, Y_2 ; and 360 for the angle.

This will yield the result seen in Figure 124.

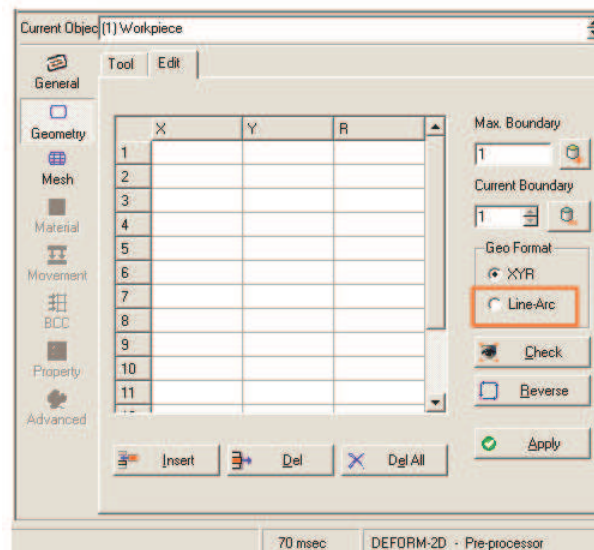


Figure 123: Setting the geometry type to Line-Arc.

4. Increment the boundary number in the geometry window and select the second boundary (See Figure 125).
5. Set the geometry type to Arc and set the curve as follows (0.5,0) for the first point; (0,0) for the second point; and -360 as the angle.

At this point a mesh can be generated which will have a hole in the middle (See Figure 126).

Note: The exterior boundary was counter-clockwise while the interior boundary was clockwise.

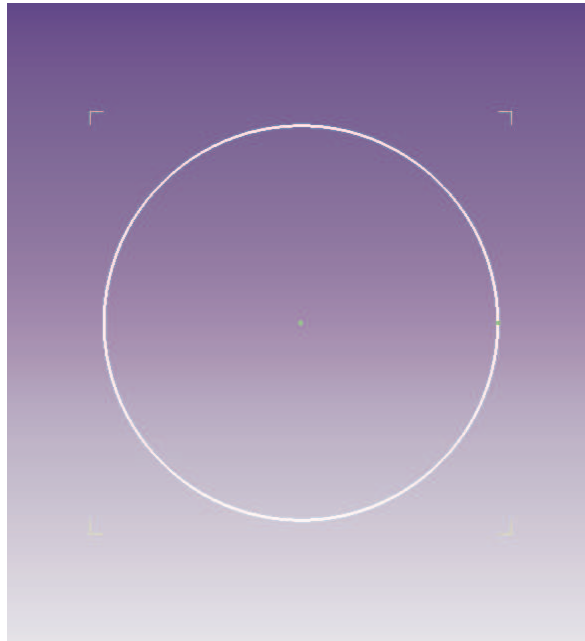


Figure 124: Generating a circular geometry.

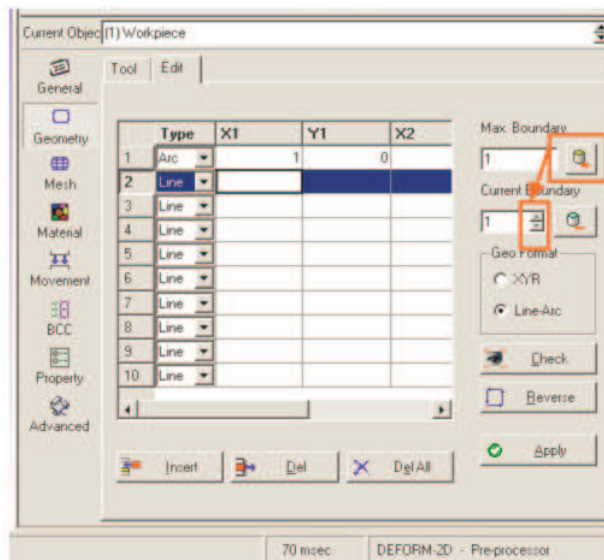


Figure 125: Incrementing the boundary number and selecting the second boundary.

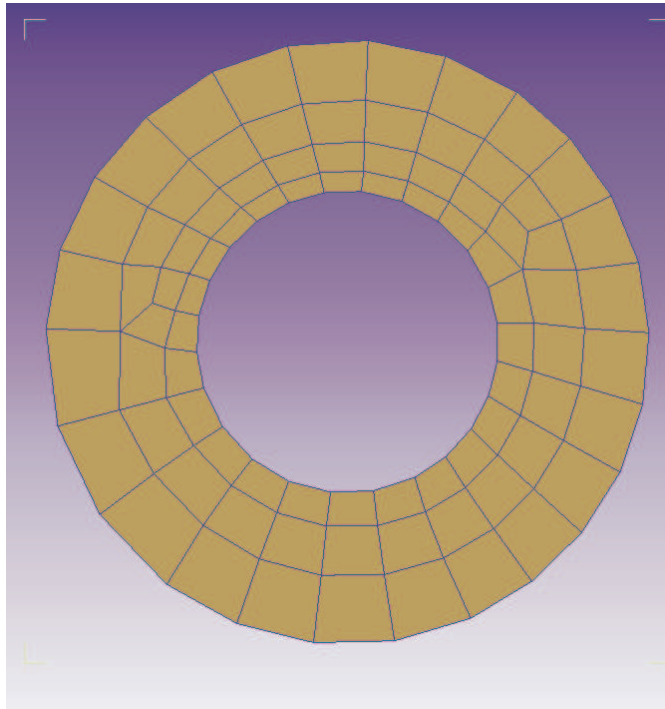


Figure 126: A circular result with a hole.

Appendix L: Launching the Simulation Engine from a Command Prompt (Windows only)

Purpose: Sometimes a new engine is given to a user for a special application that is not covered in a release version. The user can use this document to see how to run the engine without having to overwrite the current engine.

Step 1: Place the new engine in the installation directory of DEFORM-2D being careful not to overwrite the current engine.

When receiving the engine, it must be located in the directory where DEFORM-2D is installed. The current engine named will be named DEF_SIM.EXE so be sure not to copy the new executable with that name to the install directory of DEFORM-2D.

Step 2: Rename a current database to the name FOR003 in the necessary problem directory.

This is the name of the database while a simulation is running. After the simulation is completed, be sure to rename it back to its original name. *Warning: Make sure that the simulation is finished before renaming it back or risk losing all the work.* If you are unsure, rename it back to a different name than the original name.

Step 3: Execute the Simulation Engine.

This is done by opening a dos prompt and changing to the directory where the problem is located. A dos prompt can be accessed by either selected a shortcut from the Start Menu or by typing **cmd** in the Run menu. Once the dos prompt is open, it will look like the Figure 127. Let's assume that the dos prompt starts in the C:\ directory and the problem we want to run is the C:\Settings and Documents\Test directory. To navigate from the C:\ directory to the end directory type the following:

```
cd "Settings and documents/test"
```

and the current working directory will become Settings and documents/test. To then run the new version of DEFORM-2D, make sure that the file FOR003 is in this directory. After this, execute the new engine by calling the executable in the DEFORM-2D directory. An example is seen in Figure 128.

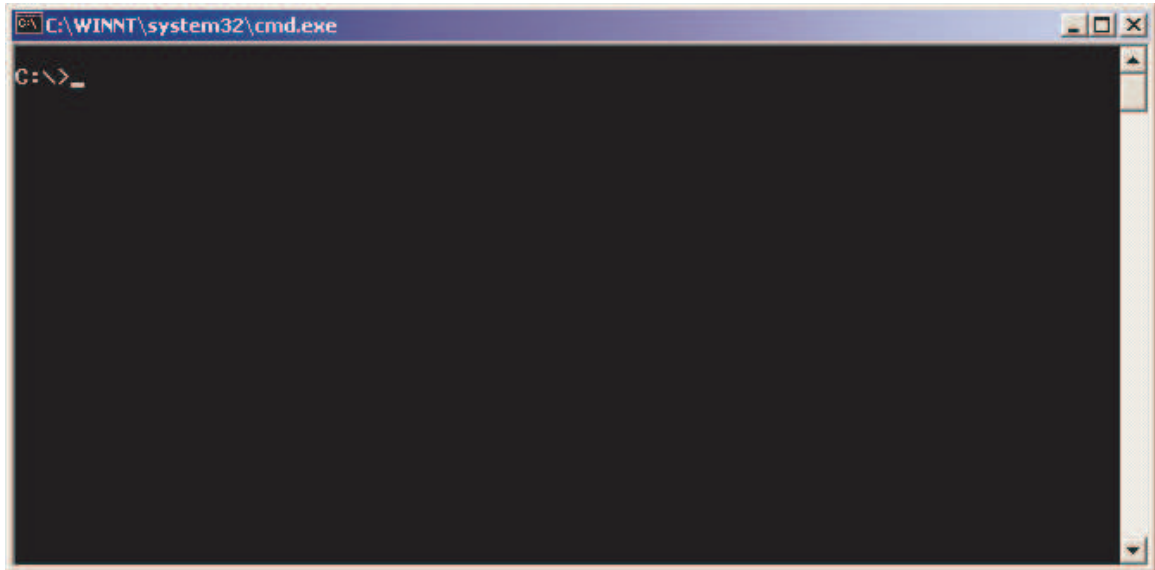


Figure 127: A new dos window.

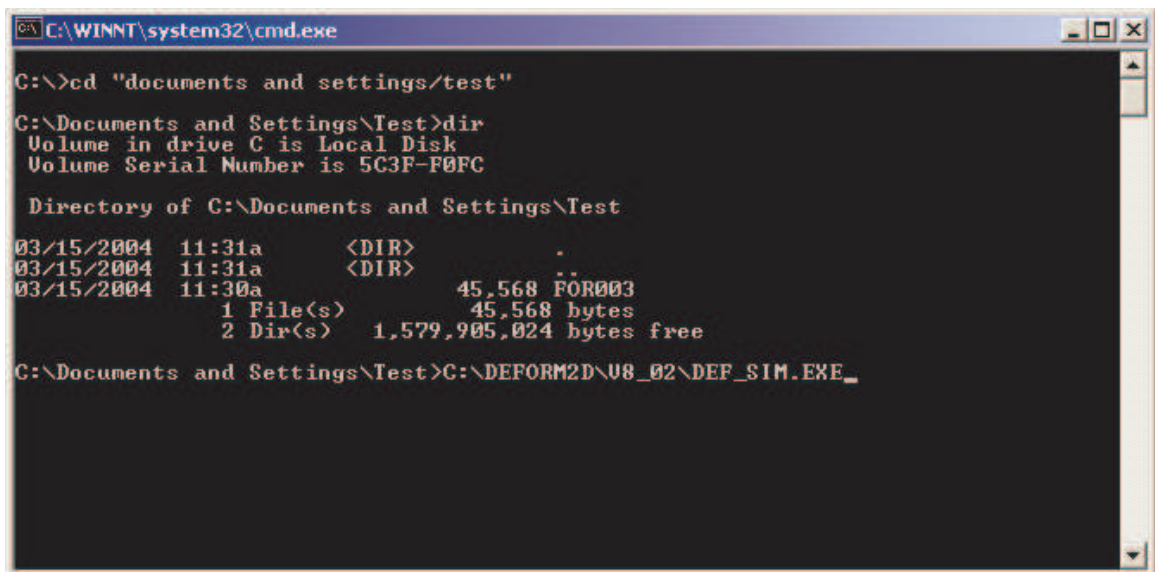


Figure 128: Running a simulation from a dos window.

Appendix M: Finer Points on Using Force Controlled Dies

Objective:

In many simulations, force-controlled dies are used in order to simulate real-world tool loading conditions. As seen in Appendix E, these can be used in order to simulate spring-loaded dies. The purpose of this section is to explain the pitfalls that can occur while trying to simulate force-loaded dies in terms of obtaining good results and improving convergence of the solution

Theory:

In reality, there is no such thing as an ideal step function. It takes time, no matter how small, for things to change. If a sudden load is applied, there must be a time span over which things occur as a ramp function. In the case of applying a load to a deformable object, there are generally two ways in which a body may respond to a given amount of force. The body may increase its reaction load by strain hardening or increase its reaction load by strain rate hardening. In the case of a high load, the final deformed shape may be quite different than the initial shape. The problem with a step function is that over one step, the body may have to deform much to achieve equilibrium with the applied load. Also, with two different possible ways in which to react can place much pressure on the simulation engine.

Solution:

As implied with the above statement on step functions, if there are no step functions then the solution must be to use ramp functions. This is the key to getting all of these simulations using force control to converge nicely. The time step size may be very small (ideally it should be realistic to the process) but as long as the deformation is spread over an ample number of steps, a good solution will result.

Appendix N: 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 customers of DEFORM:

<http://support.deform.com/2d/support/fortran/>

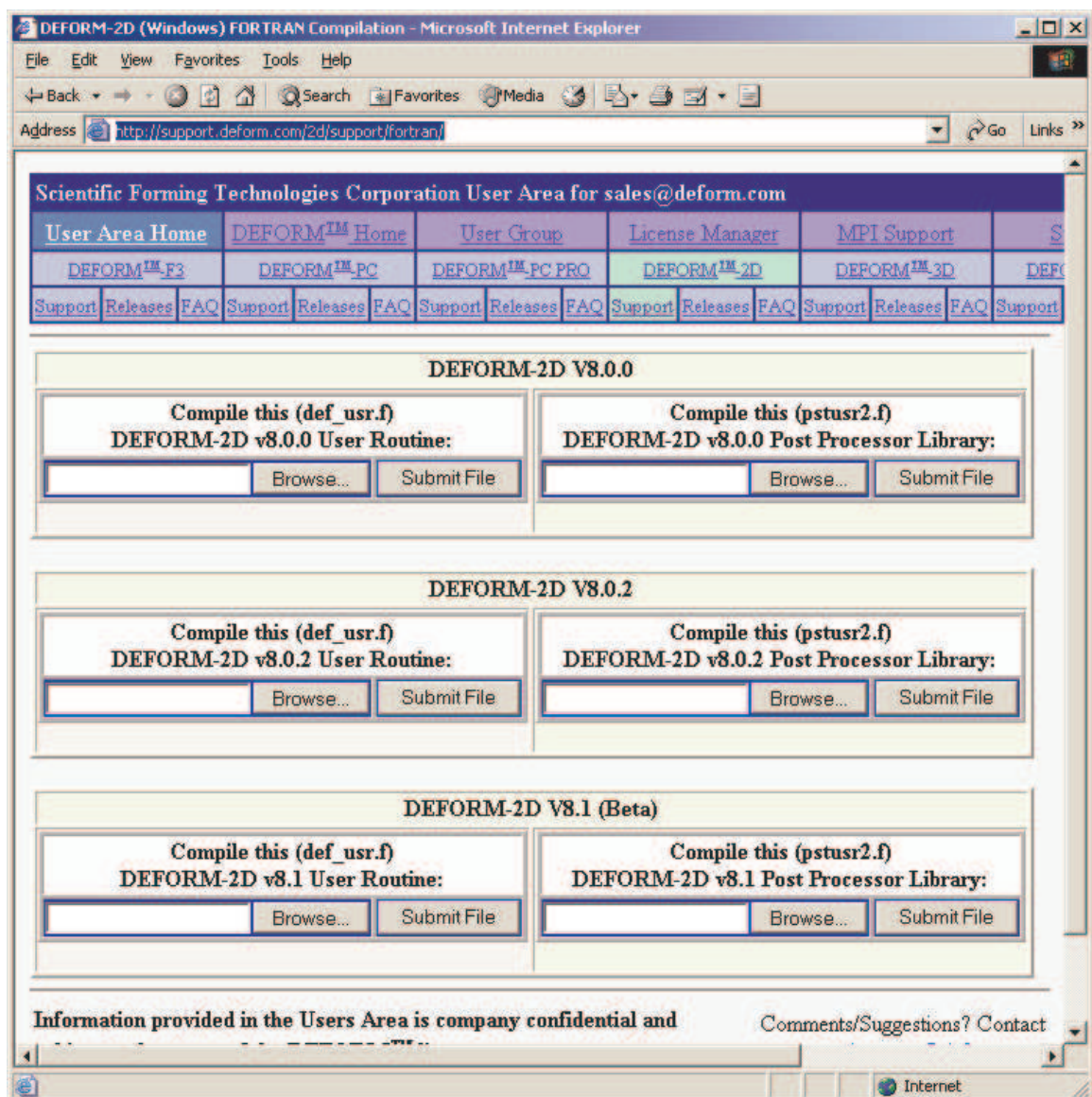


Figure 129: The DEFORM webpage for compiling user routines.

Figure 129 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 are 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.