SANDIA REPORT

Sandia National Laboratories

Sierra/SolidMechanics 5.8 Capabilities in Development

SIERRA Solid Mechanics Team
Computational Solid Mechanics and Structural Dynamics Department
Engineering Sciences Center

ABSTRACT

This user's guide documents capabilities in Sierra/SolidMechanics which remain "in-development" and thus are not tested and hardened to the standards of capabilities listed in Sierra/SM 5.8 User's Guide. Capabilities documented herein are available in Sierra/SM for experimental use only until their official release.

These capabilities include, but are not limited to, novel discretization approaches such as the conforming reproducing kernel (CRK) method, numerical fracture and failure modeling aids such as the extended finite element method (XFEM) and J-integral, explicit time step control techniques, dynamic mesh rebalancing, as well as a variety of new material models and finite element formulations.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

This page intentionally left blank.

1.   INTRODUCTION

This document is a user's guide for capabilities that are not considered mature but are available in Sierra/SolidMechanics (Sierra/SM) for early adopters. The determination of maturity of a capability is determined by many aspects: having regression and verification level testing, documentation of functionality and syntax, and usability are such considerations. Capabilities in this document are lacking in one or many of these aspects.

2. REPRESENTATIVE VOLUME ELEMENTS

This chapter describes the Representative Volume Element (RVE) capability, which is a multi-scale technique that uses a separate finite element model to represent the material response at a point.

The use of representative volume elements (RVEs) is a multi-scale technique in which the material response at element integration points in a reference mesh is computed using an RVE that is itself discretized with finite elements. RVEs are typically used to represent local, periodic material inhomogeneities such as fibers or random micro structures to avoid the requirement of a global mesh with elements small enough to capture local material phenomena.

In the current implementation of RVEs, periodic boundary conditions are applied to each RVE representing the deformation of a parent element and the stresses are computed in the elements of the RVE. These stresses are then volume-averaged over the RVE and the resulting homogenized stresses are passed back to the parent element.

This chapter explains how to use the RVE capability. Section 2.1 gives a detailed description of how RVEs are incorporated into an analysis. Details of the mesh requirements are delineated in Section 2.2 and the commands needed in an input file are described in Section 2.3.

> **Known Issue:** The capability to use RVEs with reference mesh multi integration point elements is still under development and should be used with caution.

2.1. RVE Processing

The use of the RVE capability requires two regions, each with its own mesh file. One region processes the reference mesh and the other processes all the RVEs. The commands used in the input file for the reference mesh region are the same as any other Sierra/SM region with the exception that a special RVE material model is used for every element block that uses an RVE. The RVE region is similar to an ordinary region. The only differences are that an RVE region has a line command for defining the RVEs' relationship to parent elements in the reference region and has restrictions on the use of boundary conditions.

The processing of an RVE essentially replaces the constitutive model of the parent element in the reference mesh. The steps followed at each iteration/time step of the reference mesh during an analysis using RVEs are as follows:

1. Internal force algorithm is called in the reference region to compute rate of deformation.

2. Each RVE gets the rate of deformation from its integration point on its parent element in the reference region.

3. The rate of deformation is applied to each RVE as a periodic boundary condition using prescribed velocity.

4. The RVE region is solved to obtain the stress in each element of each RVE.

5. The stresses in the elements of an RVE are volume-averaged over the RVE.

6. Each RVE passes its homogenized or volume-averaged stress tensor back to its integration point of its parent element in the reference mesh.

7. The reference region computes internal force again. Element blocks whose elements have associated RVEs do not compute a stress; they use the stress passed to them from their RVE.

2.2.   Mesh Requirements

Two mesh files, one for the reference region and one for the RVE region, are required for an RVE analysis. Figure 2-1 shows an example of the two meshes. The reference mesh of a bar with six single integration point elements is shown on the upper left. On the lower right is the mesh for the RVE region containing six RVEs, one for each element (since the elements have only one integration point) of the reference region. In this case, the first five RVEs each consist of two element blocks and the last RVE has four.
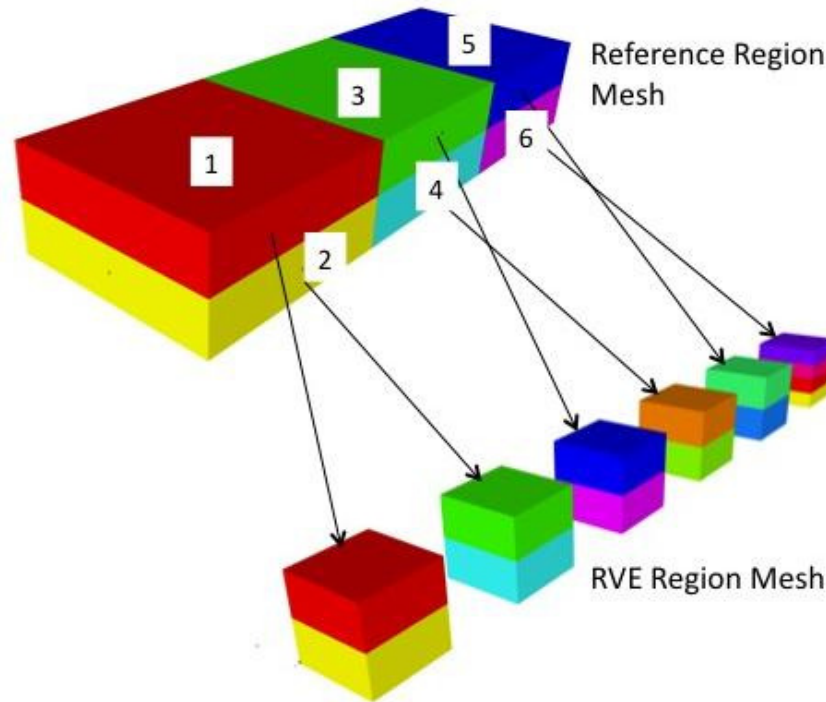
Figure 2-1. Example of meshes for RVE analysis

In general, each RVE should be a cube with any discretization the user desires. All RVEs must be aligned with the global x, y, and z axes. For stress computations, these axes are rotated into a local coordinate system, which can be specified on the reference mesh elements if these reference elements are uniform gradient hexahedra. In other words, if a local coordinate system is specified on a reference mesh uniform gradient element, the RVE global axes will be rotated internally in Sierra/SM to align with the local system on the associated parent element. The global X axis for an RVE is actually the local X' axis in the parent element.

Additional mesh requirements apply if the mesh does not match across opposing surfaces of the RVE. In this case, the RVE must include a block of membrane elements on the exterior surfaces with matching discretization on opposing surfaces (+x/-x, +y/-y, +z/-z). In order to minimize the effects of this membrane layer on the RVE response, it should be made as thin as possible. This membrane layer then must be tied to the underlying non matching RVE surfaces.

The RVE mesh must contain sidesets or node sets on each surface of every RVE. The RVE may be enclosed with one sideset that spans all six surfaces of the curb, or the user may specify

individual sidesets or node sets on each face. These sidesets/node sets are used to apply the periodic boundary conditions on the RVE. Sierra/SM generates the boundary conditions internally so the user does not have to include them in the input file. However, this assumes that the sidesets/node sets exist in the mesh file numbered in a specified order. If individual sidesets/node sets are used on each face of the RVE, the six sidesets/node sets must be numbered consecutively, starting with the positive-x face, followed by the negative-x face, positive-y face, negative-y face, positive-z face, and ending with the negative-z face. The beginning sideset id (for the positive-x face) is set by the user in the input file.

2.3.   Input Commands

There are several input commands that are relevant to RVEs. In the reference region, these commands include a special RVE material model and commands to define and use a local coordinate system along which an associated RVE will be aligned. In addition to the reference region, an RVE region is needed using the `BEGIN RVE REGION` command block. The RVE region command block uses the same nested commands as any other Sierra/SM region (with some restrictions as explained in this section) and an additional line command that relates the RVEs to their parent elements in the reference region.

2.3.1.   RVE Material Model

In an RVE analysis, any elements of the reference mesh that use an RVE must use the RVE material model. This model is defined similar to other material models as described in the Sierra/SM 5.8 User's Guide but uses the RVE keyword on the `BEGIN PARAMETERS FOR MODEL` command line as follows:

```
BEGIN MATERIAL <string>mat_name
  #
  DENSITY = <real>density_value
  #
  BEGIN PARAMETERS FOR MODEL RVE
    YOUNGS MODULUS = <real>youngs_modulus
    POISSONS RATIO = <real>poissons_ratio
  END PARAMETERS FOR MODEL RVE
  #
END [MATERIAL <string>mat_name]
```

Currently, the RVE material model tells the reference element not to perform a constitutive evaluation but to instead accept the stress tensor obtained from computation on an RVE. However, the use of an RVE material model still requires the input of Young's modulus and Poissons ratio. These values may be used for time step estimation and hourglass computations, even though they are not used in a constitutive evaluation.

Element blocks in the RVE region can use any material model that is supported in Sierra/SM other than RVE.

2.3.2.   Embedded Coordinate System

The finite element model of an element block in the reference mesh that uses RVEs can use an embedded coordinate system to orient the RVE relative to the reference element, if the reference elements are uniform gradient hexes. A coordinate system is defined in the sierra scope as described in the Sierra/SM 5.8 User's Guide. A local coordinate system is then associated with an element block through the use of a `COORDINATE SYSTEM` command line within a `BEGIN SOLID SECTION` command block.

16

```
BEGIN SOLID SECTION <string>section_name
  #
  COORDINATE SYSTEM = <string>coord_sys_name
  #
END [SOLID SECTION <string>section_name]
```

The string `coord_sys_name` must be a name associated in the input file with one of the
`COORDINATE SYSTEM` command blocks in the sierra scope. This coordinate system will then be
used on all elements of a block associated with a `BEGIN PARAMETERS FOR BLOCK` command
block that includes the command line specifying this solid section.

⚠️ **Known Issue:** Currently, the rotation of RVEs to a local element block coordinate
system only works with uniform gradient hexes in the reference mesh.

2.3.3.   RVE Region

A representative volume element (RVE) region must be a quasistatic region specified with the
RVE keyword in the `BEGIN RVE REGION` command line. The RVE region uses the same block
commands and line commands as any other quasistatic region with the addition of line commands
that define which element blocks of the reference region are associated with RVEs. There are also
some restrictions on boundary conditions as described in Section 2.3.6.

```
BEGIN RVE REGION <string>rve_region_name
   #
   # Definition of RVEs
   ELEMENTS <integer>elem_i:<integer>elem_j
     <integer>num_intg_pts_per_elem
     BLOCKS <integer>blk_i:<integer>blk_j
     SURFACE|NODESET <integer> start_id INCREMENT
                <integer> k
   #
   # Boundary Conditions
   #
   # Results Output Definition
   #
   # Solver Definition
   #
END [RVE REGION <string>rve_region_name]
```

2.3.4.   Definition of RVEs

One or more `ELEMENTS` command lines are used to associate elements of the reference region
mesh with RVEs in the RVE region. In the

17

```
ELEMENTS <integer>elem_i:<integer>elem_j
    <integer>num_intg_pts_per_elem
    BLOCKS <integer>blk_i:<integer>blk_j
    SURFACE|NODESET <integer>start_id INCREMENT
                        <integer>incr
```

command line, elements numbered `elem_i` through `elem_j` of the reference mesh and their `num_intg_pts_per_elem` integration points will be associated with RVEs (for a total number of RVEs equal to (`elem_j` - `elem_i` + 1) * `num_intg_pts_per_elem`), and each RVE will consist of `blk_i` - `blk_j` + 1 element blocks. Each integration point will be associated with a separate RVE. The block IDs of the first RVE must be `blk_i` through `blk_j` and subsequent RVEs (if `elem_j` is greater than `elem_i` or `num_intg_pts_per_elem` is greater than 1) must have consecutively increasing numbers for their block IDs.

Similarly `start_id` gives the `surface_id` of the first RVE if a single, encompassing surface is used, or the first `surface_id` or `nodelist_id` of the first RVE (the positive x surface as explained in Section 2.2) if six individual sidesets/nodeset are used. The remaining surfaces (nodesets) of the first RVE and all the surfaces of the following RVEs must be consecutively numbered following `start_id` in the mesh file as explained in Section 2.2.

The increment value `incr` indicates the number of sidesets present on the exterior of the RVEs. This is used to determine how to increment the IDs of the sidesets from one RVE to the next and to determine how to prescribe periodic boundary conditions on the RVE. The increment can have a value of either one or six. A value of one indicates that each RVE has one sideset that encompasses all six faces, while a value of six specifies that six sidesets or nodesets are present, one on each face. Nodesets are not allowed for the case where `incr` is one.

The following example shows the use of the `ELEMENTS` command line:

```
elements 1:5 1 blocks 1:2 surface 7 increment 6
elements 6:6 1 blocks 11:14 nodeset 15 increment 6
```

These commands generate the RVEs shown in Figure 2-1.

The first `ELEMENTS` command line specifies that elements with element IDs 1 through 5 in the parent region mesh each have one integration point and that each integration point has an RVE with two element blocks. The RVE associated with the integration point of element 1 of the parent region will have two element blocks starting with `block_id` of 1 and ending with a `block_id` of 2. Subsequent RVEs will have consecutively numbered element blocks. For example: the integration point of parent element 2 will be associated with an RVE that consists of element blocks 3 and 4 in the RVE region, the integration point of parent element 3 will be associated with the RVE that has element blocks 5 and 6, etc. Again, this is the case for the first five elements of the parent region mesh. The keyword `SURFACE` specifies that all the periodic boundary conditions generated by the code for the RVEs for elements 1 to 5 will use sidesets in the RVE region mesh. These sidesets will start with id 7 for the positive-x face of the RVE associated with parent element 1 and continue consecutively for the other faces of the RVE and the RVEs associated with the integration points of parent elements 2 through 5 (in the order specified in Section 2.2). In other words, the positive-x face of the RVE for parent element 1 is sideset 7, negative-x is sideset

18

8, positive-y is sideset 9, negative-y is sideset 10, positive-z is sideset 11, and negative-z is sideset 12. The sidesets for the RVE for parent element 2 will start with id 13 and continue consecutively in the same face order. The process continues for all five RVEs specified in this command line.

The second `ELEMENTS` line specifies that the integration point of element 6 of the parent region mesh will be associated with the RVE that consists of element blocks 11, 12, 13, and 14. The `NODESET` keyword says this RVE has a nodeset associated with each face of the RVE, starting with nodeset id 15 on the positive-x face, with id's increasing consecutively for the other five faces in the same order described in the paragraph above.

The six elements specified in these command lines must be in element blocks of the reference region mesh that use the RVE material model.

2.3.5.   Multi-Point Constraints

In the case in which the RVE has non matching surfaces, and therefore includes a block of membrane elements on the exterior surfaces, the user must specify a set of multi-point constraints (MPCs) to tie the membranes to the surface. This is done in the input file through use of an `MPC` command block:

```
RESOLVE MULTIPLE MPCS = ERROR
BEGIN TIED MPC
  TIED FACES = <string>membrane_surface_id
  TIED NODES = <string>RVE_surface_id
  SEARCH TOLERANCE = <real>tolerance
END
```

In this case, the `membrane_surface_id` corresponds to the single sideset that encompasses the membrane block is the side_a surface and the single sideset that encompasses the exterior surfaces of the RVE is the side_b surface. While the underlying RVE may have non matching exterior surfaces, the opposing surfaces of the membrane block must have matching discretizations. More detailed information on the use of MPCs, is discussed in the Sierra/SM 5.8 User's Guide.

2.3.6.   RVE Boundary Conditions

Strain rates computed by elements in the reference region are applied through periodic prescribed velocity boundary conditions on the faces of the associated RVEs. These are generated internally by Sierra/SM so the periodic boundary conditions do not need to be in the user's input file. However, because the RVE region is quasistatic, each of the RVEs must be fixed against rigid body motion. This must be done in the input file through use of the prescribed velocity boundary conditions:

```
BEGIN PRESCRIBED VELOCITY pres_vel_name
  NODE SET = <string>nodelist_name
  FUNCTION = <string>function_name
```

19

```
      SCALE FACTOR = <real>scale_factor
      COMPONENT = <string>X|Y|Z
   END [PRESCRIBED VELOCITY pres_vel_name]
```

This type of boundary condition is described in detail in the Sierra/SM 5.8 User's Guide but the use for RVEs is restricted- Either the function must always evaluate to 0.0 or the `scale_factor` must have a value of 0. This is essentially a way of using the prescribed velocity boundary condition to fix the nodes in `nodelist_name`. However, in order for these conditions to work with the periodic boundary conditions used to apply the strain rate, `PRESCRIBED VELOCITY` must be used rather than `FIXED DISPLACEMENT` or `PRESCRIBED DISPLACEMENT` boundary conditions.

Generally, three `BEGIN PRESCRIBED VELOCITY` command blocks will be needed, one each for X, Y, and Z components. In order to eliminate rigid body motion without over constraining the motion, each `BEGIN PRESCRIBED VELOCITY` block should constrain exactly one node of an RVE in one component direction. (However, `nodelist_name` may contain nodes from multiple RVEs. Separate boundary condition blocks are not required for each RVE.). To prevent rigid body rotations, the three constrained nodes on each RVE should not be collinear.

3. EXPLICIT SUBCYCLING

This chapter describes how to setup an analysis to use explicit subcycling. Subcycling can be used to run different parts of the mesh at different time step sizes to improve speed.

Warning: Explicit subcycling is a capability still in the development stages. This capability is not yet recommended for general use.

Explicit subcycling can be used in an explicit transient dynamics analysis to run one part of the mesh at a small time step while running another connected part of the mesh at a large time step. Explicit subcycling can provide a substantial model speedup only if two properties hold. First, some region of the mesh must have a substantially smaller element critical time step than another region of the mesh. Second, the portion of the mesh with the small critical time step must contain a small fraction of the total number of elements used by the analysis.

Explicit subcycling divides the analysis domain into two regions: A coarse region iterating with a large time step and a fine region iterating at a smaller time step that is some integer fraction of the coarse time step. At the coarse mesh time step, both regions sync up to the same analysis time and exchange information. Using the standard analysis technique, every element must run at the same small time step. Testing has shown that an analysis run using subcycling can give equally accurate results as an analysis run without subcycling. The accuracy of the simulation is subject to several restrictions on cross region communication and compatible capabilities.

3.1. Specifying Subcycling in Input

The recommended method to turn on subcycling is to use a feature to automatically generate the coarse and fine regions in the input deck. This is done by adding the following command to the presto region.

```
SUBCYCLE BLOCKS = <string list>block_names
```

If this command is present, Sierra/SMwill automatically generate and run a new input deck that can be used for the subcycling. If the original input deck is named `input.i` the automatically generated subcycling input deck will be named `input.subcycle.i`. The block_names specified are the names of the blocks targeted for inclusion in the fine region (run with the small time step).

The algorithm to split the regions is done as follows.

1. Define the trial coarse region based off of everything not in the fine region.

2. Compute the critical time step of the coarse region as the smallest element time step in that region.
3. Compute the maximum time step each node can be integrated at as the smallest time step of any element near the node.
4. For every element in the mesh, if the element is attached to only nodes with time steps greater than or equal to the coarse region time step, place the element in the coarse region.
5. For every element in the mesh, if the element is attached to any node with time steps less than the coarse region time step, place the element in the fine region.
6. Split all boundary conditions defined on the coarse and fine region appropriately and write to appropriate results files. Each region will generate independent output files.

The mathematical foundations of the subcycling algorithm used in Sierra/SMcan be found in Reference [1]. The portions of this paper specifically used in Sierra/SMare: Explicit-Explicit, Central Difference, and Linear Interpolation method.

3.2.　　　Limitations of Subcycling

Subcycling is currently incompatible either in whole or in part with many other capabilities. The capabilities that have incompatibility with subcycling include but may not be limited to the following:

- Subcycling is incompatible with most capabilities that require an auxiliary region. This include representative volume elements (RVE), Gemini coupling, and multi-procedure analysis coupled via hand-offs or solution control.
- Subcycling currently does not work with implicit dynamics, implicit statics, or modal analysis.
- Subcycling is currently not compatible with rigid bodies.
- Subcycling is incompatible with any critical time step computation method other than the default element based time step calculation. This includes nodal based and Lanczos algorithm based time step computation methods.

Additionally several capabilities will not function correctly if that capability is operating at or near the boundary between the coarse and fine region. If such a capability is included in the subcycling analysis and that capability happens to cross the coarse/fine boundary, accuracy and stability problems may result. The capabilities known to be have restrictions when used with subcycling include but may not be limited to:

- Element death near the subcycling boundary may not be able to correctly determine when a node shared between the two regions goes inactive (leading to accuracy and stability issues).
- Contact between any surface in the fine region and any surface in the coarse region cannot be evaluated.
- Methods that define a force from an external load (such as CTH) can only be coupled to the deformation of the coarse.
- No non-local element or boundary condition can span the coarse to the fine boundary. This includes nodal based tetrahedra, MPCs, Spot Welds, Super Elements, Cylindrical Joints, and the J-Integral computation.
- Nodal output quantities at the coarse to fine boundary may not be displayed properly in plot files. Contributions to quantities such as nodal force may exist in both the fine and coarse region and the outputs would need to be summed from both.

3.3. **Other Subcycling Issues**

In parallel, subcycling will perform best if a mesh rebalance is performed to ensure both the fine and coarse regions are divided evenly among the processor sets. A mesh rebalance command similar to the one below can be used to automatically performance such a mesh rebalance. See the Sierra/SM 5.8 User's Guide Section 6.8 for more information on mesh rebalancing.

```
BEGIN REBALANCE
     PERIODIC REBALANCE = AUTO
     DELETE DEACTIVATED ELEMENTS AFTER REBALANCE = ON
END
```

References

[1] Willem E.M. Bruijs. *Subcyling in Transient Finite Element Analysi*. PhD thesis, Technical University of Eindhoven, 1234.

4.   AUTOMATIC TIME STEP SELECTOR

For performance reasons, it is sometimes desired to run at the highest possible time step in explicit dynamics. The `NODE BASED TIME STEP` and the `LANCZOS TIME STEP` have proven to yield a higher time step than the default element time step for most problems. However, because these routines take significantly more time to calculate, sometimes the performance benefits are unseen. The automatic time step selector attempts to weigh the performance benefits of each time step calculation. The time steps are compared every hundred steps, and the one proving most beneficial is used for the proceeding hundred steps.

Currently, only the node based time step and the element time step are compared. Because the node based time step takes longer to calculate than the element based time step (as noted above), a scale factor is used when comparing the two. To run with the automatic time step selector, the following must be included in the input file:

In the `BEGIN PARAMETERS FOR PRESTO REGION` block, the following line must be included:

```
BEGIN PARAMETERS FOR PRESTO REGION <string>presto_region
  TIME STEP SELECTOR = AUTO
END PARAMETERS FOR PRESTO REGION <string>presto_region
```

Additionally, the node based time step syntax must also be included in the input file:

```
BEGIN NODE BASED TIME STEP PARAMETERS <string>name
END NODE BASED TIME STEP PARAMETERS <string>name
```

4.1.   Explicit Quasistatic Mode

```
BEGIN TIME CONTROL
  BEGIN TIME STEPPING BLOCK <string>time_block_name
    START TIME = <real>start_time_value
    BEGIN PARAMETERS FOR PRESTO REGION <string>region_name
      NUMBER OF QUASISTATIC TIME STEPS =
        <int>quasi_step_count
    END [PARAMETERS FOR PRESTO REGION <string>region_name]
  END [TIME STEPPING BLOCK <string>time_block_name]
  TERMINATION TIME = <real>termination_time
END [TIME CONTROL]
```

Explicit quasistatic mode provides an automated methodology to leverage explicit time stepping to efficiently solve quasistatic problems. Quasistatic mode is enabled by providing a non-zero input for the `NUMBER OF QUASISTATIC TIME STEPS` command for a solution time period. Explicit solution of a quasistatic analysis may prove to be more robust or faster than implicit solution in some cases.

Explicit quasistatic mode can be particularly helpful for analyses with contact. Analyses with several disjoint parts that are in static equilibrium only due to the contact and frictional forces between those may cause great difficulty for the standard implicit static solver. Explicit quasistatic mode can often solve these problems more robustly than the implicit solver. Additionally explicit quasistatic mode is well suited to analyses that involve buckling, strong material nonlinearity, and material failure or fracture. Such sharp nonlinearity cause substantial difficulty for standard implicit solution.

Activating explicit quasistatic mode has several effects on the analysis.

- Implicit compatible defaults are used for element formulations for numerical parameters such as effective modulus, quadratic bulk viscosity, and linear bulk viscosity.
- The analysis is automatically mass scaled (See Sierra/SM 5.8 User's Guide Section 3.3) so that `quasi_step_count` steps will be taken in the current time period. As dynamic terms still have some meaning in explicit quasistatic analysis to function efficiently the size of the implied quasistatic time step should be significantly larger than the standard explicit time step. For example if the standard explicit time step is 1.0e-6 seconds and it is desired the load up quasi-statically over 100,000 steps then the time over which to perform this loading should be one second or greater. This would yield a quasistatic time step of 1.0e-5 seconds which is significantly larger than the standard explicit time step.
- To minimize dynamic effects explicit quasistatic analysis should be run at a time step that exceeds the explicit time step. Sometimes it is convenient to greatly exceed the standard explicit time step. For stability reasons the time step is slowly increased from the standard explicit time step when entering a quasistatic solution period and slowly decreased to the standard explicit time step when exiting a quasistatic solution period.
- A dynamic viscous damping term is included to accelerate convergence to a static equilibrium solution. This viscous term is automatically tuned based on the number of time steps being taken and the current model velocity.

- A static residual norm is automatically computed and output to the global variable `quasistatic_residual`. Monitoring of this residual can indicate how closely the current solution approximates the target quasistatic state. The quasistatic residual is the L_2 norm of the current nodal force imbalance divided by the reference force (reaction or external force sum). The explicit quasistatic residual has the same meaning as the implicit relative residual, generally a good solution will have a quasistatic residual of 10^{-3} or below.

4.1.0.1. Usage Guidelines

It is important to note that quasistatic mode is a *dynamic* analysis; dynamic terms and effects will be present in the solution. However, if used properly, quasistatic mode will minimize these effects, yielding a solution with comparable accuracy to an equivalent implicit static analysis. Generally, the number of quasistatic explicit steps required to reach static equilibrium will be tens to hundreds of thousands, although a sharp estimate of the actual number of steps required is not available. Generally, dynamic effects are expected to reduce with increasing numbers of time steps. The quasistatic residual norm variable may be used to monitor static equilibrium at various steps of the analysis.

Quasistatic equilibrium is reached fastest on models with minimal potential low-mode vibrations. The most difficult types of analyses for quasistatic mode to solve include long, slender structural members that can vibrate at low frequencies.

Static equilibrium in explicit quasistatic mode is also sensitive to *loading rate*; more smoothly applied external loads will result in more rapid convergence to the quasistatic solution. It is also recommended that loads be held constant in the later part of the loading period. Ultimate quasistatic equilibrium can be reached in the contact load. An example showing recommended quasistatic mode loading is shown in Section 4.1.0.2.

Output results from quasistatic mode—particularly quantities such as reaction force and energy—are valid only when the model has reached static equilibrium. The mass scaling used by quasistatic mode can alter the results in the middle of a load step. Damping and other artificial forces may be arbitrarily large when the model is in motion in an intermediate state, while they tend to zero when a model reaches static equilibrium.

It is assumed that the ultimate material state obtained is strain path-independent as long as the strain path is monotonic. This condition holds for elastic models, J_2 plasticity models, soil foam, and most other commonly used models, but does *not* hold for rate-dependent models. Quasistatic mode will provide an answer for rate-dependent models, but a very large number of load steps may be required to ensure the loading is applied smoothly.

Although the smoothness of the strain paths and reduction of dynamic effects will improve with iterations, the most reliable way to detect possible material state overshoot error is to compare the result with a solution based on a larger number of quasistatic iterations. Current evidence indicates quasistatic material state integration errors reduce quadratically in iteration count.

4.1.0.2. Example Problem

The following input demonstrates the use of explicit quasistatic mode for a beam loaded by a pressure boundary condition.

```
begin function pfunc
  evaluate expression = cos_ramp(t, 0.0, 0.5)
end

begin time control
  begin time stepping block p1
    start time = 0.0
    begin parameters for presto region presto
      number of quasistatic time steps = 2000
    end
  end
  termination time = 1.0
end

begin presto region
  begin pressure
    surface = surface_1000
    function = pfunc
  end
end
```

5.   MODAL ANALYSIS

This chapter describes a simple modal analysis capability. This capability will compute the lowest few vibration eigenmodes and values at the end of each model load step. This capability only works with solid uniform gradient hex elements and augmented Lagrange tied contact.

5.1.   Modal Analysis

> **Warning:** Modal analysis is still a capability in the early development stages. This capability is not recommended for general use, nor will any use of this capability currently be supported by the Sierra/SMdevelopment.

```
BEGIN LANCZOS EIGEN SOLVER
  MASS MATRIX = IDENTITY|LUMPED(LUMPED)
  NUMBER OF EIGENPAIRS = <integer>N
  DEBUG = OFF|ON(OFF)
END
```

The command `NUMBER OF EIGENPAIRS` defines the number of eigenvalues and modes to compute. The lowest `N` modes will be computed. Significant expense is required to compute and store each mode, thus `N` should be kept relatively small (no more than 25 or so).

The `DEBUG` command turns on or off additional debugging outputs from the eigensolver.

The `MASS MATRIX` allows the user to selectively compute the eigenvalues of the tangent stiffness matrix when `IDENTITY` is set. Otherwise, the standard eigenvalue problem is computed with both the tangent stiffness matrix and the lumped mass matrix. A consistent mass matrix is not available at this time.

6. SOLVERS AND SOLVER OPTIONS

6.1. Newton Solver

```
BEGIN SOLVER
  BEGIN NEWTON
      #
      # convergence criteria commands
      #
      TARGET RESIDUAL = <real>target_resid
        [DURING <string list>period_names]
      TARGET RELATIVE RESIDUAL = <real>target_rel_resid(1.0e-4)
        [DURING <string list>period_names]
      ACCEPTABLE RESIDUAL = <real>accept_resid
        [DURING <string list>period_names]
      ACCEPTABLE RELATIVE RESIDUAL = <real>accept_rel_resid
        [DURING <string list>period_names]
      REFERENCE = EXTERNAL|INTERNAL|BELYTSCHKO|RESIDUAL|ENERGY
        (EXTERNAL) [DURING <string list>period_names]
      RESIDUAL NORM TYPE = ALL|TRANSLATION|SCALE_RB_ROTATIONS
        (ALL) [DURING <string list>period_names]
      #
      # iteration control
      #
      MINIMUM ITERATIONS = <integer>min_iter(0)
        [DURING <string list>period_names]
      MAXIMUM ITERATIONS = <integer>max_iter
        [DURING <string list>period_names]
      #
      # Selection of the linear solver for use in solving
      # linearized Newton iterations
      #
      LINEAR SOLVER = <string>linear_solver_name
  END
END
```

The Newton solver is an nonlinear equation solver that is an alternative to the default conjugate gradient (CG) solver. Each iteration of the Newton solver consists of reforming current tangent stiffness matrix and re-solving the equation set with that current tangent. The Newton solver is typically significantly more expensive than the CG solver but may be more robust if there is substantial nonlinearity occurring over a time step. The Newton solver may also potentially

prevent overshooting of yielding or other material nonlinearity. If a model has highly nonlinear materials and is failing to converge with the CG solver the Newton solver may be worth trying.

The convergence criteria and iteration control commands in the Newton solver behave identically to the equivalent commands in the CG solver. The `LINEAR SOLVER` specifies the solver to use during the linearized equation solution step each Newton iteration. The `FETI` solver is recommended but any available linear may work.

6.2.   Control Contact : Control Subset

```
BEGIN CONTROL CONTACT
   CONTROL SUBSET = ADAGIO|ALL|ARS|JAS|SST
END
```

By default all implicit contact constraints are enforced simultaneously. The `CONTROL SUBSET` option to the control contact solver block is an experimental option for enforcing different types of constraints at different levels of the multilevel solver. For example the following input will control the node face (ADAGIO) contact constraints at level one and the analytic rigid surface contact constraints (ARS) at level two. This means that ARS constraints are found held constant while ADAGIO constraints are iteratively solved. Then the ARS constraints are updated and again held constant while the ADAGIO constraints are iteratively solved again.

```
begin control contact control_al
  level = 1
  target relative residual = 5.e-04
  control subset = ADAGIO
end control contact

begin control contact control_ars
  level = 2
  target relative residual = 1.e-3
  control subset = ARS
end control contact
```

Use of the `CONTROL SUBSET` will substantially increase analysis cost, but may lead to more robust convergence if the model contains potentially conflicting contact constraint types acting on the same nodes.

7. EXTENDED FINITE ELEMENT METHOD (XFEM)

⚠️ **Warning:** This capability is in development, and its behavior may change considerably due to its status as an active research topic.

The `XFEM` command block may be used to introduce discontinuities in a finite element mesh via the eXtended Finite Element Method (XFEM). Use cases for XFEM include modeling stationary or propagating cracks in a finite element mesh, fast mesh generation via XFEM "carving," and adding or removing material layers to simulate, e.g., material wear or additive manufacturing processes. At its simplest, the XFEM provides a framework supporting duplication of mesh elements and subsequent partitioning and assignation of material on each side of the cut surface to each duplicate. This duplication procedure is illustrated in Figure 7-1. Piecewise planar element cuts through both two-dimensional shell and three-dimensional mesh topologies are supported in the current XFEM implementation. When an element is cut, the necessary quantities on the duplicated elements are scaled by the volume fraction of the original cut element. The mass, volume, and the internal force contribution are all scaled by the volume fraction. All other element quantities are calculated as usual.
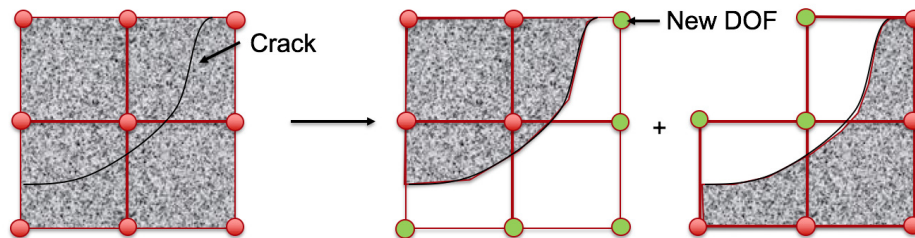
Figure 7-1. Example of XFEM element cutting and duplication.

The effective or cut volume of the domain is represented by the XFEM "submesh," a sub-element geometry which captures the discontinuity surface within each cut element duplicate. Submesh topologies for various element types are illustrated in Figure 7-2. The submesh output block, named `<block_name>_submesh`, will be created and output along with results for visualization purposes. Visualization with the submesh block is recommended as it offers an accurate representation of crack surface and fragment geometries, as well as relevant element and nodal fields, whereas the XFEM computational elements themselves overlap and are therefore difficult to visualize.

(a) Four-node quadrilateral

(b) Four-node tetrahedral

(c) Eight-node hexahedral
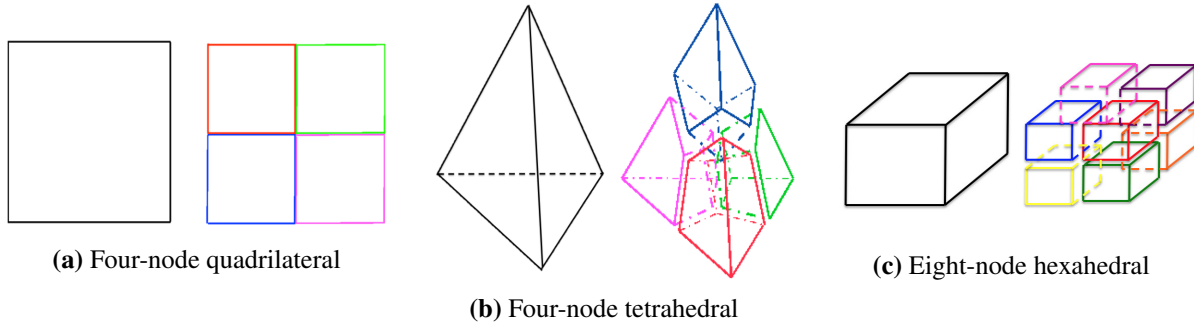
Figure 7-2. Illustration of XFEM submesh topology for various mesh element topologies.

7.1.   General XFEM Commands

```
BEGIN XFEM <string>xfem_name
  BLOCK = <string list>block_name
  ASSEMBLY = <string list>assembly_name
  INCLUDE ALL BLOCKS
  ADD INFINITE PLANE = <real>px <real>py <real>pz
          <real>nx <real>ny <real>nz
  ADD DISC = <real>px <real>py <real>pz
          <real>nx <real>ny <real>nz
          <string>radius_function
  MECHANICS GROWTH START TIME = <real>time(0.0)
  MECHANICS GROWTH METHOD = <string>NOTHING|
          MECHANICS FAILURE(NOTHING)
  CRITERION = <string>{AVG NODAL|MAX NODAL|
          MIN NODAL|ELEMENT|GLOBAL}
          VALUE OF <string>variable
          {>=|>|=|<|<=} <real>threshold
  FAILURE SURFACE EVOLUTION = <string>PLANAR|PIECEWISE LINEAR|
          SINGLE CRACK(PLANAR)
  ANGLE CHANGE = <string>NONE|STRESS EIGENVECTOR|
          ONE RING|LENGTH SCALE(NONE)
  CREATE FACES = <string>ON|OFF(ON)
  GENERATION BY NUCLEATION = <string>NO|ELEMENT-BASED(NO)
  NUCLEATION CRITERION = <string>
          {AVG NODAL|MAX NODAL| MIN NODAL|ELEMENT|
          GLOBAL} VALUE OF <string>variable
          {>=|>|=|<|<=} <real>threshold
  CRACK BRANCHING = <string> RESTRICTED|ALLOWED(RESTRICTED)
  BRANCHING CRITERION = <string>
          {AVG NODAL|MAX NODAL| MIN NODAL|ELEMENT|
          GLOBAL} VALUE OF <string>variable
          {>=|>|=|<|<=} <real>threshold
  PROPAGATION ANGLE LIMIT = <real> angle
```

```
  ANGLE CHANGE LENGTH SCALE OUTER RADIUS = <real>outer_radius
  ANGLE CHANGE LENGTH SCALE INNER RADIUS = <real>inner_radius
  START TIME = <real>start_time
  INITIAL SURFACE COHESIVE = <string>FALSE|TRUE(FALSE)
  COHESIVE SECTION = <string>cohesive_section_name
  COHESIVE MATERIAL = <string>cohesive_material_name
  COHESIVE MODEL = <string>cohesive_model_name
  SOLID GROWTH DIRECTION VARIABLE =
          <string>direction_field_name(stress)
  SHELL GROWTH DIRECTION VARIABLE =
          <string>direction_field_name(memb_stress)
  VOLUME FRACTION LOWER BOUND = <real>lower_bound(0.0) DELETE|
          RETAIN(DELETE)
  CALCULATE FRAGMENT IDS = OFF|ON(OFF)
  INITIAL CUT WITH {SIDESET|STL}
          <string>file_or_surface_name
          REMOVE {INTERIOR|EXTERIOR|NOTHING(NOTHING)}
  CUT WITH DAMAGE VARIABLE = <string>variable_name
END [XFEM <string>xfem_name]
```

Assemblies may contain blocks, or assemblies of these.

7.2. XFEM for Fracture and Fragmentation

The most common application of XFEM is modeling of fracture, fragmentation, and failure in structures. Currently supported fracture capabilities are

- prescribed, static or stationary cracks,
- prescribed cracks with a specified direction and rate of growth,
- prescribed cracks which are allowed to propagate by mechanics-based growth criteria, and
- cracks which are nucleated and propagated via mechanics-based criteria.

These capabilities are detailed below.

7.2.1. Fixed and Prescribed XFEM Discontinuities

A "fixed" XFEM discontinuity is stationary in both time and space; the failure surface does not change after initialization. A fixed infinite plane discontinuity can be inserted via the `ADD INFINITE PLANE` command, while a disc-shaped cut with a fixed radius may be inserted via the `ADD DISC` command. Note that the specified surfaces are used to cut the mesh in the reference configuration.

A "prescribed" XFEM discontinuity is restricted to propagate along a specific path in time. In order to prescribe an XFEM discontinuity, a disc must be inserted via the `ADD DISC` command. The discontinuity may "grow" by adding a time-varying function at the end of the `ADD DISC` command or by mechanics growth, described in Section 7.2.2 below.

7.2.2. Spontaneous Crack Nucleation, Growth, and Branching

The current XFEM implementation enables the natural evolution of fractures in materials based on mechanics nucleation, growth, and branching criteria.

Crack growth Growth, or propagation, can be enabled via the following command lines:

```
MECHANICS GROWTH METHOD = MECHANICS FAILURE
CRITERION = <string>{AVG NODAL|MAX NODAL|
        MIN NODAL|ELEMENT|GLOBAL}
        VALUE OF <string>variable
        {>=|>|=|<|<=} <real>threshold
FAILURE SURFACE EVOLUTION = PLANAR|PIECEWISE LINEAR|
        SINGLE CRACK(PLANAR)
```

The `CRITERION` command line specifies the criterion for propagation or growth of the crack from element to element. This command is precisely analogous to element death; refer to the Sierra/SM 5.8 User's Guide Chapter 6 for additional details. `FAILURE SURFACE EVOLUTION` specifies any geometric restrictions on fracture growth:

- `PLANAR` is the default option, which restricts the crack to grow only in the plane in which it is initialized, preventing the crack from turning or twisting.

37

- `PIECEWISE LINEAR` allows a crack to change directions such that it is planar within a single element; however, this option may lead to a fracture surface which is discontinuous from element to element.

Mechanics growth can be delayed in the analysis by specifying a start time (≥ 0) in the `MECHANICS GROWTH START TIME` command.

The way in which the crack growth angle change is computed can be specified via the `ANGLE CHANGE` command line to smooth or regularize sharply varying stress fields in the neighborhood of crack fronts. Available angle change options are

- `STRESS EIGENVECTOR`, which calculates the growth angle of the crack from the maximum principal stress eigenvector in the element to be cut;
- `ONE RING`, which defines the new failure plane by the maximum principal stress eigenvector of the *average* stress in the node-connected neighboring elements (or one-ring) of the element to be cut; and
- `LENGTH SCALE`, which computes the crack failure plane as the maximum principal stress eigenvector of the average stress in elements within a specified radial distance of the element to be cut. This distance can be specified via the `ANGLE CHANGE LENGTH SCALE OUTER RADIUS` command. By specifying `ANGLE CHANGE LENGTH SCALE INNER RADIUS`, in addition to including elements inside a given outer length scale, the growth algorithm will *exclude* elements within a given inner radius of the crack front from the direction computation. Because the length scale entails a computation involving, in general, a number of elements surrounding the crack front, this option may incur significant additional simulation time within in each load step.

The variable used to calculate the angle change can be specified via

```
SOLID GROWTH DIRECTION VARIABLE = ...
SHELL GROWTH DIRECTION VARIABLE = ...
```

for solid and shell elements, respectively. The default variable used for solid elements is "stress," while the default variable used for shell elements is "membrane stress."

Crack nucleation   Spontaneous nucleation, or initiation, of cracks may be controlled by the command lines

```
GENERATION BY NUCLEATION = <string>NO|ELEMENT-BASED(NO)
NUCLEATION CRITERION = ...
```

Currently, only element-based nucleation is supported, in which a single element is cut if it exceeds the user-defined nucleation criterion (which follows the same form as the growth criterion). Nucleated cracks then grow normally according to the specified mechanics growth criterion.

Crack branching   Branching behavior may also be modeled via the commands

```
CRACK BRANCHING = ALLOWED
BRANCHING CRITERION = ...
```

Currently, cracks may only branch from a single point on an element edge (i.e., from a virtual node on the element edge created by the first cut). Examples of eligible and ineligible branching locations are illustrated in Figure 7-3. All presently cut elements are branching candidates. The user-defined failure condition is examined for each element, and if the value exceeds the failure criteria, the stress eigenvectors are calculated and used to determine the possible branching direction.
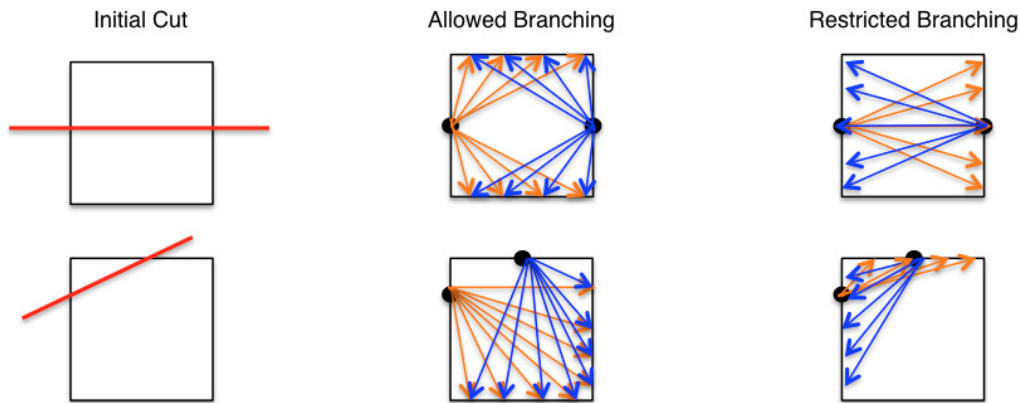
Figure 7-3. Example of allowed and restricted branching.

7.2.3.   *Cohesive Zone Insertion*

Cohesive zones can be adaptively inserted between the XFEM discontinuities in order to better capture fracture patterns, convergence, and energy dissipation. To insert cohesive zones with XFEM,

- a cohesive section must be specified in the XFEM command block via the `COHESIVE SECTION` command line,
- a cohesive material must be specified via the `COHESIVE MATERIAL` command line, and
- a cohesive model must be specified via the `COHESIVE MODEL` command line.

In order for the cohesive zones to be inserted with the stress initialized to that of the failing element, the `INITIAL SURFACE COHESIVE = TRUE` option must be used.

⚠️   **Warning:** Cohesive zone insertion for tetrahedral elements is not yet supported.

7.2.4.   *Other Options*

Several miscellaneous or experimental XFEM capabilities are available for fracture and fragmentation analysis.

39

Volume Fraction Lower Bound　By default, the XFEM implementation in Sierra does not "clip" or remove elements with arbitrarily small volume fractions. This can create issues with the conditioning of implicit solves.

The VOLUME FRACTION LOWER BOUND command allows the user to specify a threshold. By default, when a lower bound is provided with this command, elements whose volume fractions are below the specified threshold will be removed from the calculation (DELETE). When the RETAIN option is specified, elements whose volume fractions are below this specified threshold will be retained, but have their volume fractions are reset to the lower bound specified by the threshold value. This insures that the smallest volume fraction of any partial element anywhere in the domain will not be smaller than the threshold.

> ⚠️ **Warning:** The VOLUME FRACTION LOWER BOUND can result in the loss of mass conservation for an embedded object, whether in the default mode when these small volume fractions are removed or in the RETAIN mode when mass is added.

XFEM damage-based failure　XFEM can also be used to cut the mesh along a specific field on the mesh (such as a phase field damage variable). The name of this variable is specified via the CUT WITH DAMAGE VARIABLE command.

> ⚠️ **Warning:** The CUT WITH DAMAGE VARIABLE option is in-development and not a hardened capability.

Identification of separate XFEM fragments　The CALCULATE FRAGMENT IDS command can be used to output both element and nodal fragment ID fields. Turning this option to ON will set both the element variable called element_fragment_id as well as the nodal variable called node_fragment_id at the end of the simulation. Each ID corresponds to a distinct fragment from the XFEM simulation. Elements and nodes within a fragment will all be assigned the same fragment ID. Labeling of the fragment IDs is arbitrary, but the numbering always begins with 1 and goes to the total number of fragments in the simulation. Post-processing scripts can be use in conjunction with these fields to compute quantities such as fragment mass and momentum distributions.

7.3.　　　XFEM Carving

In addition to modeling fracture and fragmentation, XFEM can also be used for fast mesh generation or wearing of surfaces via "carving." The carving procedure is roughly equivalent to an immersed boundary approach; boundary and contact surfaces are represented by the XFEM cut surface, and the effective carved element response is computed via XFEM volume fraction scaling.

The initial mesh may be carved with the command line

```
INITIAL CUT WITH {SIDESET|STL} = <string>file_or_surface_name
   [REMOVE {INTERIOR|EXTERIOR|NOTHING(NOTHING)}]
```

where `STL` indicates to carve the mesh with a stereolithography (STL) file [1], while the `SIDESET` option indicates to carve with a specified sideset from the input mesh file.

Carved material may be removed after the cut is made via the option
`REMOVE {INTERIOR|EXTERIOR}`, where the "exterior" consists of all material points lying outside of the region bounded by the carving surface in the direction of its outward normal vector; similarly, the "interior" is the region bounded by the carving surface in the direction opposite its outward normal. As an example, the XFEM command block to cut all blocks with a surface defined in an STL file `file.stl` and remove material interior to the surface is the following:

```
BEGIN XFEM
  INCLUDE ALL BLOCKS
  INITIAL CUT WITH STL file.stl REMOVE INTERIOR
END [XFEM]
```

7.4. Use of XFEM with Existing Capabilities

An `XFEM` command block may be used in conjunction with a number of other core code capabilities, as enumerated in [2]. A brief list of compatible capabilities and usage guidelines are given below.

7.4.1. *Contact*

```
BEGIN CONTACT DEFINITION <string>name
  ...
  CONTACT SURFACE <string>name CONTAINS
        <string_block_name>_CONTACT_SURFACE
  BEGIN INTERACTION DEFAULTS
    SELF CONTACT = ON
    GENERAL CONTACT = ON
  END
  ...
END [CONTACT DEFINITION <string> name]
```

Contact may be enforced on a block that has been cut using XFEM, including the cut surface itself. Contact can be defined using the `GENERAL CONTACT = ON` command within the `INTERACTION DEFAULTS` section of the contact definition. A contact surface called `<string_block_name>_CONTACT_SURFACE` is created for each XFEM block; thus, a contact surface may be defined on an XFEM block by using the `CONTACT SURFACE <string>name CONTAINS <string_block_name>_CONTACT_SURFACE` command line. The XFEM contact surface is also output to the results file *as a shell element block* for visualization purposes.

7.4.2. *CONWEP Blast Pressure*

```
BEGIN BLAST PRESSURE <string> name
  BLOCK = <string_block_name>_submesh
  ASSEMBLY = <string_assembly_name>_submesh
  ...
END [BLAST PRESSURE <string> name]
```

XFEM can also be used in conjunction with a CONWEP blast pressure. The pressures that are applied to the cut faces are scaled by the area fraction of that cut face. The pressures are applied to the face throughout the duration of the blast. Assemblies may contain blocks, surfaces, or assemblies of these.

7.4.3. *Implicit Dynamics*

XFEM may be run in implicit dynamics. If an implicit simulation is run, it is highly recommended to include an adaptive time stepping block, as shown below. Adaptive time

stepping helps to account for the increased complexity of the problem during crack growth. For additional guidance and command syntax, consult the Sierra/SM 5.8 User's Guide Chapter 4.

Warning: Convergence of XFEM simulations in implicit dynamics mode is currently tenuous; robustness issues may occur when using this analysis combination.

References

[1] 3D Systems, Inc. SLC File Specification, 1994.

[2] Sierra/SolidMechanics Team. Sierra/SolidMechanics VOTD User's Guide. Technical Report In draft, Sandia National Laboratories, Albuquerque, NM, 2019.

8.   EXPLICIT CONTROL MODES

8.1.   Limitations and Requirements

In explicit dynamic calculations, the Explicit Control Modes method uses a coarse mesh overlaying the actual problem mesh (called the reference or fine mesh) to increase the critical time step. The name *control mode* comes from the implicit multigrid solution algorithm in Adagio. The Explicit Control Modes algorithm is discussed in [1] and [2].

Warning: Explicit Control Modes is an experimental analysis technique. It has been shown to be an extremely useful technique on specific problems. However, it does not interoperate with some features, such as rigid bodies. Contact the Sierra/SM development team for more information on the features that do cooperate with Explicit Control Modes.

Known Issue: When using Explicit Control Modes, the Lanczos and Power Method time step estimators cannot yet be used with problems that have contact, rigid bodies, blocks in the fine mesh that are not controlled by the coarse mesh, or coarse elements that contain no fine nodes.

In explicit dynamics, nodal accelerations are computed by dividing the residual (external minus internal) force by the nodal mass. In the Explicit Control Modes algorithm, the reference mesh residual is mapped to the coarse mesh, and accelerations are computed on the coarse mesh. These accelerations are then interpolated back to the reference mesh. The portion of the residual with higher frequency content (i.e., that which is not representable by the basis functions of the coarse mesh) determines a fine mesh acceleration that is added to the acceleration interpolated from the coarse mesh.

By computing the acceleration on the coarse mesh, the Explicit Control Modes algorithm allows for the critical time step to be computed based on the size of the coarse mesh rather than the size of the reference mesh. A critical time step is estimated based on the coarse mesh. Mass scaling is applied to the high frequency component of the acceleration (computed on the reference mesh to increase the time step to the coarse mesh critical time step). Mass scaling introduces error, but the error only occurs in the high frequency part of the response. In contrast, traditional mass scaling affects the full spectrum of structural response. Explicit Control Modes is effective in problems in which the coarse grid represents the frequency range of interest and is significantly coarser than the reference mesh to maximize the critical time step.

The choice of the degree of refinement in the coarse and reference meshes has a large influence on the effectiveness of the Explicit Control Modes algorithm. The reference mesh should be created

45

to give a discretization that is appropriate to capture the geometry of the problem with sufficient refinement to adequately represent gradients in the discretized solution. The coarse mesh should completely overlay the reference mesh, and it should be coarser than the reference mesh at every location in the model. All coarse elements need not contain elements in the reference mesh; it is possible to use a coarse mesh that extends significantly beyond the domain of the reference mesh.

The user has the freedom to create a coarse mesh that gives an acceptable critical time step without using an excessively crude discretization. Remember that the reference mesh controls the spatial discretization, while the coarse mesh controls the temporal discretization of the model.

To use Explicit Control Modes, the user should set up the reference mesh file and the input file as usual, except that the following additional items must be provided:

- A coarse mesh must be generated, as discussed above. The coarse mesh must be in a separate file from the reference mesh, which is the real model.
- A second `FINITE ELEMENT MODEL` command block must be provided in addition to the standard definition for the reference finite element model in the input file. This command block is set up exactly like the first `FINITE ELEMENT MODEL` command block (described in Sierra/SM 5.8 User's Guide Section 2.3) except that the mesh file referenced is the coarse mesh instead of the reference mesh. Even though the coarse mesh uses no material models, each block in the coarse mesh must be assigned a material model.
- A `CONTROL MODES REGION` command block must appear alongside the standard `PRESTO REGION` command block within the `PRESTO PROCEDURE` command block. The presence of the `CONTROL MODES REGION` command block instructs Presto to use the Explicit Control Modes algorithm. The `CONTROL MODES REGION` command block is documented in Section 8.2. It contains the same commands used within the standard `PRESTO REGION` command block, except that the commands in the `CONTROL MODES REGION` command block are used to control the control modes algorithm and the boundary conditions on the coarse mesh.

8.2.   Control Modes Region

```
BEGIN CONTROL MODES REGION
   #
   # model setup
   USE FINITE ELEMENT MODEL <string>model_name
   CONTROL BLOCKS [WITH <string>coarse_block] =
                        <string list>control_blocks
   CONTROL ASSEMBLIES [WITH <string>coarse_block] =
                        <string list>control_assemblies
   #
   # time step control
   TIME STEP RATIO SCALING = <real>cm_time_scale(1.0)
   TIME STEP RATIO FUNCTION = <string>cm_time_func
   LANCZOS TIME STEP INTERVAL =
                           <integer>lanczos_interval
   POWER METHOD TIME STEP INTERVAL =
                             <integer>pm_interval
   #
   # mass scaling
   HIGH FREQUENCY MASS SCALING =
                           <real>cm_mass_scale(1.0)


   #
   # stiffness damping
   HIGH FREQUENCY STIFFNESS DAMPING COEFFICIENT =
                           <real>cm_stiff_damp(0.0)
   #
   # kinematic boundary condition commands
   BEGIN FIXED DISPLACEMENT
                           # Parameters for
                           # fixed displacement
   END [FIXED DISPLACEMENT]
   #
   # output commands
   BEGIN RESULTS OUTPUT <string> results_name
                           # Parameters for
                           # results output
   END RESULTS OUTPUT <string> results_name
END [CONTROL MODES REGION]
```

The CONTROL MODES REGION command block controls the behavior of the control modes
algorithm and is placed alongside a standard PRESTO REGION command block within the
PRESTO PROCEDURE scope. With the exception of the CONTROL BLOCKS and
CONTROL ASSEMBLIES command line, all the commands that can be used in this block are
standard commands that appear in the Presto region. These commands have the same meaning in
either context- they apply to the coarse mesh or to the reference mesh, depending on the region

block in which they appear. Sections 8.2.1 through 8.2.3 describe the components of the
`CONTROL MODES REGION` command block.

8.2.1. *Model Setup Commands*

```
USE FINITE ELEMENT MODEL <string>model_name
CONTROL BLOCKS [WITH <string>coarse_block] =
      <string list>control_blocks
CONTROL ASSEMBLIES [WITH <string>coarse_block] =
      <string list>control_assemblies
```

The command lines listed above must appear in the `CONTROL MODES REGION` command block if
Explicit Control Modes is used. The `USE FINITE ELEMENT MODEL` command line should
reference the finite element model for the coarse mesh. This command line is used in the same
way that the command line is used for the reference mesh [3].

The `CONTROL BLOCKS` and `CONTROL ASSEMBLIES` command lines provide a list of blocks or
assemblies of blocks in the reference mesh controlled by the coarse mesh. The block names are
listed using the standard format for referencing mesh entities [3]. For example, the block with an
ID of 1 would be listed as `block_1` in this command. Multiple `CONTROL BLOCKS` and
`CONTROL ASSEMBLIES` command lines may be used.

The `CONTROL BLOCKS` and `CONTROL ASSEMBLIES` command lines do not require the coarse
blocks used to control the fine blocks or fine assemblies of blocks to be listed. In the following
example, blocks 10 and 11 and assemblies 12 and 13 are controlled by the coarse mesh, but the
element blocks in the coarse mesh that control those blocks are not listed:

```
    CONTROL BLOCKS = block_10 block_11
    CONTROL ASSEMBLIES = assembly_12 assembly_13
```

If the `CONTROL BLOCKS` and `CONTROL ASSEMBLIES` command lines are used in this manner,
the search for fine nodes contained within coarse elements will be conducted for all elements in
the coarse mesh. The coarse block used to control a given set of fine blocks or assemblies of
blocks can optionally be specified by using the `CONTROL BLOCKS WITH coarse_block` or
`CONTROL ASSEMBLIES WITH coarse_block` variants of the command. For example, the
command:

```
    CONTROL BLOCKS WITH block_1 = block_10 block_11
    CONTROL ASSEMBLIES WITH block_1 = assembly_12 assembly_13
```

would use block 1 on the coarse mesh to control blocks 10 and 11 and control assemblies 12 and
13 on the fine mesh. These variants of the commands are necessary when the coarse blocks or
assemblies of blocks overlap. It removes any ambiguity about which coarse elements control
which fine nodes. This is particularly useful for contact problems where the fine block or
assembly on one side of an interface should be controlled by one block, and the fine block or
assembly on the other side of the interface should be controlled by a different block. Only one
coarse block can be listed in a given instance of this command. If there are multiple coarse
blocks, they must be listed in separate commands.

8.2.2. Time Step Control Commands

The time step control commands for Explicit Control Modes are based on the Sierra/SM time step control commands (consult Sierra/SM 5.8 User's Guide Chapter 3.

```
TIME STEP RATIO SCALING = <real>cm_time_scale(1.0)
TIME STEP RATIO FUNCTION = <string>cm_time_func
LANCZOS TIME STEP INTERVAL =
                        <integer>lanczos_interval
POWER METHOD TIME STEP INTERVAL =
                        <integer>pm_interval
```

The control modes algorithm computes a node-based time step for the coarse mesh at each time step and uses this as the default time step. This time step is typically larger than the critical time step for the fine mesh.

The `TIME STEP RATIO SCALING` and `TIME STEP RATIO FUNCTION` command lines allow the user to control the time step used with explicit control modes. The `TIME STEP RATIO SCALING` command specifies a scale factor `cm_time_scale`, which has a default value of 1.0. The `TIME STEP RATIO FUNCTION` command specifies a function `cm_time_func` that is used to control the scale factor as a function of time. At any given time, a scale factor, f_{ts}, is computed by multiplying `cm_time_scale` by the current value of the function. Both of these commands are optional and one can be used without the other.

The time step Δt, is computed as a function of f_{ts}, as well as of the time step of the fine mesh, Δt_f and the time step of the coarse mesh, Δt_c.

$$\Delta t = \Delta t_f + f_{ts}(\Delta t_c - \Delta t_f) \tag{8.1}$$

Thus, if the scale factor is zero, the time step of the fine mesh is used, and if it is one, the time step of the coarse mesh is used.

The nodal time step estimator for the coarse mesh typically works well on problems where the fine mesh overlaid by the coarse mesh is essentially isotropic. In cases where it is not, such as when there are significant voids covered by the coarse mesh, the nodal time step can be non-conservative, resulting in stability problems. The time step control command lines described above can be used to manually scale down the time step in such scenarios.

Here is an example of stubs for controlling the time step.

```
# Coarse/Fine mesh scaling function.
begin function cm_time_ratio
  type is piecewise linear
  begin values
    0.0      1.0
    0.005    1.0
    0.01     1.0
  end values
```

```
end
begin control modes region beamCoarse
    use finite element model beamCoarse
    control blocks = block_1
    filter = mass
    time step ratio scaling = 1.0
    time step ratio function = cm_time_ratio
    high frequency mass scaling = 1.0
end
```

This particular example uses the default parameters. Non-default parameters may be accessed by changing the corresponding 1.0 values.

Alternatively, either the Lanczos or Power Method global time step estimators can be applied to the coarse mesh to give an improved estimate of the stability limit. These are invoked using the `LANCZOS TIME STEP INTERVAL` or `POWER METHOD TIME STEP INTERVAL` command lines, respectively. Only one of these command lines can be used at a time, and both commands specify an interval at which the global time step estimate is calculated. When the global time step estimate is calculated, a ratio of the global estimate to the nodal estimate is calculated, and this ratio is used to scale the nodal estimate in subsequent time steps in which the global estimate is not computed.

Experience has shown that the time step predicted by the global time step estimators is typically slightly higher than the actual stability limit. For this reason, it is recommended that a scale factor of 0.9 be used in conjunction with these estimators. This can be set using the `TIME STEP SCALE FACTOR` command line in the `TIME CONTROL` block as described in the Sierra/SM 5.8 User's Guide Section 3.1.

8.2.3.   Mass Scaling Commands

`HIGH FREQUENCY MASS SCALING = <real>cm_mass_scale(1.0)`

The `HIGH FREQUENCY MASS SCALING` command line allows the user to control the mass scaling applied to the high frequency component of the response. The mass scaling factor required to stably integrate the high frequency response at the time step being used is computed at every node on the fine mesh. The parameter `cm_mass_scale` that can optionally be supplied with this command line is applied as a multiplier to that mass scaling. If that mass scaling (multiplied by `cm_mass_scale`) is greater than 1.0, then the scaled mass is used at that node. If not, the original nodal mass is used.

It may be useful for some models to use this command line to set `cm_mass_scale` to a value greater than 1.0 to stabilize the high frequency response. Experience has shown, however, that this is rarely needed.

It is also possible to request a consistent coarse mass matrix, instead of the default lumped mass matrix. Here is an example illustrating the syntax.

```
begin control modes region beamCoarse
```

```
   use finite element model beamCoarse
   control blocks = block_1
   coarse mass linear solver = feti
   coarse mass matrix = consistent
   filter = consistent
   time step ratio scaling = 0.2
end
```

8.2.4.   *Damping Commands*

```
HIGH FREQUENCY STIFFNESS DAMPING COEFFICIENT =
                           <real>cm_stiff_damp(0.0)
```

The `HIGH FREQUENCY STIFFNESS DAMPING COEFFICIENT` command is used to apply
stiffness-proportional damping on the high frequency portion of the response in Explicit Control
Modes. This may help reduce high frequency noise in problems that have abrupt loading such as
that caused by contact. The default value of `cm_stiff_damp` is 0.0. The value specified for
`cm_stiff_damp` can be between 0 and 1. It is recommended that small values (around 0.001) be
specified if this option is used.

8.2.5.   *Kinematic Boundary Condition Commands*

```
BEGIN FIXED DISPLACEMENT
        #
        # Parameters for fixed displacement
        #
END FIXED DISPLACEMENT
```

All types of kinematic boundary conditions can be applied to the coarse mesh. This is done by
inserting a kinematic boundary condition command block in the `CONTROL MODES REGION`
command block. The mesh entity (node set, surface, or block) to which the boundary condition is
applied must exist on the coarse mesh.

This capability is potentially useful to ensure better enforcement of kinematic boundary
conditions on the fine mesh by applying the same type of boundary condition on the portion of the
coarse mesh that overlays the portion of the fine mesh to which boundary conditions are applied.
For example, if there is a node set on the fine mesh that has a fixed displacement boundary
condition, a node set can be created on the coarse mesh that covers the same physical domain.
The same fixed displacement boundary condition could then be applied to the coarse mesh.

Although the capability to enforce boundary conditions on the coarse mesh is provided, it is not
necessary to do so. It is also often difficult to create a node set on the coarse mesh that matches
the discretization of the node set on the fine mesh. Users are advised to initially prescribe
kinematic boundary conditions only on the fine mesh and only prescribe boundary conditions on
the coarse if the initial results appear questionable.

8.2.6. Output Commands

```
BEGIN RESULTS OUTPUT <string> results_name
        #
        # Parameters for results output
        #
END RESULTS OUTPUT <string> results_name
```

Variables can be output from the coarse mesh just as they can from the fine mesh with Explicit Control Modes. Because the actual results of interest for the model all reside on the fine mesh, it is typically not necessary to output results on the coarse mesh. However, this can be helpful for debugging purposes.

The syntax for the results output for the coarse mesh is identical to that used for output from the fine mesh [3]. The only thing that differentiates the `RESULTS OUTPUT` command block for the coarse mesh from that of the fine mesh is that the results output block for the coarse mesh is put in the `CONTROL MODES REGION` command block instead of in the `PRESTO REGION` command block. The output files for the coarse and fine mesh must be different from each other, so different output file names must be used within the output blocks for the coarse and fine meshes.

One of the most useful variables to output from the coarse mesh is the nodal `timestep`. This variable is similar in nature to the element `timestep`, which exists on the fine mesh, but is a nodal variable rather than an element variable and exists on the coarse mesh. The nodal `timestep` reports the critical time step calculated for each node on the coarse mesh. If the coarse time step is higher than expected, the output from `nodal_time_step` can be examined to see which region of the coarse mesh is controlling the time step.

Central difference time integration is performed on the coarse mesh in addition to the fine mesh, so the `displacement`, `velocity`, and `acceleration` variables can be requested for visualization on the coarse mesh.

8.2.7. ECM with Lanczos

As the known issue suggests that Lanczos works with ECM. Here's an example.

```
begin control modes region beamCoarse
  use finite element model beamCoarse
  control blocks = block_1
  filter = mass
  time step ratio scaling = 0.9
  high frequency mass scaling = 1.0
  lanczos time step interval = 100
end
```

Assume for clarity that the Lanczos time step size is significantly larger than the element time step size. Initially the `Element time step in Lanczos` is used. At the following time steps, the time step size increases the time step size by a factor of 1.1, the default value of the

`time step increase factor` described in Sierra/SM 5.8 User's Guide Section 3.1.4. When the time step size is near to time Time step from Lanczos method, Lanczos time step, a different time step is used. The maximal time step may depend on `time step increase factor` and other Lanczos parameters. To compare to Lanczos with no coarse grid, add to the presto region block

```
begin lanczos parameters set1
  number eigenvalues = 50
  eigenvalue convergence tolerance = 1.0e-4
  update step interval = 100
  vector scale = 1.e-8
  debug = on
end lanczos parameters set1
```

In the case at hand, the ECM increases the time step from $4e-7$ to $1.5e-6$.

8.3.   ECM Theory

Explicit transient dynamics is a well-established capability for modeling large deformations of structures. It is common practice in explicit transient dynamics to seek a balance between computational efficiency and accuracy. Mass scaling [4] has traditionally been used as an approach to increase the critical time step limit associated with the central difference time integrator. Unfortunately, this has an undesirable side effect of mass damping dynamic modal response over the entire frequency spectrum. To mitigate this effect, methods have been developed in which the damping is proportional to the frequency [5]. In Adagio the Explicit Control Modes algorithm performs an efficient modal decomposition of the frequency spectrum, allowing mass damping only on the high frequency modes. Examples will be presented that demonstrate that this approach yields accurate low frequency response, while often using larger time steps due to the mass scaling the high frequency response.

8.3.1.   Introduction

Finite element analysis of transient dynamic problems is a production capability in many application areas. In these analyses an important question to be addressed by the analyst is the choice of using an explicit or implicit time integrator. It is well understood that the central difference explicit time integrator is efficient per time step but is restricted to taking relatively small, critical (or stable) time steps [6, 7]. An implicit time integrator, specifically the Hilber–Hughes–Taylor (HHT) time integrator [8], with the proper choice of parameters has no such stability limit allowing larger time steps but produces a system of equations that need to be solved every time step. As noted in [8], a consequence of choosing a large time step for the implicit time integrator is that it produces numerical damping in all frequencies, but predominantly in the highest frequencies. How much damping and in what modes depends on the particular problem. Thus, the question of what time integrator to use is much more than one of efficiency. Certainly, one must know well the class of problems to be solved when making this choice.

Simulations requiring necessarily finer discretizations to accurately represent modal stiffness and resolve details in the stress field are amenable to Explicit Control Modes. For the explicit time integrator, this imposes a critical time step restriction that can be limiting. However for some—possibly many—analyses the structural response is in the lower frequency spectrum, i.e., the influence of the modal content inherent in fine mesh on the low-frequency dynamics is of interest and not necessarily the high-frequency dynamics themselves. More precisely, spatial resolution as opposed to temporal resolution is needed for many problems (this premise is one that we intend to support in the examples).

It seems appropriate, then, to consider an algorithmic approach that can improve the stability limit of the explicit time integrator. Most importantly, we recognize that this approach needs to be accurate for low-mode response and competitive with implicit dynamics.

8.3.2.   *Modal Decomposition Approach*

The objective of this algorithm is to modally decompose the dynamics (in the context of an explicit transient dynamics time integrator) into low-frequency and high frequency response. Having this decomposition may provide options such as integrating the low-frequency modes with explicit time integration and the high-frequency modes with an implicit time integrator.

The decomposition algorithm is based on applying multigrid concepts within an explicit central difference time integrator. We will limit the algorithm to consider only one addition level of coarsening. Thus, in addition to the fine mesh or reference mesh, we introduce a set of coarse basis functions that will describe the low-mode response.

The vector of external nodal forces on the fine mesh is f_{fm}^{ext}. Also the vector of internal nodal forces on the fine mesh f_{fm}^{int} is a obtained from the divergence of the stress. In this work we assume that there is no contact, in which case the nodal residual force is

$$r = f_{fm}^{ext} - f_{fm}^{int}.$$

Let M denote the diagonal, lumped [9], mass matrix for the fine mesh, and let x denote the vector of displacements on the fine mesh. Expressed in terms of the nodal displacements, the dynamic equilibrium equations is

$$M\ddot{x} = r. \tag{8.2}$$

Let Φ denote an interpolation (prolongation) matrix associated with a coarse space of functions. The number of rows in Φ equals the number of rows in x or r, while the number of columns in is typically smaller. The matrix can be obtained from either a coarse finite element mesh or by using an algebraic approach [10, 11]. Given Φ, the acceleration on the fine mesh can be written as

$$\ddot{x} = \Phi\ddot{q} + \ddot{x}_{hf} \tag{8.3}$$

where q is a vector of generalized displacements associated with the low frequency part of the response, and x_{hf} is a vector of displacements associated with the high frequency part of the response.

The task now is to derive the equation to accomplish this decomposition making use only of the residual vector, r, and mass matrix, M, on the fine mesh, recognizing that there are no properties on the coarse mesh in the usual finite element sense. As in the multigrid method, the modal stiffness of low-mode response and the corresponding mass matrix is obtained using a restriction operator of properties/quantities from the fine mesh.

The low and high frequencies are decoupled by imposing the M orthogonality,

$$\ddot{q}^T \Phi^T M \ddot{x}_{hf} = 0,$$

of the high and low frequency displacements. The orthogonality condition holds for all $\ddot{q}$ if and only if

$$\Phi^T M \ddot{x}_{hf} = 0. \tag{8.4}$$

Equation (8.4) implies that the high frequency part of the residual is orthogonal to the coarse space spanned by the columns of Φ. The coarse mesh mass matrix is given by

$$M_c = \Phi^T M \Phi.$$

Substitution of equation (8.3) into equation (8.2), pre-multiplying by Φ^T and making use of equation (8.4) leads to the low frequency equilibrium condition

$$M_c \ddot{q} = \Phi^T r. \tag{8.5}$$

This way of obtaining a coarse system from Φ is called *Galerkin coarsening* [11]. For reference, the coarse grid stiffness matrix K_c corresponding to the fine mesh tangent stiffness matrix K is given by

$$K_c = \Phi^T K \Phi.$$

Next the high frequency equilibrium equation is derived. Solving for $\ddot{q}$ in equation (8.5) gives

$$\ddot{q} = M_c^{-1} \Phi^T r. \tag{8.6}$$

From equation (8.2), equation (8.3), and equation (8.6) determine the high frequency equilibrium condition

$$M \ddot{x}_{hf} = r - M \Phi M_c^{-1} \Phi^T r. \tag{8.7}$$

At this point no approximations have been made. To sum up, substituting equation (8.6) and equation (8.7) into equation (8.3) leads to

$$\ddot{x} = \underbrace{\Phi M_c^{-1} \Phi^T r}_{\text{low frequency}} + \underbrace{M^{-1} \left(r - M \Phi M_c^{-1} \Phi^T r \right)}_{\text{high frequency}} \tag{8.8}$$

The lumped mass matrix is required to obtain the most accurate approximation properties for the explicit central difference time integrator [7, 12]. Thus, given that we are integrating the low-frequency response with central difference, a lumped representation is needed. It is unclear that the argument for finite elements and the fine mesh extend to the Galerkin coarse problem. The lumping is done by applying the restriction operator to the diagonal lumped fine mesh mass matrix,

$$M_c = \Phi^T M \Phi.$$

8.3.3. *Explicit-Explicit Partitioning*

First we consider explicit time integration for the low-frequency modes. The critical time step for integrating these modes is constructed, again via projection of nodal quantities on the fine mesh. If Δt_{cr}^{cm} denotes the critical time step for the coarse mesh, then a node-based estimate [6] is given by

$$\Delta t_{cr}^{cm} = \min_{\text{coarse nodes}} 2 \sqrt{\frac{\Phi^T M}{\Phi^T K^{max}}}$$

where K^{max} is a vector that contains the maximum modal stiffness for each node of the fine mesh. Details of the calculation of the maximum modal stiffness can be found in [6].

Next, we wish to make use of the assumption that the high-frequency dynamics are negligible. The accelerations represented by the second term in equation (8.8) correspond to those

56

high-frequency modes. The idea is to replace M^{-1} in the second term of equation (8.8) by $\tilde{M}^{-1}$, where

$$\tilde{M} = M\alpha, \tag{8.9}$$

for a diagonal matrix α that contains a scale factor for each node of the fine mesh. These scale factors are greater than 1 wherever the nodal based time step at a fine mesh node is smaller than the critical time step on the coarse mesh.

$$\alpha_i = \begin{cases} \frac{K_i^{max}}{4M_i} \left(\Delta t_{cr}^{cm}\right)^2 & \text{if } 2\sqrt{\frac{M_i}{K_i^{max}}} > \Delta t_{cr}^{cm} \\ 1 & \text{otherwise} \end{cases} \tag{8.10}$$

Consequently, the mass scaling produced by equation (8.10) is applied *only* to the high-frequency modes that could not otherwise be integrated stably with the central difference time integrator at the critical time step on the coarse mesh. The net result for the acceleration, $\ddot{x}$, on the reference mesh is of the form

$$\ddot{x} = \underbrace{\Phi M_c^{-1}\Phi^T r}_{\text{low frequency}} + \underbrace{\tilde{M}^{-1}\left(r - M\Phi M_c^{-1}\Phi^T r\right)}_{\text{mass-damped high frequency}} \tag{8.11}$$

8.3.4. *Energy Ratio: a Measure of Approximation*

Kinetic energy calculations can be performed for the low frequency and high frequency contributions separately. Using time integrated acceleration components in (16), the kinetic energy in the low frequencies is,

$$KE_{lf} = \frac{1}{2}M\|v_{lf} + \Delta t\Phi M_c^{-1}\Phi^T r\|^2 \tag{8.12}$$

Likewise, the kinetic energy in the high frequencies is,

$$KE_{hf} = \frac{1}{2}M\|v_{hf} + \Delta t\tilde{M}^{-1}\left(r - M\Phi M_c^{-1}\Phi^T r\right)\|^2 \tag{8.13}$$

With the kinetic energy quantities, an energy ratio is computed as follows,

$$\text{Energy Ratio} = \frac{KE_{lf}}{KE_{lf} + KE_{hf}}$$

Obviously, the time integrated estimates of the kinetic energies require additional memory yet they provide a useful measure for the approximations being made with the explicit-explicit modal filtering. When there is little or no approximation made using a mass-damped high frequency response the energy ratio is asymptotically approaching unity. In contrast, when the approximation error is significant, the energy ratio is well below one.

References

[1] Sierra/SolidMechanics Team. Sierra/SolidMechanics VOTD Theory Manual. Technical Report In draft, Sandia National Laboratories, Albuquerque, NM, 2019.

[2] B.W. Spencer, M.W. Heinstein, J.D. Hales, K.H. Pierson, and J.R. Overfelt. Multi-length scale algorithms for failure modeling in solid mechanics. Technical Report SAND2008-6499, Sandia National Laboratories, Albuquerque, NM, 2008.

[3] Sierra/SolidMechanics Team. Sierra/SolidMechanics VOTD User's Guide. Technical Report In draft, Sandia National Laboratories, Albuquerque, NM, 2019.

[4] T. Belytschko, W.K. Liu, and B. Moran. *Nonlinear Finite Elements for Continua and Structures*. John Wiley & Sons, 2000.

[5] L. Olovsson, K. Simonsson, and M. Unosson. Selective mass scaling for explicit finite element analyses. *International Journal for Numerical Methods in Engineering*, 63(10): 1436–1445, 2005.

[6] M. Heinstein, F. Mello, and C. Dohrmann. A nodal-based stable time step predictor for transient dynamcis with explicit time integration. *Am Soc Mech Eng Press Vessels, Piping Division*, 343:225–229, 1996.

[7] T.J.R. Hughes. *The Finite Element Method: Linear static and dynamic finite element analysis*. Dover, 2000. Reprint of "The Finite Element Method", Prentice-Hall, 1987.

[8] H.M. Hilber, T.J.R. Hughes, and R.L. Talor. Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering and Structural Dynamics*, 5:283–292, 1977.

[9] E. Hinton, T. Rock, and O.C. Zienkiewicz. A note on mass lumped and related processes in the finite element method. *Earthquake Engineering and Structural Dynamics*, 4(3): 245–249, 1976.

[10] A. Toselli and O. Widlund. *Domain Decomposition Methods – Algorithms and Theory*. Springer Series in Computational Mathematics. Springer, 2005.

[11] A. Brandt. General highly accurate algebraic coarsening. *Elect. Trans. Numer. Anal.*, 10: 1–20, 2000.

[12] R.D. Krieg and S.W. Key. Transient shell response by numerical time integration. *International Journal for Numerical Methods in Engineering*, 7:273–286, 1973.

Implicit Only

9.   EXTERNAL LOADSTEP PREDICTOR

Production-ready loadstep predictor types are available in Sierra/SM [1]. The
`LOADSTEP PREDICTOR` command block controls the behavior of the predictor that is used to
predict the solution at the beginning of a new load step. This command block is placed in the
`SOLVER` scope.

The `EXTERNAL`, `EXTERNAL_FIRST` and `TANGENT` predictor types are special use capabilities
currently under development.

```
BEGIN LOADSTEP PREDICTOR
       TYPE = <string>EXTERNAL|EXTERNAL_FIRST|TANGENT
END [LOADSTEP PREDICTOR]
```

The tangent predictor is selected with the `TANGENT` option, which is useful in combination with
the tangent preconditioner. This type of predictor uses the tangent preconditioner to estimate the
next load step's solution.

The other two predictor types use the solution from a file to predict the solution at new load steps.
For instance, the external predictor file can come from the results output of a previous model run
that included the command `OUTPUT EXTERNAL PREDICTOR VARIABLES` in the output block,
i.e.:

```
BEGIN RESULTS OUTPUT
       OUTPUT EXTERNAL PREDICTOR VARIABLES
END [RESULTS OUTPUT]
```

If you would like to try the external predictor, please contact Sierra support for more information.

References

[1] Sierra/SolidMechanics Team. Sierra/SolidMechanics VOTD User's Guide. Technical Report
    In draft, Sandia National Laboratories, Albuquerque, NM, 2019.

10. BOLT

⚠️ **Warning:** The Bolt section is known to have limited functionality in implicit analyses.

```
BEGIN BOLT SECTION <string>section_name
  ATTACHMENT RADIUS = <real>radius
  SURFACE 1 = <string>surf1
  SURFACE 2 = <string>surf2
  NORMAL DISPLACEMENT FUNCTION = <string>normFunc
  SHEAR DISPLACEMENT FUNCTION = <string>shearFunc
END
```

The `BOLT` command block is used to define a two node beam or set of beams representing individual bolts or other fasteners. This capability is similar to the `SPOT WELD` capability. The beam elements should be meshed such that one beam end node is roughly on surface 1 and the other beam end node is roughly on surface 2. The beam element does not need to be meshed contiguous with the surface nodes.

The beam element is attached to all nodes and faces within a specified radius of the beam end nodes given by the `ATTACHMENT RADIUS` command. To be valid the bolt must find at least one face and three nodes within this radius on each surface.

The `NORMAL DISPLACEMENT FUNCTION` and `SHEAR DISPLACEMENT FUNCTION` define normal and shear force displacement functions for the bolt. The normal displacement function defines tensile response in positive x and compressive response in negative x. The shear displacement function is radially symmetric and only the positive x portion of the function will be used. The last point on the shear displacement function and the first and last points on the normal displacement function implicitly define the bolt failure criteria. Once a bolt fails the strength will ramp down over 10 steps and the bolt will provide zero force thereafter.

The bolt uses the same combined shear/normal mode failure as does the spot weld as defined in Equation 10.1. u_n is the bolt normal extension. The maximum value given for u_n in the normal displacement curve is $u_{n_{crit}}$, but is different for positive and negative displacements. u_t is the bolt shear deformation. The maximum value given for u_t in the normal displacement curve is $u_{t_{crit}}$. The value p is a exponent that controls the shape of the failure surface, currently this exponent is defaulted to 2.

$$(u_n/u_{n_{crit}})^p + (u_t/u_{t_{crit}})^p < 1.0. \qquad (10.1)$$

The original direction defining normal and shear displacement is defined by the bolt element orientation. This normal will rotate based on the rotation of attached faces, not rotation of the bolt element itself.

Table 10-1 describes the output variables available on the bolt elements.

Table 10-1. Bolt Element Output Variables

Name	Description
displacement_normal	Current normal displacement in bolt
displacement_shear	Current shear displacement in bolt
force_normal	Current normal force in bolt
force_shear	Current shear force in bolt
bolt_death_status	One for alive, zero for dead, some value between zero and one when fading out immediately after hitting the death criteria.

11.   LINEAR BEAM

⚠️ **Warning:** The Linear Beam section is known to have limited functionality in implicit analyses.

```
BEGIN LINEAR BEAM SECTION <string>section_name
    T AXIS = <real>tx <real>ty <real>tz
    AREA = <real>area
    I11 = <real>i11
    I22 = <real>i22
    I12 = <real>i12(0.0)
    J = <real>J
    SHEAR AREA 1 = <real>val(AREA)
    SHEAR AREA 2 = <real>val(AREA)
END
```

The `LINEAR BEAM SECTION` command block specifies the properties for a linear beam element. If this command block is referenced in an element block of three-dimensional, two-node elements, the elements in the block will be treated as beam elements. The name, `beam_section_name`, can be used by the `SECTION` command line in a `PARAMETERS FOR BLOCK` command block.

The beam geometry properties are defined via areas and moments of inertia for the beam section. The linear beam will behave as a linear elastic element. If a linear beam has a nonlinear material, only the elastic constants of that material, such as Young's modulus and Poisson's ratio, will affect the beam behavior.

The beam element is formulated in a local orthogonal RST coordinate system. The R axis of the beam lies along the beam element. The T axis direction is given in the input deck. If the provided T axis is not orthogonal to R, the closest vector to T that is orthogonal to R will be used define the T axis. The S axis is then constructed orthogonal to R and T based on the right hand rule (The actual method of forming these axes is slightly different from this description.). The `T AXIS` command in the linear beam behaves identically to the `T AXIS` command in the standard beam. See the `BEAM SECTION` description in the Sierra/SM 5.8 User's Guide Section 6.2.8 for more examples and discussion on use of the `T AXIS` command.

The following cross sectional properties are available for linear beams.

- `AREA`: Cross sectional area used to define axial and shear properties.
- `I11`: Bending moment of inertia in the T direction of the beam.
- `I22`: Bending moment of inertia in the S direction of the beam.
- `I12`: Product of inertial of the beam for asymmetric sections. This value is by default set to zero.

- `J`: Polar moment of inertia used to define beam torsional properties.
- `SHEAR AREA 1`: Area used for shear resistance in the T direction. If unspecified the cross sectional area `AREA` will be used.
- `SHEAR AREA 2`: Area used for shear resistance in the S direction. If unspecified the cross sectional area `AREA` will be used.

This linear beam is a Timoshenko (also called a Reissner-Mindlin) shear deformable thick beam. If the thickness is small relative to the length, it behaves like an Euler-Bernoulli beam. The pre-integrated element stiffness was taken directly from Reference [1].

Note, linear beam elements do not calculate element stress or stress based quantities. Linear beam elements generate nodal internal forces however no element specific output quantities are currently available on linear beam elements.

References

[1] J.S. Przemieniecki. *Theory of Matrix Structural Analysis*. Dover Publications Inc., New York, NY, 1985.

12.   CONTACT

This chapter describes contact features that are not fully tested or are still in development or have usability issues.

12.1.   Implicit Solver Control Contact Options

```
BEGIN CONTROL CONTACT
    CONTROL SUBSET = <list>
            controlTypes(ADAGIO, ARS, JAS)
END [CONTROL CONTACT]
```

The `CONTROL SUBSET` command restricts a control contact block to only apply to some contact enforcement types. The options to the command are ADAGIO to control kinematic and augmented Lagrange contact, JAS to control JAS mode contact, and ARS (Goodyear specific) to control analytic rigid surface contact. By default, the control contact block applies to all three contact types. Use of the control subset logic may be useful if it is desired to have the different enforcement types use different control contact option sets.

The `CONTROL CONTACT` block is described in Sierra/SM 5.8 User's Guide [1].

The `AREA UPDATE FREQUENCY` is a performance option used to control how often the analytic contact surfaces update the local areas and normal directions. Updating these values more frequency (lower `numStep`) may lower performance but yield greater accuracy (especially in derived output quantities such as contact traction).

If the `CONTACT FORCE PREDICTOR` option is on the previous step contact forces will be used as an initial guess to the current step contact forces. This could improve results if the contact forces are stable step to step or make results worse if the contact forces are highly volatile. The default value for this option is `ON` as ARS contact is often used to model mostly static contacts.

12.2.   Contact on Smooth Surfaces

A developmental option is available for contact on smooth surfaces (such as pins rotating in holes or ball bearings). In its current implementation, contact constructs a Gregory patch (following the method of [2]) for each face in `Side A` that is in contact, predicting the curvature of the face from an average node normal calculation. The node normals are updated every timestep in order to capture the deformation of the smooth surface. The contact gap and resulting forces are computed in this current configuration, and the forces are distributed out to the nodes via the linear shape functions of the original face. Convergence is achieved via this method due to the Augmented Lagrangian enforcement iterations performed by DASH. Example syntax is shown below:

```
BEGIN CONTACT DEFINITION EXAMPLE_SMOOTH_CONTACT
  CONTACT SMOOTH SURFACE SMOOTH_SURF CONTAINS SURFACE_1
  CONTACT NODE SET NODE_SIDE CONTAINS NODELIST_1000
  BEGIN INTERACTION SMOOTH_1
    CONSTRAINT FORMULATION = NODE_FACE
    SIDE A = SMOOTH_SURF
    SIDE B = NODE_SIDE
  END
END
```

⚠️ **Warning:** Smooth contact is only available for solid quad faces.

⚠️ **Warning:** This option is only implemented for node-face contact. If unspecified it will default to node-face.

`Side A` is the smooth surface side and `Side B` is the node side in a smooth contact node-face interaction.

12.2.1.   *Surface Normals and Curvature Metrics*

Smooth surfaces have two metrics that allow us to assess the curvature of the surface `smooth_surface_altitude` and `smooth_surface_curvature`. Both of these are calculated on the interior nodes of the smooth contact surfaces. Interior nodes are nodes that belong to only one smooth surface. The nodal variables `smooth_surface_curvature` and `smooth_surface_altitude` can be requested for output onto the results file. `smooth_surface_altitude` is calculated as the average signed distance of adjacent nodes to the plane created at the node with the nodal normal of the smooth surface, the average is scaled with the characteristic length. The `curvature` is the mean curvature calculated using the method of [3].

The min and max curvature metrics are collected and printed to the log file for each smooth surface. For example:

```
-------- DASH SMOOTH SURFACE METRICS --------
Contact Entity   Surf Index   Node Normal             Min Altitude   Max Altitude   Min Curvature   Max Curvature
---------------  ----------   ----------------------  ------------   ------------   -------------   -------------
smooth_surf      0            smooth_surface_normal   5.105e-02      5.105e-02      2.500e-01       2.500e-01
```

In addition the table in the log file indicates the nodal variable name used for the smooth surface nodal normal for each smooth surface. Each smooth surface has a unique normal so nodes shared between smooth surfaces will have multiple normals. The node normal variable can be used to visualize the smooth surface normal, when visualizing surface normals care should be taken to match the node normal variable name with the surface being visualized.

> ⚠️ **Warning:** Multiple smooth surfaces can be defined but a maximum of 5 surfaces can share a common node.

> ⚠️ **Warning:** Node normals are crucial for accurate smooth surfaces. Symmetry boundaries have a lack of information for the normal calculation, which leads to unexpected behavior. Modeling with symmetry boundaries are not currently recommended.

12.2.2.   Automatic Selection of Side A and Side B

The best way to guarantee that a surface is used as a smooth surface is to set Side A using the side a = command to setup contact. The automatic selection of Side A and Side B occurs when the surface are specified with surfaces =. The same selection logic applied to node-face contact is used when the characteristic lengths of the two sides are not within 2x of each other. Otherwise, if one side is smooth and the other surface is not the Side A is set to the smooth side. Then if one side is flat and the other side has curvature, the side with curvature is used as Side A. If both sides have curvature the side with the lower curvature is used as Side A.

> ⚠️ **Warning:** In the case of two smooth surfaces, the current method will always assign a flat surface (i.e. one with zero curvature) as Side B. If the intention is that the flat surface should be smoothed, e.g. in the case where it is expected to deform substantially, then it must be directly specified as Side A or it must be the only smooth surface in the interaction.

References

[1] Sierra/SolidMechanics Team. Sierra/SolidMechanics VOTD User's Guide. Technical Report In draft, Sandia National Laboratories, Albuquerque, NM, 2019.

[2] M. A. Puso and T. A. Laursen. A 3d contact smoothing method using gregory patches. *International Journal for Numerical Methods in Engineering*, 54(8):1161–1194, 2002.

[3] S Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. *Proceedings. 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, pages 486–493, 2004.

13.   *J*-**INTEGRALS**

Sierra/SM provides a capability to compute the *J*-integral via a domain integral.

⚠️ **Known Issue:** Currently, the *J*-Integral evaluation capability is based on assumptions of elastostatics and a stationary crack, and is only implemented for uniform gradient hex elements.

J is analogous to *G* from linear elastic fracture mechanics $(-\delta\pi/\delta a)$ and is the driving force on the crack tip of length *a* [1, 2]. Crack propagation occurs when $J(a) \geq R(a)$, where $R(a)$ is the material resistance. For constant *R*, the resistance is often termed J_c. In the reference configuration, the vector form of the *J*-integral in finite deformation [3] is

$$\boldsymbol{J} = \int_{\Gamma_0} \boldsymbol{\Sigma N} dA \qquad (13.1)$$

where $\boldsymbol{\Sigma} = W\boldsymbol{I} - \boldsymbol{F}^T\boldsymbol{P}$ is called the Eshelby energy-momentum tensor [4]. *W* is the stored energy density in the reference configuration and $\boldsymbol{F}$ and $\boldsymbol{P}$ are the deformation gradient and first Piola-Kirchhoff stress, respectively. Rice [2] realized that because $\boldsymbol{\Sigma}$ is divergence-free in the absence of body forces, one can examine $\boldsymbol{J}$ in the direction of the defect $\boldsymbol{L}$ (unit vector) and obtain a path-independent integral for traction-free crack faces. *J* can be written as

$$J = \int_{\Gamma_0} \boldsymbol{L} \cdot \boldsymbol{\Sigma N} dA \qquad (13.2)$$

and interpreted as a path-independent driving force in the direction of the defect. We note that one can also express $\boldsymbol{\Sigma}$ in terms of $\bar{\boldsymbol{\Sigma}}$, where $\bar{\boldsymbol{\Sigma}} = W\boldsymbol{I} - \boldsymbol{H}^T\boldsymbol{P}$ and $\boldsymbol{H} = \text{Grad}\,\boldsymbol{u}$. Although $\boldsymbol{\Sigma}$ is symmetric and $\bar{\boldsymbol{\Sigma}}$ is not symmetric, they are equivalent when integrated over the body $(\text{Div}\boldsymbol{P} = \boldsymbol{0})$. Differences in the energy-momentum tensor stem from the functional dependence of the stored energy function *W*. $\boldsymbol{\Sigma}$ and $\bar{\boldsymbol{\Sigma}}$ derive from $W(\boldsymbol{F})$ and $W(\boldsymbol{H})$, respectively. When integrated, both collapse to the familiar 2-D relation for infinitesimal deformations.

$$J = \int_{\Gamma} \boldsymbol{e}_1 \cdot \boldsymbol{\Sigma n} ds = \int_{\Gamma} (Wn_1 - u_{i,1}\sigma_{ij}n_j) ds \qquad (13.3)$$

13.1. Technique for Computing J

J is often expressed as a line (2D) or surface (3D) integral on a ring surrounding the crack tip. Defining a smooth ring over which to compute this surface integral and performing projections of the required field values onto that ring presents many difficulties in the context of a finite element code.

To compute the J-integral in a finite element code, it is more convenient to perform a volume integral over a domain surrounding the crack tip. We can then leverage the information at integration points rather than rely on less accurate projections. To do this, we follow the method described in [5]. We replace L with a smooth function q. On the inner contour of the domain Γ_0, $q = L$. On the outer contour of the domain C_0, $q = 0$. Because the outer normal of the domain M is equal and opposite of the normal N on Γ_0, there is a change of sign. For traction-free surfaces, we can apply the divergence theorem, enforce $\text{Div}\bar{\Sigma} = 0$, and find that the energy per unit length $\bar{J}$ is

$$\bar{J} = -\int_{\Omega_0} (\bar{\Sigma} : \text{Grad}\,q)\mathrm{d}V. \tag{13.4}$$

We can also introduce a correction if we seek to consider cases in which the crack faces, $A_0^+ + A_0^-$, or "caps", $S_0^+ + S_0^-$, of the domain are not traction free. For example, both cases would be satisfied for a thumbnail crack in a pressure vessel. The correction

$$\bar{J} = -\int_{\Omega_0} (\bar{\Sigma} : \text{Grad}\,q)\mathrm{d}V - \int_{S_0^+ + S_0^- + A_0^+ + A_0^-} q \cdot H^T T \,\mathrm{d}A. \tag{13.5}$$

effectively removes the contribution of the applied tractions and ensures that the integral remains path independent. We note that the correction is integrated in the reference configuration with referential traction T. When pressure is applied to S_0^+, S_0^-, A_0^+, or A_0^-, Sierra employs Equation (13.5).

We note that all the field quantities are given via simulation and we choose to define q on the nodes of the domain q^I. We then employ the standard finite element shape functions to calculate the gradient. We can specify the crack direction L or assume that the crack will propagate in the direction normal to the crack front $-M$. For a "straight" crack front, $L = -M$. If S is tangent to the crack front and T is normal to the lower crack surface, $S \times M = T$. We note that for non-planar, curving cracks, M, S, and T are functions of the arc length S. For ease, we employ the notation N rather than $-M$. For a crack front S_0, we can define the average driving force J_{avg} as

$$J_{avg} = \frac{\bar{J}}{\int_{S_0} L \cdot N \mathrm{d}S}. \tag{13.6}$$

While the average driving force is useful for interpreting experimental findings and obtaining a macroscopic representation of the driving force, we also seek to examine the local driving force $J(S)$. Using the finite element interpolation functions to discretize L through the smooth function q, we find $q = \lambda^I q^I$. For a specific node K, we can define $|q^K| = 1$ and $q^I = 0$ for all other $I \neq K$ on S_0. Note that we still need to specify the function q in the $S - T$ plane from the inner contour Γ_0 to the outer contour C_0. The resulting expression for the approximate, point-wise driving force at

69

node K on the crack front is

$$J^K = \frac{\bar{J}}{\int_{S_0} \lambda^K \boldsymbol{q}^K \cdot \boldsymbol{N} \mathrm{d}S}. \tag{13.7}$$

Again, we note that if the direction of propagation $\boldsymbol{L}$ is taken in the direction of the normal $\boldsymbol{N}$, the denominator is $\int_{S_0} \lambda^K \mathrm{d}S$. More information regarding the point-wise approximation of J^K can be found in [6, 7].

Additional information on the J Integral capability can be found in [8].

13.2.   Input Commands

Output quantities related to *J*-integrals may be computed during the analysis by including one or more `J INTEGRAL` command blocks in the `REGION` scope. This block can contain the following commands:

```
BEGIN J INTEGRAL <jint_name>
        #
        # Definition of integration domain
        BLOCK = <string list>blockNames
        REMOVE BLOCK = <string list>removeBlocks
        ELEMENT = <int_list>elemNumbers
        ASSEMBLY = <string list>assemblyNames
        INCLUDE ALL BLOCKS
        #
        # integral parameter specification commands
        CRACK PLANE SIDE SET = <string list>side_sets
        CRACK TIP NODE SET = <string list>node_sets
        USE SURFACE FOR EDGE DIRECTION = OFF|ON(ON)
        CRACK DIRECTION = <real>dir_x
                          <real>dir_y
                          <real>dir_z
        INTEGRATION RADIUS = <real>int_radius
        NUMBER OF DOMAINS = <integer>num_domains
        FUNCTION =
            PLATEAU|PLATEAU_RAMP|LINEAR(PLATEAU)
        SYMMETRY = OFF|ON(OFF)
        DEBUG OUTPUT = OFF|ON(OFF) WITH
          <integer>num_nodes NODES ON THE CRACK FRONT
        #
        # time period selection commands
        ACTIVE PERIODS = <string list>period_names
        INACTIVE PERIODS = <string list>period_names
END J INTEGRAL <jint_name>
```

The *J*-integral is performed over a domain defined by a set of elements using the standard element assignment commands. See Sierra/SM 5.8 User's Guide Section 7.1.1 for details.

A set of parameters must be provided to define the crack geometry used in the calculation of the *J*-integral. The *J*-integral command block uses a sideset on one surface of the crack plane behind the crack tip and a nodeset containing the nodes on the crack tip. Both the `CRACK PLANE SIDE SET` and `CRACK TIP NODE SET` commands are required. These commands specify the names of the sideset behind the crack tip and the nodeset on the crack tip, respectively. Assemblies may contain blocks, or assemblies of these.

By default, the direction of crack propagation is computed from the geometry of the crack plane and tip, as provided in the crack plane sideset and crack tip nodeset ($L = N$). At locations where

the crack intersects a surface, the computed N is commonly less accurate. At these locations, the USE SURFACE FOR EDGE DIRECTION = ON command will attempt to improve N by projecting it onto the intersected surface. The default is ON, and USE SURFACE FOR EDGE DIRECTION = OFF should only be set if the crack tip is known to intersect the surface at a sharp angle. Alternatively, the CRACK DIRECTION command can be used to override the computed direction of crack propagation (L). This command takes three real numbers that define the three components of the crack direction vector in the global XYZ coordinate system.

To fully define the domains used for the domain integrals, the radius of the domains and the number of domains must also be specified. A series of disc-shaped integration domains are formed with varying radii going out from the crack tip. The INTEGRATION RADIUS command specifies the radius of the outermost domain. The number of integration domains is specified using the NUMBER OF DOMAINS command. The radii of the domains increase linearly going from the innermost to the outermost domain. The domains will only include elements included in the overall integration domain defined by the BLOCK command and other related commands.

The weight function q used to calculate the J-integral is specified by use of the FUNCTION command line. The LINEAR function sets the weight function to 1.0 on the crack front Γ_0 and 0.0 at the edge of the domain C_0, int_radius away from the crack tip. The PLATEAU function, which is the default behavior, sets all values of the weight function to 1.0 that lie within the domain of integration and all values outside of the domain are set to 0.0. This allows for integration over a single ring of elements at the edge of the domain. The third option for the FUNCTION command is PLATEAU_RAMP, which for a single domain will take on the same values as the LINEAR function. However, when there are multiple domains over the radius int_radius, the nth domain will have weight function values of 1.0 over the inner $n-1$ domains and will vary from 1.0 to 0.0 over the outer n^{th} ring of the domain. These functions can be seen graphically in Figure 13-1.

We note that in employing both the PLATEAU and the PLATEAU_RAMP functions, one is effectively taking a line integral at finite radius (albeit different radii). In contrast, the LINEAR option can be viewed as taking the $lim\ \Gamma_0 \rightarrow 0^+$. If the model is a half symmetry model with the symmetry plane on the plane of the crack, the optional SYMMETRY command can be used to include the symmetry conditions in the formation of the integration domains and in the evaluation of the integral. The default behavior is for symmetry to not be used.

The user may optionally specify the time periods during which the J-integral is computed. The ACTIVE PERIODS and INACTIVE PERIODS command lines are used for this purpose. See the Sierra/SM 5.8 User's Guide Section 2.6 for more information about these command lines.
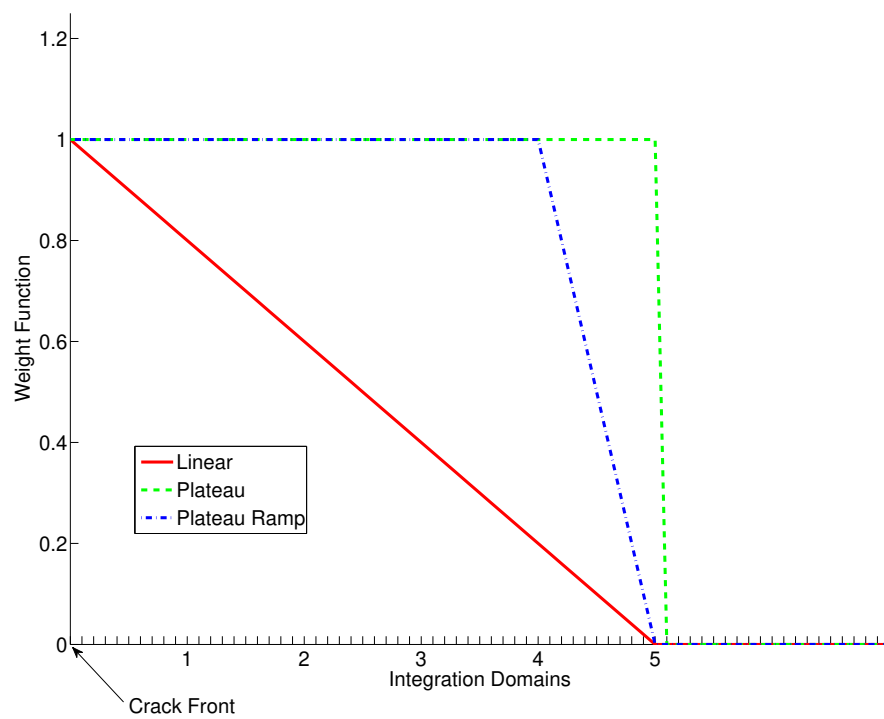
Figure 13-1. Example weight functions for a J-integral integration domain. Weight functions shown for domain 5.

13.3.   Output

Many variables are generated for output when the computation of the *J*-integral is requested. The average value of *J* for each integration domain is available as a global variable, as described in Table 13-1. The point-wise value of *J* at nodes along the crack for each integration domain is available as a nodal variable, as shown in Table 13-2. Element variables such as the Eshelby energy-momentum tensor and fields defining the integration domains are also available, as listed in Table 13-3.

The `DEBUG OUTPUT` command is useful to generate output data for debugging the *J*-integral. If the `DEBUG OUTPUT = ON|OFF(OFF) WITH <integer>num_nodes NODES ON THE CRACK FRONT` line command is set to `ON`, the weight functions, *q*, will be output for each node-based *J* value that is calculated. The user must specify `num_nodes`, which represents the number of nodes along the crack front. An internal check is performed during problem initialization that will verify that the number of nodes specified by the user on the command line matches the number of nodes associated with the crack front.

> **Warning:** Using the `DEBUG OUTPUT` command line can potentially result in an extremely large output file because every node in the integration domain will now compute and store $(NumNodeOnCrackFront + 1) * NumDomains$ weight function vectors. This can also potentially exhaust the available memory on the machine.

13.4.   Required Discretization

In order to enable the correct construction of the test function $\boldsymbol{q}^I$, the hexahedral mesh should be orthogonal to the crack front. An orthogonal mesh will ensure that the elements are not skewed along the crack front. Because these elements will experience large deformation during crack-tip blunting, well-formed elements increase the accuracy of the solution. We note that this capability is not specific to crack front nodes. Any ellipsoidal surface with a constant bias will generate skewed elements.

In addition, an orthogonal mesh will ensure that the location of a point-wise surface integral will be a closest point projection from the crack-tip node. Consequently, any surface integral via a domain integral at a node along the crack front will be most accurate if the specified radius is a minimum. In addition to increasing the accuracy of point-wise evaluations of the J-integral, an orthogonal mesh will also ease the search algorithm for point-wise evaluations. The search is performed along the normal to the crack front. If the mesh is aligned with the normal, the specification of $\boldsymbol{q}$ is straightforward. Misalignment can result in a "checker boarding" of the integration domains which presents the possibility that $\boldsymbol{q}^I$ will always be one and the J-integral will be zero. Future work may generalize the calculation of J^K, but we are currently limited to hexahedra. Given these requirements, we collaborated with the Cubit team to add the capability to generate meshes orthogonal to a surface. The Cubit team implemented the command

```
adjust boundary surface AA snap_to_normal curve A
```

which enables the generated elements on surface `AA` to be "snapped" normal to the curve `A`. For example, one may choose to sweep adjacent squares along the crack front curve `A`. For crack plane surfaces `AA` and `AB` joined by the crack front curve `A`, one would issue the following commands in Cubit

```
surface AA AB scheme map
mesh surface AA
node in curve A position fixed
adjust boundary surface AA snap_to_normal curve A
mesh surface AB
node in curve A position fixed
adjust boundary surface AB snap_to_normal curve A
        fixed curve A
```

to obtain an orthogonal mesh. The next step is to sweep that mesh "up" and "down" from the crack surface. To ensure that Cubit employs a "simple" sweep so that the search is consistent through the direction of the sweep, we use

```
volume ABC  scheme sweep source surface AA target
    surface AC
propagate_bias autosmooth_target off sweep_smooth
    linear
sweep_transform translate
```

for volume `ABC`. Because proper mesh construction ensures accuracy and smoothness in J^K, we encourage analysts to use the `snap_to_normal` and `autosmooth_target off` options.

13.5. Results and History Output

This section lists output variables for *J*-Integral.

- Table 13-1 Global Variables for *J*-Integral
- Table 13-2 Nodal Variables for *J*-Integral
- Table 13-3 Element Variables for *J*-Integral

Table 13-1. Global Variables for *J*-Integral

Variable Name	Type	Comments
j_average_ <jint_name>	Real[]	Average value of the *J*-integral over the crack. Array sized to number of integration domains and numbered from inner to outer domain. <jint_name> is the name of the J INTEGRAL block.

Table 13-2. Nodal Variables for *J*-Integral

Variable Name	Type	Comments
j_<jint_name>	Real[]	Point-wise value of *J*-integral along crack. Array sized to number of integration domains and numbered from inner to outer domain. <jint_name> is the name of the J INTEGRAL block.

Table 13-3. Element Variables for *J*-Integral

Variable Name	Type	Comments
energy_ momentum_ tensor	FullTen36	Energy momentum tensor
integration_ domains_ <jint_name>	Integer[]	Flag indicating elements in integration domains. Set to 1 if in domain, 0 otherwise. Array sized to number of domains and numbered from inner to outer domain. <jint_name> is the name of the J INTEGRAL block.

References

[1] J.D. Eshelby. The force on an elastic singularity. *Philosophical Transactions of the Royal Society of London A*, 244:87–112, 1951. URL http://dx.doi.org/10.1098/rsta.1951.0016.

[2] J.R. Rice. A path independent integral and the approximate analysis of stress concentration by notches and cracks. *Journal of Applied Mechanics*, 35:379–386, 1968.

[3] G.A. Maugin. *Material Inhomogeneities in Elasticity*. Chapman & Hall/CRC, New York, 1993.

[4] J.D. Eshelby. Energy relations and the energy-momentum tensor in continuum mechanics. In M. Kannien, editor, *Inelastic Behavior of Solids*, pages 77–114. McGraw-Hill, New York, 1970.

[5] F.Z. Li, C.F. Shih, and A. Needleman. A comparison of methods for calculating energy release rates. *Engineering Fracture Mechanics*, 21:405–421, 1985. URL http://dx.doi.org/10.1016/0013-7944(85)90029-3.

[6] C.F. Shih, B. Moran, and T. Nakamura. Energy release rate along a three-dimensional crack front in a thermally stressed body. *International Journal of Fracture*, 30:79–102, 1986. URL http://dx.doi.org/10.1007/BF00034019.

[7] ABAQUS. *Theory Manual*, volume Version 6.7. Hibbitt, Karlsson and Sorensen, Providence, RI, 2007.

[8] Y. Ohashi, J.W. Foulk, and A.J. Lindblad. Verification of J integral capability in sierra mechanics. Technical Report SAND2012-8720, Sandia National Laboratories, Albuquerque, NM, 2012.

14.   NONLOCAL REGULARIZATION

> ⚠ **Known Issue:** Each nonlocal block must be uniquely paired with a material. A single material *cannot* have local and nonlocal blocks. Future work will generalize the methodology.

Using material models that employ strain softening to capture the micromechanics of the failure process will result in mesh-dependent solutions. Fundamentally, the partial differential equation is changing character and the problem becomes ill-posed (for both elliptic and hyperbolic systems). There are multiple methodologies to regularize the solution and nonlocality has been employed to converge to a single solution from a multiplicity of solutions.

14.1.   Variational nonlocal method

In the vein of nonlocality, a variational nonlocal method was derived such that one can identify the state variable that controls softening Z and pose a variational principle such that the stored energy is dependent on a nonlocal state variable $\bar{Z}$. At a point, a Lagrange multiplier enforces $\bar{Z} = Z$. When we minimize and discretize, however, we derive an L_2 projection for the "coarser" $\bar{Z}$ and the balance of linear momentum for the "fine" scale. If we assume that the basis functions for the coarser discretization D are constant and discontinuous, we obtain the nonlocal $\bar{Z}$ as a simple volume average of Z.

$$\bar{Z} = \frac{1}{\int_D dV} \int_D Z dV \tag{14.1}$$

In this particular case, less is more. We do not want to recover the mesh-dependent solution inherent in Z with a $\bar{Z}$. Instead, we seek to specify an additional discretization (length scale) independent of the discretization for Z. Because $\bar{Z}$ is just an average, we can consider a coarse domain to be a patch of fine scale elements having volume V that is consistent with a prescribed length scale l where $V = l^3$. For example, one might correlate the mesh dependence in the solution with scalar damage ϕ. The variational nonlocal method would construct a $\bar{\phi}$ for each nonlocal domain D. The stress would then evolve from $\bar{\phi}$ and not ϕ.

Domain decomposition algorithms are invoked to construct coarse scale domains of common volume. For parallel execution, each processor (having nonlocal element blocks) is partitioned during initialization. Nonlocal averages are calculated on the processor and no communication is necessary between processors.

> ⚠ **Warning:** Because nonlocal domains are initially decomposed on each processor, nonlocal geometries will not a) be consistent with different parallel decompositions and b) admit rebalancing. No infrastructure exists to maintain the character of the nonlocal domains during rebalancing.

14.2. Nonlocal partitioning

Because communication in parallel processing scales with the surface area of the domain, we believed that software designed with the intent of limiting communication would be ideal for application to nonlocal regularization. Hence, graph-based (METIS, Zoltan Hypergraph) and geometric (Zoltan Recursive Coordinate Bisection, Zoltan Recursive Inertial Bisection) decomposition algorithms were implemented and available for the analyst. Figure 14-1 illustrates the Zoltan partitioning methodologies for a circular region surrounding a sharp crack tip. We note that non-contiguous domains can occur in graph-based methodologies. For these reasons, ZOLTAN_RCB was selected to be the default partitioning scheme.

Zoltan RCB   Zoltan RIB   Zoltan Hypergraph
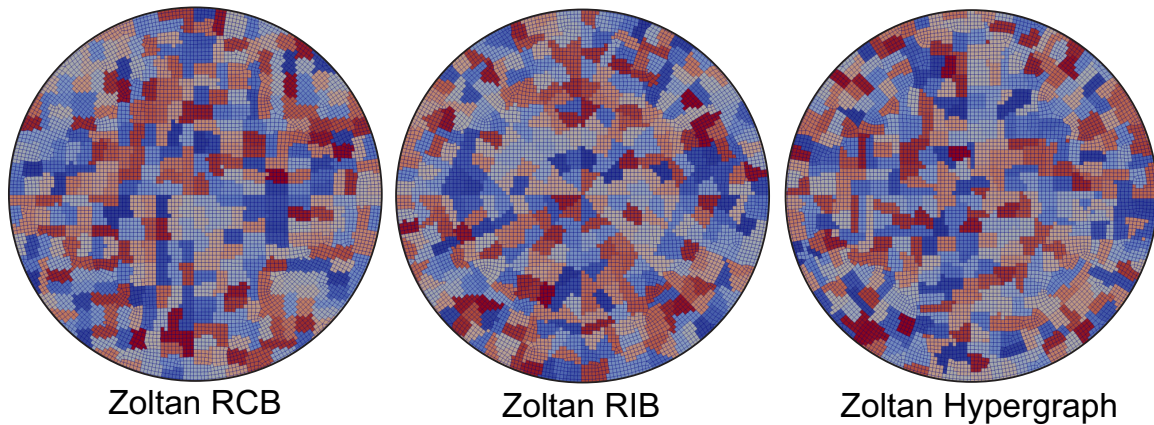
Figure 14-1. Illustration of 400 nonlocal partitions at a sharp crack tip using Zoltan Recursive Coordinate Bisection (RCB), Zoltan Recursive Inertial Bisection (RIB), and Zoltan Hypergraph partitioning methodologies. Note that Zoltan Hypergraph can generate non-contiguous domains. The default partitioning methodology in Sierra is Zoltan RCB.

Initial findings employing geometric partitioning illustrated a sensitivity to domain shape. A re-examination of Figure 14-1 will reveal that the aspect ratios of the domains are significant. Because we are aligning the evolution of a nonlocal variable with the nonlocal domain shape, domains of increasing aspect ratio result in anisotropic evolution. Although other researchers have developed methods for domain decomposition that focus on domain shape [1], we gravitated towards clustering algorithms and the resulting isotropy [2]. Figure 14-2 illustrates the mesh, grid, and result of k-means clustering, a centroid Voronoi Tessellation (CVT). Given a body on processor with mesh size h, we overlay a grid with uniform cell size c. We then find points both inside (red) and outside (blue) the body. After calculating the number of nonlocal volumes N for a body of volume B through $N = B/l^3$, we seed the centroids of the nonlocal domains through Zoltan RCB. K-means clustering of points inside the body evolves the locations of the centroids via Lloyd's algorithm. The algorithm will converge to a CVT, independent of the FE discretization. The tolerance for convergence *tol* is specified as a fraction of the cell size c. The nonlocal domains are then populated by each element's proximity to the nearest CVT centroid. The resulting nonlocal domains are illustrated in Figure 14-2. We note that the nonlocal domain size is only illustrative. Nonlocality in damage, for example, would require a smaller length scale l resulting in a finer discretization of Voronoi polygons.

initial mesh

$h = 125 \ \mu m$

uniform grid for k-means clustering

grid points inside body

$c = 100 \ \mu m$

grid points outside body

nolocal domains (CVTs)

$l = 1300 \ \mu m$
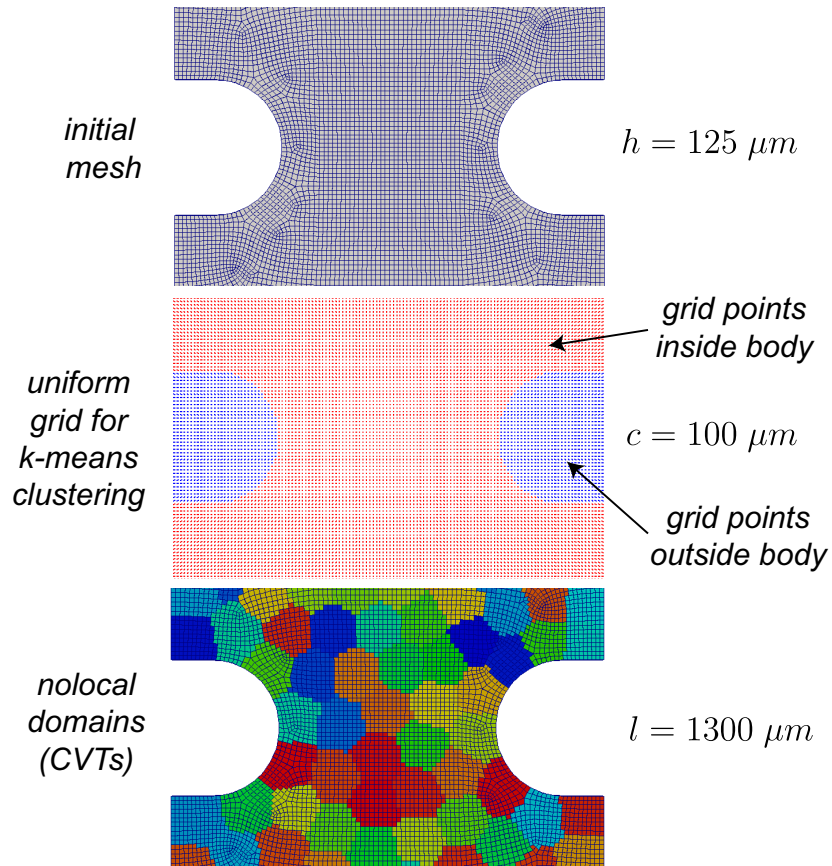
Figure 14-2. Nonlocal domains derive from a Centroidal Voronoi Tessellation (CVT). A partitioned mesh for parallel processing with element size h determines the boundaries of a uniform grid with cell size c. K-means clustering evolves a set of N centroids into a CVT.

14.3.　　　Command summary

In the specification of the block, one can invoke nonlocality in a state variable Z through

```
begin parameters for block block_1
    material ductile_metal
    solid mechanics use model elasto_thermo_visco_poro_plasticity
    section = solid_1
    NONLOCAL REGULARIZATION ON <string>varName WITH LENGTH SCALE =
    <real>length [AND STAGGERING]
    NONLOCAL REGULARIZATION PARTITIONING SCHEME =
    METIS|ZOLTAN_HYPERGRAPH|ZOLTAN_RCB|ZOLTAN_RIB|KMEANS (ZOLTAN_RCB)
#
# Options for k-means clustering
#
    NONLOCAL REGULARIZATION KMEANS CELL SIZE = <real>cell_size
    NONLOCAL REGULARIZATION KMEANS MAXIMUM ITERATIONS = <int>max_iter
    NONLOCAL REGULARIZATION KMEANS TOLERANCE = <real>tol
 end parameters for block block_1
```

where the `varName` is the state variable Z to be averaged and `length` defines the nonlocal volume $V = $ `length`^{3}. The k-means clustering employs a uniform grid having a size `cell_size` and tolerance for convergence `tol`. The maximum number of iterations for k-means clustering is given by `max_iter`. One can output the partitions through the `NONLOCAL_ELEMENT_DOMAIN` element variable. The output of element variables is described in the Sierra/SM 5.8 User's Guide Section 9.3.1.4. In addition, each partition and its volume is noted in the log file. The nonlocal variable $\bar{Z}$ can be output through the element variable `NONLOCAL_varName_AVERAGE` while the local variable Z is output through `varName`. We remind the reader that material points contain both Z and $\bar{Z}$. The energy, stress, and tangent depends on $\bar{Z}$. The constitutive update evolves Z. This process, however, is not employed when using `AND STAGGERING`. In this specific case, local variables are averaged after each time step t_n and used as the initial conditions for t_{n+1}. Strictly speaking, `AND STAGGERING` approximates the variational nonlocal method. A fundamental assumption of the nonlocal method is that one is employing a constitutive model in which the state variable update is separate from the evaluation of the energy, stress, or tangent. Currently, only one model in LAMÉ, `HYPERELASTIC_DAMAGE`, separates these functions. All the other constitutive models, however, update the internal state variables and the stress simultaneously. In an attempt to employ the majority of models that do not adhere to this separation, the `AND STAGGERING` option was implemented and does regularize the failure process. This approximation to the nonlocal method is more accurate for small time steps and may require limited hourglass viscosity to stabilize the evolved perturbations (post bifurcation) in uniform-gradient elements. Although we initially envisioned the `AND STAGGERING` option to be most applicable to explicit dynamics, simulations with nonlocal damage evolution for implicit dynamics have illustrated mesh-independent solutions.

> ⚠️ **Warning:** The tangent generated in Adagio currently derives from Z and is local. Future development will implement a nonlocal, finite-difference tangent.

Warning: Element death for nonlocal domains is work in progress. Additionally this capability will not function with 'death on inversion'.

14.4.   Usage guidelines

The nonlocal length scale `length` l is a material parameter that will set the length scale over which localization will occur. Although the parameterization of l is indirect, it will control the dissipation and should have an experimental basis.

For a typical application, the analyst might

1. Identify a constitutive model that captures the failure process. This might include a local damage model or any model that employs strain softening to facilitate strain localization.

2. Conduct mesh-dependent simulations with bulk elements of size h to understand potential paths for crack initiation and growth in specimen geometries targeted for parameterizing constitutive model parameters.

3. Invoke nonlocality through a nonlocal length scale l. Mesh-independent solutions stem from resolved nonlocal domains where $l > 3h$. The nonlocal domain size should be small compared to the relevant dimensions (features) of the body.

4. Specify `KMEANS` partitioning. Choose the cell size c such that it is small compared to the nonlocal length scale. We recommend $\frac{1}{20} < \frac{c}{l} < \frac{1}{10}$ for the clustering algorithm to sample between ~ 1000 and ~ 8000 points per nonlocal domain and obtain a converged CVT. Please note that memory requirements will scale geometrically with the cell size. One can easily run out of memory on a cluster given decrements in the cell size. Candidate values for convergence and the maximum number of iterations are 0.02 and 256, respectively. Because the clustering process is only performed during the initialization of the simulation, decreased tolerances and increased iterations are not cost prohibitive.

5. Inspect the character of the nonlocal volumes through `NONLOCAL_ELEMENT_DOMAIN` and determine whether or not there are sufficient nonlocal volumes per partition for parallel processing. Because the nonlocal domains are formed on processor, processor boundaries represent nonlocal domain boundaries. One can enable greater smoothness in the nonlocal response through the mitigation of processor boundaries.

6. Incorporate nonlocality into the fitting process. The fitting process may not be unique in that the same far-field response might be obtained from multiple combinations of both l and the material parameters that govern the failure process.

7. Understand the impact of l. If l is too large, the failure process will be "lumped" over a large region resulting in a non-smooth response. Please consider refitting model parameters with smaller values of l (and h) to obtain the localized nature of the failure process.

8. Explore component or system level geometries with nonlocality. Refine the mesh to ensure that the far-field predictions are indeed mesh independent and that the process zone that evolves from the given micromechanics is resolved.

9. Reflect on the fields employed for model parameterization and the fields evolving in component and system level models. Contrast the evolution of local field variables governed by the mesh discretization with the nonlocal variable governed by the CVT discretization. If possible, align field evolution in component/system geometries with field

evolution in specimen geometries. Disparities may drive the need for additional calibration experiments.

Although these usage guidelines have not focused on incorporating stochastic processes, one may sample distributions in material parameters. The inclusion of a method for regularization enables such findings in that the mesh-dependence associated with fracture/failure is not convoluted with a stochastic representation of the micro mechanical process.

References

[1] H. Meyerhenke, B. Monien, and T. Sauerwald. A new diffusion-based multilevel algorithm for computing graph partitions. *Journal of Parallel Distributed Computing*, (69):750–761, 2009.

[2] J. Burkardt, M. Gunzburger, J. Peterson, and R. Brannon. User manual and supporting information for library of codes for centroidal voronoi point placement and associated zeroth, first, and second moment determination. Technical Report SAND2002-0099, Sandia National Laboratories, Albuquerque, NM, 2002.

15. POD

Proper Orthogonal Decomposition (POD) and Explicit Control Modes (ECM, refer to Chapter 7.4.3) should have almost the same name according to the rule

> The name of a method should describe what is done, not how it is done.

A difference from ECM is that POD does not require a coarse finite element mesh. POD is intended for explicit analyses in which the time step is too small, and constructing a coarser mesh for ECM is unfeasible.

Warning: POD is an experimental analysis technique.

15.1. Time Step Control Commands

The larger the time step is, the greater the mass scaling. Part of the Cylinder With Legs test input file invokes POD and controls the way that it works.

```
BEGIN PARAMETERS PRESTO REGION
  #USER TIME STEP = 1.2e-08
END PARAMETERS PRESTO REGION

BEGIN PROPER ORTHOGONAL DECOMPOSITION
  NUMBER OF POD MODES = 21
  SNAPSHOTS INTERVAL = 405
  POD MODES COMPUTATION TIME STEP = 1.e-4
  ENERGY PERCENTAGE = 99.999
  POD FILTER = On
  MODE REFRESH = Off
  #USER FILTER TIME STEP = 1.2327e-08
  USER FILTER TIME STEP = 6.1635e-08 #5X
END PROPER ORTHOGONAL DECOMPOSITION
```

The `USER TIME STEP` overrides the element time step, including any increase in the time step due to POD.

The initialization needs the `NUMBER OF POD MODES` for post-processing. It is the number of fields to allocate that will store POD modes. The way that POD is set up at this time, this is also the number of snapshots and the number of eigenvalue eigenvector pairs.

`SNAPSHOTS INTERVAL` is the number of time steps taken between snapshots used to build the correlation matrix. It is not the number of time steps between adding a POD vector.

`POD MODES COMPUTATION TIME STEP` is the time at which Adagio POD is activated.

`ENERGY PERCENTAGE` is percentage that the sum of the eigenvalues used for the ECM/POD run accounts for with respect to the sum of the total eigenvalue spectrum. It is related to the kinetic energy of the system, but is not the ECM energy percentage [1].

`POD FILTER` activates the high frequency mass scaling. If `OFF`, then the simulation is equivalent to a simulation without ECM/POD.

`MODE REFRESH` This refreshes the number of POD modes throughout the simulation.

`USER FILTER TIME STEP` This is the user defined time step.

In theory the Lanczos algorithm provides the time step, but this has not been implemented.

The Plastic Cylinder test of POD activates the `MODE REFRESH`.

`MODE REFRESH=ON`

Snapshots are stored throughout the simulation. At every `POD MODES COMPUTATION TIME STEP`, 10^{-4} seconds here, the POD modes are updated using the new information.

References

[1] G.J. de Frias, W. Aquino, K.H. Pierson, M.W. Heinstein, and B.W. Spencer. A multiscale mass scaling approach for explicit time integration using proper orthogonal decomposition. *International Journal for Numerical Methods in Engineering*, 97(11):799–818, 2014. doi: 10.1002/nme.4608.

16.   CONFORMING REPRODUCING KERNEL METHOD

> **Warning:** The Conforming Reproducing Kernel method is a beta capability and is still in active development.

16.1.   Overview

The Conforming Reproducing Kernel (CRK) method uses ideas from meshfree methods, such as the reproducing kernel particle method (RKPM), but uses a mesh to provide boundary information and a structure for domain integration [1, 2]. The goal of CRK is to provide the advantages of meshfree methods (robustness in large deformation, greatly reduced sensitivity to discretization quality) while overcoming some of the drawbacks (solution quality near convex boundaries or material interfaces, enforcement of essential boundary conditions, computational efficiency). CRK uses robust integration techniques for nearly incompressible materials [3] and is being used to explore bond-based material failure methodologies [2].

The domain of the CRK kernels are defined by "stars" of elements surrounding a given node. Kernel functions are defined on the stars and then subject to the reproducing conditions to form a basis for the CRK formulation. Examples kernel supports include one-star, two-star and spherical-star. The one-star kernel includes all the elements attached to a node. For tetrahedral elements, the one-star kernel recovers linear finite element shape functions after enforcing linear reproducing conditions. The two-star includes the one-star plus an additional ring of elements (all the elements attached to the one-star). The spherical-star uses all the elements fully or partially contained in a ball about the node, providing the lowest discretization sensitivity of these kernel shapes. All kernel shapes are modified to conform to essential boundaries and material interfaces. Kernel values, and thus shape functions are forced to zero at these locations for nodes that are not on the boundary (i.e., the functions "conform" to the boundary). This results in simple essential boundary condition enforcement and greatly improved solution quality near these locations when compared to RKPM or other common meshfree methods where kernel and shape functions overlap boundaries and interfaces.

Domain integration is accomplished using strain-smoothing techniques [4]. Derivatives are consistently approximated using integration-cell boundary integrals of the field as opposed to direct derivative evaluation at Gaussian quadrature points, as is done in most finite element methods. The strain-smoothing method has proven to be robust for large deformation and an efficient way to integrate consistently. However, as with finite elements and Gaussian quadrature, nearly incompressible materials require additional considerations to avoid displacement and pressure instabilities. CRK uses a sub-cell strain-smoothing approach in an $\overline{\mathbf{F}}$ framework to handle the near-incompressibility constraints [3]. An integration cell is represented as a collection of sub-cells. Each sub-cell provides a unique deviatoric portion of the deformation gradient but

the volumetric portion is the volume weighted average of the all the sub-cells in the cell. Appropriate balancing of the number of cells, sub-cells and nodal degrees-of-freedom are used to provide solutions that are stable in both pressure and displacement.

CRK supports both hexahedral and tetrahedral meshes in general. Tetrahedral meshes are typically preferred due to the relative ease of the meshing procedure. Many features are tailored to tetrahedral inputs.

16.2.   Usage Guidelines

The following block command defines a `CRK SECTION`.

```
BEGIN CRK SECTION <string>crk_name
  # Commands controlling approximation properties
  SHAPE FUNCTION TYPE = SPHERICAL|SPHERICAL_SURFACE_KRONECKER|
                        TWO_STAR|LINEAR_TET (SPHERICAL)
  SUPPORT SIZE = <real>support_size (1.1)
  SUPPORT SIZE TYPE = RELATIVE_LOCAL|RELATIVE_GLOBAL|
                      PHYSICAL (RELATIVE_LOCAL)
  BLOCK CONFORMING = ON|OFF (ON)

  # Commands controlling numerical integration
  FORMULATION = ELEMENT|NODAL_FROM_THEX|TET_FROM_THEX (ELEMENT)
  NUMBER OF INTEGRATION SUBCELLS = <int>num_sub_cells (1)

  # Controls which nodes are DOFs in the model
  THEX ACTIVE NODES = NODES|EDGES|FACES|CENTROIDS (NODES)

  # Controls when post processing occurs for performance
  POST PROCESS EVERY STEP = FALSE|TRUE(TRUE)
END [CRK SECTION <string>crk_name]
```

The `SHAPE FUNCTION TYPE`, `SUPPORT SIZE` `SUPPORT SIZE TYPE`, and `BLOCK CONFORMING` options control the approximation properties of the CRK formulation.

The command `SHAPE_FUNCTION_TYPE` determines the kernel shape used in the CRK analysis. The option `SPHERICAL` is for the spherical-star where the elements that are partially or fully contained within a ball are included in the support. Similarly, the `SPHERICAL_SURFACE_KRONECKER` uses spherical-star kernels for the interior of the domain but with the support restricted to a one-star at surfaces, giving the same shape functions as a linear tetrahedral finite element on the surface and thus possessing the Kronecker delta property at these locations. The option `TWO_STAR` uses kernels with a two-star kernel definition. Finally, the option `LINEAR_TET`, gives linear tetrahedral finite element shape functions, equivalent to using a one-star kernel on a tetrahedral mesh.

The commands `SUPPORT SIZE` and `SUPPORT SIZE TYPE` control the support size and are only valid for `SHAPE FUNCTION TYPE = SPHERICAL` or `SPHERICAL_SURFACE_KRONECKER`. `SUPPORT SIZE TYPE` dictates whether the size is relative to the maximum length of attached edges (`RELATIVE_LOCAL`), relative to the global max element edge length (`RELATIVE_GLOBAL`), or absolute (`PHYSICAL`). The command `SUPPORT SIZE` sets the absolute or relative support size. Relative support sizes near but greater than 1 are recommended.

The command `BLOCK CONFORMING = ON|OFF` provides the option to make the kernel functions to conform to block boundaries or not. Non-conforming will give a behavior similar to traditional meshfree methods.

91

The command FORMULATION determines which type of numerical integration should be performed and supports three options. The option ELEMENT integrates using the elements of the input mesh as the strain-smoothing domains. NODAL_FROM_THEX specifies that nodal integration / strain-smoothing domains should be used and requires tetrahedral elements that are subdivided beforehand into hexahedral elements using the thex command in Cubit. The nodal integration domains are formed by aggregating all of the sub-tet hex elements connected to each node to form the barycentric dual. This results in the SCNI strategy [4]. The option TET_FROM_THEX integrates over the original tetrahedra before subdivision. For the latter two options, the sub-tet hex elements can be used for sub-cells in the $\overline{\mathbf{F}}$ integration technique for improving performance with nearly incompressible materials. The number of sub-cells is specified with NUMBER OF INTEGRATION CELLS. Elements will be grouped into sub-cells using a k-means clustering algorithm if NUMBER OF INTEGRATION CELLS is less than the number of sub-tet hex elements for an integration cell. Each sub-tet hex element will be used as a sub-cell if NUMBER OF INTEGRATION CELLS is equal to or greater than the available elements for an integration cell. The volumetric portion of the deformation gradient is calculated over the cell. These are the nodal or tetrahedral domains for the NODAL_FROM_THEX and TET_FROM_THEX options, respectively.

The THEX ACTIVE NODES command controls which nodes should be active. This command is only valid with FORMULATION = NODAL_FROM_THEX or TET_FROM_THEX where the input mesh has been subdivided using the thex command in cubit. The thex command splits every tetrahedral element into four hexahedral elements, requiring the creation of nodes at the edge and face midpoints and the centroid of the tetrahedron. Any of the nodes in the resulting mesh can be activated (have degrees-of-freedom associated with them). The option NODES activates the nodes that were in the original tet mesh. The options EDGES, FACES and CENTROIDS activates the nodes that the thex process created on the input tetrahedral edges, faces and centroids, respectively.

Given its nonlocal nature, the CRK method typically lacks the Kronecker delta property and thus incurs additional post-processing steps for output purposes. The POST PROCESS EVERY STEP command controls whether post-processing procedures occur every step (TRUE) or during output steps only (FALSE). This improves overall performance as these computations are not necessary for the internal force routine and will decrease the total runtime if set to FALSE (contingent on the output frequency). However, derived output or user routines which depend on CRK variables on every step will not function correctly as a result. As such, the command POST PROCESS EVERY STEP = FALSE should only be used when user output, user functions, and other derived variables do not depend on CRK fields. The default is TRUE.

References

[1] Jacob J. Koester and Jiun-Shyan Chen. Conforming window functions for meshfree methods. *Computer Methods in Applied Mechanics and Engineering*, 347:588 – 621, 2019. ISSN 0045-7825. doi: https://doi.org/10.1016/j.cma.2018.12.042. URL http://www.sciencedirect.com/science/article/pii/S0045782519300015.

[2] Jacob Koester, Michael Tupek, and Scott Mitchell. An agile design-to-simulation workflow using a new conforming moving least squares method. Technical Report SAND2019-11851, Sandia National Laboratories, Albuquerque, NM, 2019.

[3] Georgios Moutsanidis, Jacob Koester, Michael Tupek, Yuri Bazilevs, and Jiun-Shyan Chen. Treatment of near-incompressibility in meshfree and immersed-particle methods. *Computational Particle Mechanics*, 2019.

[4] Jiun-Shyan Chen, Cheng-Tang Wu, Sangpil Yoon, and Yang You. A stabilized conforming nodal integration for galerkin mesh-free methods. *International Journal for Numerical Methods in Engineering*, 50(2):435–466, 2001.

17. MATERIAL MODELS

This chapter describes materials available in Sierra that are currently under development.

17.1. Elastic Orthotropic Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC
    #
    # Elastic constants
    YOUNGS MODULUS = <real>youngs_modulus
    POISSONS RATIO = <real>poissons_ratio
    SHEAR MODULUS = <real>shear_modulus
    BULK MODULUS = <real>bulk_modulus
    LAMBDA = <real>lambda
    TWO MU = <real>two_mu
    #
    # required parameters
    E11 = <real>e11
    E22 = <real>e22
    E33 = <real>e33
    NU12 = <real>nu12
    NU13 = <real>nu13
    NU23 = <real>nu23
    G12 = <real>g12
    G13 = <real>g13
    G23 = <real>g23
    COORDINATE SYSTEM = <string>coordinate_system_name
    #
    # optional parameters
    ANGLE_1_ABSCISSA = <real>angle_1_abscissa
    ANGLE_2_ABSCISSA = <real>angle_2_abscissa
    ANGLE_3_ABSCISSA = <real>angle_3_abscissa
    ROTATION_AXIS_1 = <real>rotation_axis_1
    ROTATION_AXIS_2 = <real>rotation_axis_2
    ROTATION_AXIS_3 = <real>rotation_axis_3
    ANGLE_1_FUNCTION = <string>angle_1_function_name
    ANGLE_2_FUNCTION = <string>angle_2_function_name
    ANGLE_3_FUNCTION = <string>angle_3_function_name
    THERMAL_STRAIN_11_FUNCTION =
      <string>thermal_strain_11_function_name
    THERMAL_STRAIN_22_FUNCTION =
```

```
              <string>thermal_strain_22_function_name
        THERMAL_STRAIN_33_FUNCTION =
              <string>thermal_strain_33_function_name
  END [PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC]
```

The elastic orthotropic model is a Kirchhoff linear elastic constitutive relation that is useful for modeling polymer matrix composite structures undergoing small strains. Required inputs are

1. two of the five general elastic material constants,
2. directional properties, and
3. the coordinate system.

The following is a brief description of each input.

- See Section 5.1.5 for more information on elastic constants input.
- In each material direction, moduli E_{11}, E_{22}, and E_{33} are defined with the E11, E22, and E33 command lines, Poisson's ratios ν_{12}, ν_{13}, and ν_{23} are given by the NU12, NU13, and NU23 command lines, and shear moduli G_{12}, G_{13}, and G_{23} are defined using command lines G12, G13, and G23.
- Principal material direction specification requires a user specified coordinate system given by the COORDINATE SYSTEM command line, as detailed in the Sierra/SM 5.8 User's Guide Section 2.1.8. The material orientation may then be specified using *one* of the following approaches:
  1. Three spatially varying, ordered Euler angle functions are given in terms of global coordinates (X, Y, Z = 1, 2, 3) for rotations about a corresponding local axis:
     - The rotation angle function abscissas x_1, x_2, and x_3, corresponding to the global system (X, Y, Z) = (1, 2, 3), are defined with the ANGLE_1_ABSCISSA, ANGLE_2_ABSCISSA, and ANGLE_3_ABSCISSA command lines, respectively.
     - The axes of rotation i, j, and k are defined with the ROTATION_AXIS_1, ROTATION_AXIS_2, and ROTATION_AXIS_3 command lines, respectively.
     - The rotation angle functions $\theta_i(x_1)$, $\theta_j(x_2)$, and $\theta_k(x_3)$ are defined with the ANGLE_1_FUNCTION, ANGLE_2_FUNCTION, and ANGLE_3_FUNCTION command lines, respectively. Angles are specified in degrees. Section 2.1.5 in the Sierra/SM 5.8 User's Guide provides additional details about defining functions.

     The rotation axis and angle are applied successively in order (1, 2, 3), or (X, Y, Z); that is, each sequential Euler angle rotation defines a new coordinate system in which the subsequent rotation axis and angle are defined.
  2. Alternatively, the angles and axes may be defined directly at each element integration point using INITIAL CONDITION command blocks, as described in the Sierra/SM 5.8 User's Guide Section 7.3. Angles may be specified in degrees using the variables ANGLE_1, ANGLE_2, and ANGLE_3, while axes are given by AXIS_1, AXIS_2, and AXIS_3.
  3. A final option is to initialize the rotation tensor to correspond to the local coordinate system defined in the COORDINATE SYSTEM command line.

  The resulting material directions may be visualized by requesting the element fields MATERIAL_DIRECTION_1, MATERIAL_DIRECTION_2, and MATERIAL_DIRECTION_3 in the results output block.

- Functions to describe normal thermal engineering strains along the principal material directions are given by the `THERMAL_STRAIN_11_FUNCTION`, `THERMAL_STRAIN_22_FUNCTION`, and `THERMAL_STRAIN_33_FUNCTION` command lines. See the Sierra/SM 5.8 User's Guide Section 2.1.5 for guidance on defining functions.

> ⚠️ **Warning:** The `ELASTIC_ORTHOTROPIC` model has not been tested in conjunction with the control stiffness implicit solver block.

Output variables available for this model are listed in Table 17-1.

Table 17-1. State Variables for ELASTIC ORTHOTROPIC Model (Section 17.1)

Index	Name	Variable Description
1	Q_XX	X component of the material 11 unit vector
2	Q_YY	Y component of the material 22 unit vector
3	Q_ZZ	Z component of the material 33 unit vector
4	Q_XY	Y component of the material 11 unit vector
5	Q_YZ	Z component of the material 22 unit vector
6	Q_ZX	X component of the material 33 unit vector
7	Q_YX	X component of the material 22 unit vector
8	Q_ZY	Y component of the material 33 unit vector
9	Q_XZ	Z component of the material 11 unit vector
10	ANGLE_1	Rotation angle about axis 1 (degrees)
11	ANGLE_2	Rotation angle about axis 2 (degrees)
12	ANGLE_3	Rotation angle about axis 3 (degrees)
13	AXIS_1	First axis of rotation
14	AXIS_2	Second axis of rotation
15	AXIS_3	Third axis of rotation
16	TH_STR_XX	Thermal stretch ratio in material 11 direction
17	TH_STR_YY	Thermal stretch ratio in material 22 direction
18	TH_STR_ZZ	Thermal stretch ratio in material 33 direction
19	MAT_STRAIN_XX	Green Lagrange strain in material 11 direction
20	MAT_STRAIN_YY	Green Lagrange strain in material 22 direction
21	MAT_STRAIN_ZZ	Green Lagrange strain in material 33 direction
22	MAT_STRAIN_XY	Green Lagrange strain in material 12 direction
23	MAT_STRAIN_YZ	Green Lagrange strain in material 23 direction
24	MAT_STRAIN_ZX	Green Lagrange strain in material 31 direction
25	MAT_STRESS_XX	2nd P-K stress in material 11 direction
26	MAT_STRESS_YY	2nd P-K stress in material 22 direction
27	MAT_STRESS_ZZ	2nd P-K stress in material 33 direction
28	MAT_STRESS_XY	2nd P-K stress in material 12 direction
29	MAT_STRESS_YZ	2nd P-K stress in material 23 direction
30	MAT_STRESS_ZX	2nd P-K stress in material 31 direction
31	MAT_LOG_STRAIN_XX	Log (Hencky) strain in material 11 direction
32	MAT_LOG_STRAIN_YY	Log (Hencky) strain in material 22 direction
33	MAT_LOG_STRAIN_ZZ	Log (Hencky) strain in material 33 direction
34	MAT_LOG_STRAIN_XY	Log (Hencky) strain in material 12 direction
35	MAT_LOG_STRAIN_YZ	Log (Hencky) strain in material 23 direction
36	MAT_LOG_STRAIN_ZX	Log (Hencky) strain in material 31 direction
37	MAT_CAUCHY_STRESS_XX	Cauchy stress in material 11 direction
38	MAT_CAUCHY_STRESS_YY	Cauchy stress in material 22 direction
39	MAT_CAUCHY_STRESS_ZZ	Cauchy stress in material 33 direction
40	MAT_CAUCHY_STRESS_XY	Cauchy stress in material 12 direction
41	MAT_CAUCHY_STRESS_YZ	Cauchy stress in material 23 direction
42	MAT_CAUCHY_STRESS_ZX	Cauchy stress in material 31 direction

17.2.   Elastic Orthotropic Damage Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_DAMAGE
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> v
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  #
  # Required parameters
  #
  E11    = <real> E₁₁
  E22    = <real> E₂₂
  E33    = <real> E₃₃
  NU12   = <real> v₁₂
  NU13   = <real> v₁₃
  NU23   = <real> v₂₃
  G12    = <real> g₁₂
  G13    = <real> g₁₃
  G23    = <real> g₂₃
  ALPHAD = <real> αᵈ
  BETAD  = <real> βᵈ
  GAMMA0 = <real> γ₀
  J1     = <real> j1
  J2     = <real> j2
  J3     = <real> j3
  CN11   = <real> cn11
  CN22   = <real> cn22
  CN33   = <real> cn33
  CS12   = <real> cs12
  CS13   = <real> cs13
  CS23   = <real> cs23
  COORDINATE SYSTEM = <string> coordinate_system_name
  #
  # Optional parameters
  #
  ANGLE_1_ABSCISSA = <real>angle_1_abscissa
  ANGLE_2_ABSCISSA = <real>angle_2_abscissa
  ANGLE_3_ABSCISSA = <real>angle_3_abscissa
  ROTATION_AXIS_1  = <real>rotation_axis_1
  ROTATION_AXIS_2  = <real>rotation_axis_2
  ROTATION_AXIS_3  = <real>rotation_axis_3
  ANGLE_1_FUNCTION = <string>angle_1_function_name
  ANGLE_2_FUNCTION = <string>angle_2_function_name
```

```
      ANGLE_3_FUNCTION = <string>angle_3_function_name
      E11 FUNCTION    = <string>func_name
      E22 FUNCTION    = <string>func_name
      E33 FUNCTION    = <string>func_name
      NU12 FUNCTION   = <string>func_name
      NU13 FUNCTION   = <string>func_name
      NU23 FUNCTION   = <string>func_name
      G12 FUNCTION    = <string>func_name
      G13 FUNCTION    = <string>func_name
      G23 FUNCTION    = <string>func_name
    END [PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_DAMAGE]
```

The elastic orthotropic damage model is an empirically based constitutive relation that is useful for modeling polymer matrix composite structures. Refer to SAND2013-7257 for a full description of the material model theory and usage.

The command block for an elastic orthotropic damage material starts with the line:

```
    BEGIN PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_DAMAGE
```

and terminates with the line:

```
    END [PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_DAMAGE]
```

In the above command block, the required inputs are: two of the five general elastic material constants, directional properties, and the coordinate system. The following is a brief description of each input.

- The density of the material is defined with the DENSITY command line.
- The Biot's coefficient of the material is defined with the BIOTS COEFFICIENT command line.
- Any two of the following elastic constants are required:
  - Young's modulus is defined with the YOUNGS MODULUS command line.
  - Poisson's ratio is defined with the POISSONS RATIO command line.
  - The bulk modulus is defined with the BULK MODULUS command line.
  - The shear modulus is defined with the SHEAR MODULUS command line.
  - Lambda is defined with the LAMBDA command line.
- The directional moduli E_{11}, E_{22}, and E_{33} are defined with the E11, E22, and E33 command lines.
- The directional Poisson's ratios ν_{12}, ν_{13}, and ν_{23} are defined with the NU12, NU13, and NU23 command lines.
- The directional shear moduli G_{12}, G_{13}, and G_{23} are defined with the G12, G13, and G23 command lines.
- The specification of the principal material directions begins with the selection of a user specified coordinate system given by the COORDINATE SYSTEM command line (see below).
- The damage surface evolution terms are given with the ALPHAD and BETAD command lines.

- The initial damage threshold is defined with the GAMMA0 command line.
- The directional damage surface coefficients with the J1, J2 and J3 command lines.
- The directional normal crack closure coefficients defined with the CN11, CN22 and CN33 command lines.
- The directional shear crack closure coefficients are defined with the CS12, CS13 and CS23 command lines.
- For material orientation definition instructions see the Sierra/SM 5.8 User's Guide Chapter 5.

Warning: The ELASTIC_ORTHOTROPIC_DAMAGE model has not been tested in conjunction with the control stiffness implicit solver block.

17.3.   Elastic Orthotropic Fail Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_FAIL
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Required parameters
  #
  E11 = <real>e11
  E22 = <real>e22
  E33 = <real>e33
  NU12 = <real>nu12
  NU13 = <real>nu13
  NU23 = <real>nu23
  G12 = <real>g12
  G13 = <real>g13
  G23 = <real>g23
  #
  COORDINATE SYSTEM = <string>coordinate_system_name
  #
  # Normal thresholds
  #
  TENSILE_MATRIX_STRENGTH_11     = <real>f1mp
  COMPRESSIVE_MATRIX_STRENGTH_11 = <real>f1mn
  TENSILE_FIBER_STRENGTH_11      = <real>f1fp
  COMPRESSIVE_FIBER_STRENGTH_11  = <real>f1fn
  TENSILE_MATRIX_STRENGTH_22     = <real>f2mp
  COMPRESSIVE_MATRIX_STRENGTH_22 = <real>f2mn
  TENSILE_FIBER_STRENGTH_22      = <real>f2fp
  COMPRESSIVE_FIBER_STRENGTH_22  = <real>f2fn
  TENSILE_MATRIX_STRENGTH_33     = <real>f3mp
  COMPRESSIVE_MATRIX_STRENGTH_33 = <real>f3mn
  TENSILE_FIBER_STRENGTH_33      = <real>f3fp
  COMPRESSIVE_FIBER_STRENGTH_33  = <real>f3fn
  #
  # Shear thresholds
  #
  SHEAR_MATRIX_STRENGTH_12       = <real>s12m
  SHEAR_FIBER_STRENGTH_12        = <real>s12f
  SHEAR_MATRIX_STRENGTH_23       = <real>s23m
```

```
SHEAR_FIBER_STRENGTH_23         = <real>s23f
SHEAR_MATRIX_STRENGTH_13        = <real>s13m
SHEAR_FIBER_STRENGTH_13         = <real>s13f
#
# Fracture parameters
#
TENSILE_FRACTURE_ENERGY_11      = <real>gi1p
COMPRESSIVE_FRACTURE_ENERGY_11 = <real>gi1n
TENSILE_FRACTURE_ENERGY_22      = <real>gi2p
COMPRESSIVE_FRACTURE_ENERGY_22 = <real>gi2n
TENSILE_FRACTURE_ENERGY_33      = <real>gi3p
COMPRESSIVE_FRACTURE_ENERGY_33 = <real>gi3n
SHEAR_FRACTURE_ENERGY_12        = <real>gii12
SHEAR_FRACTURE_ENERGY_23        = <real>gii23
SHEAR_FRACTURE_ENERGY_13        = <real>gii13
CHARACTERISTIC_LENGTH           = <real>l_star
#
# Damage evolution parameters
#
MAXIMUM_COMPRESSIVE_DAMAGE_11  = <real>dmax1n
MAXIMUM_COMPRESSIVE_DAMAGE_22  = <real>dmax2n
MAXIMUM_COMPRESSIVE_DAMAGE_33  = <real>dmax3n
COMPRESSION_COUPLING_FACTOR_11 = <real>a1pn
COMPRESSION_COUPLING_FACTOR_22 = <real>a2pn
COMPRESSION_COUPLING_FACTOR_33 = <real>a3pn
TENSILE_DAMAGE_MODULUS_11       = <real>k1p
COMPRESSIVE_DAMAGE_MODULUS_11  = <real>k1n
TENSILE_DAMAGE_MODULUS_22       = <real>k2p
COMPRESSIVE_DAMAGE_MODULUS_22  = <real>k2n
TENSILE_DAMAGE_MODULUS_33       = <real>k3p
COMPRESSIVE_DAMAGE_MODULUS_33  = <real>k3n
SHEAR_DAMAGE_MODULUS_12         = <real>k12
SHEAR_DAMAGE_MODULUS_23         = <real>k23
SHEAR_DAMAGE_MODULUS_13         = <real>k13
HARDENING_EXPONENT_11           = <real>n11
HARDENING_EXPONENT_22           = <real>n22
HARDENING_EXPONENT_33           = <real>n33
HARDENING_EXPONENT_12           = <real>n12
HARDENING_EXPONENT_23           = <real>n23
HARDENING_EXPONENT_13           = <real>n13
#
# Optional parameters follow
# Orientation Parameters
#
ANGLE_1_ABSCISSA  = <real>angle_1_abscissa
ANGLE_2_ABSCISSA  = <real>angle_2_abscissa
ANGLE_3_ABSCISSA  = <real>angle_3_abscissa
```

```
ROTATION_AXIS_1   = <real>rotation_axis_1
ROTATION_AXIS_2   = <real>rotation_axis_2
ROTATION_AXIS_3   = <real>rotation_axis_3
ANGLE_1_FUNCTION  = <string>angle_1_function_name
ANGLE_2_FUNCTION  = <string>angle_2_function_name
ANGLE_3_FUNCTION  = <string>angle_3_function_name
#
# Coefficient of thermal expansion functions
#
THERMAL_STRAIN_11_FUNCTION = <string>cte11_function_name
THERMAL_STRAIN_22_FUNCTION = <string>cte22_function_name
THERMAL_STRAIN_33_FUNCTION = <string>cte33_function_name
#
# Temperature dependent property functions
#
E11_FUNCTION  = <string>e11_function_name
E22_FUNCTION  = <string>e22_function_name
E33_FUNCTION  = <string>e33_function_name
NU12_FUNCTION = <string>nu12_function_name
NU23_FUNCTION = <string>nu23_function_name
NU13_FUNCTION = <string>nu13_function_name
G12_FUNCTION  = <string>g12_function_name
G23_FUNCTION  = <string>g23_function_name
G13_FUNCTION  = <string>g13_function_name
#
# Strain rate dependent parameters
#
REFERENCE_STRAIN_RATE                = <real>epsdot0
ELASTIC_RATE_COEFFICIENT_11          = <real>ce11
ELASTIC_RATE_COEFFICIENT_22          = <real>ce22
ELASTIC_RATE_COEFFICIENT_33          = <real>ce33
ELASTIC_RATE_COEFFICIENT_12          = <real>ce12
ELASTIC_RATE_COEFFICIENT_23          = <real>ce23
ELASTIC_RATE_COEFFICIENT_13          = <real>ce13
FIBER_STRENGTH_RATE_COEFFICIENT_11   = <real>cf11
FIBER_STRENGTH_RATE_COEFFICIENT_22   = <real>cf22
FIBER_STRENGTH_RATE_COEFFICIENT_33   = <real>cf33
FIBER_STRENGTH_RATE_COEFFICIENT_12   = <real>cf12
FIBER_STRENGTH_RATE_COEFFICIENT_23   = <real>cf23
FIBER_STRENGTH_RATE_COEFFICIENT_13   = <real>cf13
MATRIX_STRENGTH_RATE_COEFFICIENT_11 = <real>cm11
MATRIX_STRENGTH_RATE_COEFFICIENT_22 = <real>cm22
MATRIX_STRENGTH_RATE_COEFFICIENT_33 = <real>cm33
MATRIX_STRENGTH_RATE_COEFFICIENT_12 = <real>cm12
MATRIX_STRENGTH_RATE_COEFFICIENT_23 = <real>cm23
MATRIX_STRENGTH_RATE_COEFFICIENT_13 = <real>cm13
END [PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_FAIL]
```

The elastic orthotropic fail model is an empirically based constitutive relation that is useful for modeling polymer matrix composite structures. Refer to the SAND report by English [1] for a full description of the material model theory and usage.

This model has identical input requirements to the Elastic Orthotropic Model detailed in Section 17.1, supplemented with additional parameters for failure modeling. The following is a brief description of additional inputs required for the Elastic Orthotropic Fail Model.

- The strengths for each component of damage are given by the commands:

```
# Normal thresholds
TENSILE_MATRIX_STRENGTH_11     = <real>f1mp
COMPRESSIVE_MATRIX_STRENGTH_11 = <real>f1mn
TENSILE_FIBER_STRENGTH_11      = <real>f1fp
COMPRESSIVE_FIBER_STRENGTH_11  = <real>f1fn
TENSILE_MATRIX_STRENGTH_22     = <real>f2mp
COMPRESSIVE_MATRIX_STRENGTH_22 = <real>f2mn
TENSILE_FIBER_STRENGTH_22      = <real>f2fp
COMPRESSIVE_FIBER_STRENGTH_22  = <real>f2fn
TENSILE_MATRIX_STRENGTH_33     = <real>f3mp
COMPRESSIVE_MATRIX_STRENGTH_33 = <real>f3mn
TENSILE_FIBER_STRENGTH_33      = <real>f3fp
COMPRESSIVE_FIBER_STRENGTH_33  = <real>f3fn
# Shear thresholds
SHEAR_MATRIX_STRENGTH_12       = <real>s12m
SHEAR_FIBER_STRENGTH_12        = <real>s12f
SHEAR_MATRIX_STRENGTH_23       = <real>s23m
SHEAR_FIBER_STRENGTH_23        = <real>s23f
SHEAR_MATRIX_STRENGTH_13       = <real>s13m
SHEAR_FIBER_STRENGTH_13        = <real>s13f
```

- The fracture energies (energy per unit area) for each plane of damage are given by the commands:

```
# Fracture parameters
TENSILE_FRACTURE_ENERGY_11     = <real>gi1p
COMPRESSIVE_FRACTURE_ENERGY_11 = <real>gi1n
TENSILE_FRACTURE_ENERGY_22     = <real>gi2p
COMPRESSIVE_FRACTURE_ENERGY_22 = <real>gi2n
TENSILE_FRACTURE_ENERGY_33     = <real>gi3p
COMPRESSIVE_FRACTURE_ENERGY_33 = <real>gi3n
SHEAR_FRACTURE_ENERGY_12       = <real>gii12
SHEAR_FRACTURE_ENERGY_23       = <real>gii23
SHEAR_FRACTURE_ENERGY_13       = <real>gii13
CHARACTERISTIC_LENGTH          = <real>l_star
```

  The total energy density dissipated (the area under the stress-strain curve) is given by the fracture energy divided by the characteristic length `l_star`.

- The maximum allowable damage values under compression on each plane are given by the commands:

```
MAXIMUM_COMPRESSIVE_DAMAGE_11  = <real>dmax1n
MAXIMUM_COMPRESSIVE_DAMAGE_22  = <real>dmax2n
```

```
                 MAXIMUM_COMPRESSIVE_DAMAGE_33   = <real>dmax3n
```
- The proportion of tensile damage translating to compressive damage for each of the orthotropic planes are given by the commands:
```
        COMPRESSION_COUPLING_FACTOR_11 = <real>a1pn
        COMPRESSION_COUPLING_FACTOR_22 = <real>a2pn
        COMPRESSION_COUPLING_FACTOR_33 = <real>a3pn
```
- The slopes of the matrix mode damage portion of the stress-strain curve, or damage moduli terms, are given by the commands:
```
        TENSILE_DAMAGE_MODULUS_11       = <real>k1p
        COMPRESSIVE_DAMAGE_MODULUS_11   = <real>k1n
        TENSILE_DAMAGE_MODULUS_22       = <real>k2p
        COMPRESSIVE_DAMAGE_MODULUS_22   = <real>k2n
        TENSILE_DAMAGE_MODULUS_33       = <real>k3p
        COMPRESSIVE_DAMAGE_MODULUS_33   = <real>k3n
        SHEAR_DAMAGE_MODULUS_12         = <real>k12
        SHEAR_DAMAGE_MODULUS_23         = <real>k23
        SHEAR_DAMAGE_MODULUS_13         = <real>k13
```
- Small nonlinearity in the matrix mode damage evolution can be added using the hardening exponents for each of the orthotropic planes via the commands:
```
        HARDENING_EXPONENT_11           = <real>n11
        HARDENING_EXPONENT_22           = <real>n22
        HARDENING_EXPONENT_33           = <real>n33
        HARDENING_EXPONENT_12           = <real>n12
        HARDENING_EXPONENT_23           = <real>n23
        HARDENING_EXPONENT_13           = <real>n13
```
- Strain rate dependence is defined by the commands:
```
        REFERENCE_STRAIN_RATE                  = <real>epsdot0
        ELASTIC_RATE_COEFFICIENT_11            = <real>ce11
        ELASTIC_RATE_COEFFICIENT_22            = <real>ce22
        ELASTIC_RATE_COEFFICIENT_33            = <real>ce33
        ELASTIC_RATE_COEFFICIENT_12            = <real>ce12
        ELASTIC_RATE_COEFFICIENT_23            = <real>ce23
        ELASTIC_RATE_COEFFICIENT_13            = <real>ce13
        FIBER_STRENGTH_RATE_COEFFICIENT_11  = <real>cf11
        FIBER_STRENGTH_RATE_COEFFICIENT_22  = <real>cf22
        FIBER_STRENGTH_RATE_COEFFICIENT_33  = <real>cf33
        FIBER_STRENGTH_RATE_COEFFICIENT_12  = <real>cf12
        FIBER_STRENGTH_RATE_COEFFICIENT_23  = <real>cf23
        FIBER_STRENGTH_RATE_COEFFICIENT_13  = <real>cf13
        MATRIX_STRENGTH_RATE_COEFFICIENT_11 = <real>cm11
        MATRIX_STRENGTH_RATE_COEFFICIENT_22 = <real>cm22
        MATRIX_STRENGTH_RATE_COEFFICIENT_33 = <real>cm33
        MATRIX_STRENGTH_RATE_COEFFICIENT_12 = <real>cm12
        MATRIX_STRENGTH_RATE_COEFFICIENT_23 = <real>cm23
        MATRIX_STRENGTH_RATE_COEFFICIENT_13 = <real>cm13
```
The rate dependence is calculated with respect to the reference strain rate `epsdot0`. The rate coefficients for the purely empirical rate equation in each material direction are given

105

for elastic moduli and failure parameters by the scalar values of the elastic rate coefficients `ceij` and fiber and matrix strength rate coefficients `cfij` and `cmij`.

> ⚠️ **Warning:** The `ELASTIC_ORTHOTROPIC_FAIL` model has not been tested in conjunction with the control stiffness implicit solver block.

Output variables available for this model are listed in the Elastic Orthotropic Model in Tables .

> ⚠️ **Warning:** Strongly rate-dependent models may fare poorly in implicit quasistatic solution. In implicit analyses the rate term used to evaluate the current load step is the rate seen by the model in the previous load step. This may cause the solution to oscillate between high- and low-rate equilibrium states from step to step.

Table 17-2. Additional State Variables for ELASTIC ORTHOTROPIC FAIL Model (Section 17.3). Other state variables associated with the Elastic Orthotropic Model are listed in Table 17-1.

Index	Name	Variable Description
43	R1MP	Damage evolution variable 11, matrix, tension
44	R1FP	Damage evolution variable 11, fiber, tension
45	R1MN	Damage evolution variable 11, matrix, compression
46	R1FN	Damage evolution variable 11, fiber, compression
47	R2MP	Damage evolution variable 22, matrix, tension
48	R2FP	Damage evolution variable 22, fiber, tension
49	R2MN	Damage evolution variable 22, matrix, compression
50	R2FN	Damage evolution variable 22, fiber, compression
51	R3MP	Damage evolution variable 33, matrix, tension
52	R3FP	Damage evolution variable 33, fiber, tension
53	R3MN	Damage evolution variable 33, matrix, compression
54	R3FN	Damage evolution variable 33, fiber, compression
55	D1MP	Normal damage 11, matrix, tension
56	D1FP	Normal damage 11, fiber, tension
57	D1MN	Normal damage 11, matrix, compression
58	D1FN	Normal damage 11, fiber, compression
59	D2MP	Normal damage 22, matrix, tension
60	D2FP	Normal damage 22, fiber, tension
61	D2MN	Normal damage 22, matrix, compression
62	D2FN	Normal damage 22, fiber, compression
63	D3MP	Normal damage 33, matrix, tension
64	D3FP	Normal damage 33, fiber, tension
65	D3MN	Normal damage 33, matrix, compression
66	D3FN	Normal damage 33, fiber, compression
67	D12M	Shear damage 12, matrix
68	D12F	Shear damage 12, fiber
69	D23M	Shear damage 23, matrix
70	D23F	Shear damage 23, fiber
71	D13M	Shear damage 13, matrix
72	D13F	Shear damage 13, fiber
73	ORTHOTROPIC_DAMAGE_XX	Effective and active normal damage 11
74	ORTHOTROPIC_DAMAGE_YY	Effective and active normal damage 22
75	ORTHOTROPIC_DAMAGE_ZZ	Effective and active normal damage 33
76	ORTHOTROPIC_DAMAGE_XY	Effective and active shear damage 12
77	ORTHOTROPIC_DAMAGE_YZ	Effective and active shear damage 23
78	ORTHOTROPIC_DAMAGE_ZX	Effective and active shear damage 31

17.4. Elastic Orthotropic Shell Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_SHELL
  #
  # required parameters
  #
  YOUNGS MODULUS RR = <real> E_RR
  YOUNGS MODULUS SS = <real> E_SS
  YOUNGS MODULUS TT = <real> E_TT
  POISSONS RATIO RS = <real> v_RS
  POISSONS RATIO ST = <real> v_ST
  POISSONS RATIO TR = <real> v_TR
  SHEAR MODULUS RS  = <real> G_RS
  SHEAR MODULUS RT  = <real> G_RT
  SHEAR MODULUS ST  = <real> G_ST
END [PARAMETERS FOR ELASTIC_ORTHOTROPIC_SHELL]
```

The ELASTIC ORTHOTROPIC SHELL model describes the linear elastic response of an orthotropic material where the planar orientation of the principal material directions can be arbitrary. This material uses the shell section orthotropic alignment commands described in the Sierra/SM 5.8 User's Guide Section 6.2.5.1 to define the local RST coordinate system.

The general 3D response of an orthotropic material is given above. For the shell model, the ABC coordinate system is replaced with the RST coordinate system. For plane stress the stiffness is calculated with the constraint that $\sigma_{TT} = 0$. From this constraint the thickness strain, which is used to calculate the thickness change for the shell, is

$$\varepsilon_{TT} = -\frac{1}{1 - v_{RS} v_{SR}} [(v_{RT} + v_{RS} v_{ST}) \varepsilon_{RR} + (v_{ST} + v_{SR} v_{RT}) \varepsilon_{SS}] \tag{17.1}$$

and results in the following stiffness

$$\begin{Bmatrix} \sigma_{RR} \\ \sigma_{SS} \\ \sigma_{RS} \\ \sigma_{ST} \\ \sigma_{TR} \end{Bmatrix} = \begin{bmatrix} \bar{\mathbb{C}}_{RRRR} & \bar{\mathbb{C}}_{RRSS} & 0 & 0 & 0 \\ \bar{\mathbb{C}}_{RRSS} & \bar{\mathbb{C}}_{SSSS} & 0 & 0 & 0 \\ 0 & 0 & 2G_{RS} & 0 & 0 \\ 0 & 0 & 0 & 2G_{ST} & 0 \\ 0 & 0 & 0 & 0 & 2G_{TR} \end{bmatrix} \begin{Bmatrix} \varepsilon_{RR} \\ \varepsilon_{SS} \\ \varepsilon_{TT} \\ \varepsilon_{RS} \\ \varepsilon_{ST} \\ \varepsilon_{TR} \end{Bmatrix} \tag{17.2}$$

where

$$\bar{\mathbb{C}}_{RRRR} = \frac{E_R}{1 - v_{RS} v_{SR}} \quad ; \quad \bar{\mathbb{C}}_{SSSS} = \frac{E_S}{1 - v_{RS} v_{SR}} \quad ; \quad \bar{\mathbb{C}}_{RRSS} = \frac{v_{SR} E_R}{1 - v_{RS} v_{SR}} = \frac{v_{RS} E_S}{1 - v_{RS} v_{SR}} \tag{17.3}$$

In the above command blocks, all the following are required inputs.

- Young's modulus of the orthogonal R, S, and T axes are defined with the `YOUNGS MODULUS RR`, `YOUNGS MODULUS SS` and `YOUNGS MODULUS TT` command lines.
- `POISSONS RATIO RS` defines the strain in the S direction when the material is pulled in the R direction
- `POISSONS RATIO ST` defines the strain in the T direction when the material is pulled in the S direction
- `POISSONS RATIO TR` defines the strain in the R direction when the material is pulled in the T direction
- The shear moduli in each of shear directions are defined with the `SHEAR MODULUS RS`, `SHEAR MODULUS RT`, and `SHEAR MODULUS ST` command lines.

Error messages for the ELASTIC ORTHOTROPIC SHELL model concern input that results in a non-positive definite stiffness. The error messages, and their meanings, are

```
Model parameters chosen so that determinant of stiffness < 0
```

$$1 - v_{RS}^2 \frac{E_S}{E_R} - v_{ST}^2 \frac{E_T}{E_S} - v_{TR}^2 \frac{E_R}{E_T} - 2v_{RS}v_{ST}v_{TR} < 0$$

```
Model parameters chosen so that RS sub-determinant of stiffness < 0
```

$$1 - v_{RS}^2 \frac{E_S}{E_R} < 0$$

```
Model parameters chosen so that ST sub-determinant of stiffness < 0
```

$$1 - v_{ST}^2 \frac{E_T}{E_S} < 0$$

```
Model parameters chosen so that TR sub-determinant of stiffness < 0
```

$$1 - v_{TR}^2 \frac{E_R}{E_T} < 0$$

Warning: In previous releases the `ELASTIC_ORTHOTROPIC_SHELL` model required input parameter `POISSONS RATIO SR`, which could have led to an inconsistent set of parameters. The model also previously did not require `YOUNGS_MODULUS_TT`, `POISSONS RATIO ST`, or `POISSONS RATIO TR`, which are now required parameters in the current version.

For backward compatibility *only*, the original input syntax remains valid. However, the new behavior of the model is to ignore any input value of v_{SR} and compute it automatically as $v_{SR} = v_{RS}(E_{SS}/E_{RR})$. If E_{TT} is not input, it is computed as $(E_{TT} = (E_{RR} + E_{SS})/2$ by default. If no value is input for v_{ST} or v_{TR}, it will default to zero.

For best results *all required values should be input* in future usages of this model.

17.5.   Karagozian and Case Concrete Model

```
BEGIN PARAMETERS FOR MODEL KC_CONCRETE
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  #
  #
  COMPRESSIVE STRENGTH = <real>compressive_strength
  FRACTIONAL DILATANCY = <real>omega
  HARDEN-SOFTEN FUNCTION = <string>harden_soften_function_name
  LAMBDAM = <real>lambda_m
  LAMBDAZ = <real>lambda_z
  MAXIMUM AGGREGATE SIZE = <real>max_aggregate_size
  ONE INCH = <real>one_inch
  PRESSURE FUNCTION = <string>pressure_function_name
  RATE SENSITIVITY FUNCTION = <string>rate_function_name
  SINGLE RATE ENHANCEMENT = <enum>TRUE|FALSE
  TENSILE STRENGTH = <real>tensile_strength
  UNLOAD BULK MODULUS FUNCTION = <string>bulk_function_name
END PARAMETERS FOR MODEL KC_CONCRETE
```

The Karagozian & Case (or K&C) concrete model is an inelasticity model appropriate for approximating the constitutive behavior of concrete. Coupled with appropriate elements for capturing the embedded deformation of reinforcing steel, the K&C concrete model can be used effectively for simulating the mechanical response of reinforced concrete structures. The K&C model has several useful characteristics for estimating concrete response, including strain-softening capabilities, some degree of tensile response, and a nonlinear stress-strain characterization that robustly simulates the behavior of plain concrete. This model is described in detail in [2].

In the above command blocks:

- Consult the Sierra/SM 5.8 User's Guide Chapter 5 for more information on elastic constants input.
- The compressive strength for a uniaxial compression test is defined with the COMPRESSIVE STRENGTH command line.
- The tensile strength for the uniaxial tension test is defined with the TENSILE STRENGTH command line.
- The abscissa of the hardening/softening curve where this curve takes on the value of one is

termed Lambda-M, and it is defined with the `LAMBDAM` command line (see [2], pg. B-3).

- The abscissa of the hardening/softening curve where this curve takes on the value of zero after its peak value has been attained is termed Lambda-Z, and it is defined with the `LAMBDAZ` command line. This parameter should satisfy $\lambda_z > \lambda_m$ (see [2], pg. B-3). This input is Sierra-specific, and differs from the previous PRONTO3D definitions.

- The `SINGLE RATE ENHANCEMENT` parameter indicates whether the rate enhancement of the model should be independent of the sign of the deformation. If this parameter is set to `TRUE`, the same enhancement function is used for both compression and tension. If it is set to `FALSE`, the enhancement function must assign values for both positive and negative values of strain rate (see [2], pg. B-5). This parameter is also Sierra-specific, and is different from the previous PRONTO3D definitions.

- The `FRACTIONAL DILATANCY` is an estimate of the size of the plastic volume strain increment relative to that corresponding to straining in the hydrostatic plane. This value normally ranges from 0.3 to 0.7, and a value of one-half is commonly used in practice.

- The `MAXIMUM AGGREGATE SIZE` parameter provides an estimate of the largest length dimension for the aggregate component of the concrete mix. The American Concrete Institute code [3] includes specifications for maximum aggregate size that are based on member depth and clear spacing between adjacent reinforcement elements. This parameter is also useful in modifying the post-peak fall-off of both compressive and tensile behaviors. A large aggregate size (e.g. 2.5 inches) may result in a rapid drop after reaching peak stress. A small aggregate size (e.g. 0.5 inches) may result in a gradual decline after reaching peak stress.

> **Warning:** Simulations utilizing the K&C model can be subject to damage-failure waves wherein the initiation of element damage can cause a rapid chain reaction of damage that propagates through the entire simulation within a few time steps. This damage- failure wave could be caused by a too-sudden drop in element strength after failure. One method to stabilize the simulation is to lower the value used for `MAXIMUM AGGREGATE SIZE`, allowing for a more gradual post-peak element strength decline.

- The parameter `ONE INCH` provides for conversion to units other than the pounds/inch system commonly used in U.S. concrete venues. This parameter should be set to the equivalent length in the system used for analysis. If centimeters are to be used, for example, then `ONE INCH = 2.54`.

The following functions describe the evolution of material coefficients in this model:

- The function characterizing the enhancement of strength with strain rate is described via the `RATE SENSITIVITY FUNCTION` (see [2], pg. B-3).

> **Warning:** The `RATE SENSITIVITY FUNCTION` command should be used with caution. The implementation appears to overestimate concrete strength in tension, and users are cautioned to provide rate sensitivity function values that have the value of 1.0 for positive (tensile) values of strain rate. These values correspond to no additional strength in tension due to strain rate, and are both physically realistic and conservative.

- The function describing the relationship between pressure and volumetric strain is described via the PRESSURE FUNCTION.
- The function characterizing the relationship between bulk modulus and volumetric strain during unloading is described via the UNLOAD BULK MODULUS FUNCTION.
- The function describing the hardening and softening functions function eta as a function of the material parameters lambda (see LAMBDAM and LAMBDAZ) is defined via the HARDEN-SOFTEN FUNCTION (see [2], pg. B-3). The HARDEN-SOFTEN FUNCTION dictates damage accrual. ETA is a function of LAMBDA. At LAMBDA = ETA = 0, the material is undamaged. At ETA(LAMBDA) = 1, damage = 1; the concrete has reached maximum stress and cannot support more. At ETA(LAMBDA) = 0 (after ETA(LAMBDA) = 1), damage is approaching 2; the concrete is mostly rubble/cracked. At damage = 2, the concrete has fully become rubble/cracked.

The above-listed functions are calculated as follows. The HARDEN-SOFTEN FUNCTION is referenced in (see [2], pg. B-3).

```
RATE SENSITIVITY FUNCTION CALCULATIONS
  #
  # Equation Constants
  #
  delta   = 1/(1+8*fpc/1450) fpc in psi
  beta    = 10**(6*delta-2)
  difmax  = beta*(300/1e-6)**(1/3)
  alpha   = 1/(5+9*fpc/1450) fpc in psi
  gamma   = 10**(6.156*alpha-2)
  difmaxc = gamma*(300/(30e-6)**(1/3))
  #
  # Value Calculations
  #
  -30.e5 difmax
  -3.0e2 difmax
  -100.0 beta*(100/1e-6)**(1/3)
  -10.00 beta*(10/1e-6)**(1/3)
  -1.000 (1/1e-6)**delta
  -0.100 (0.1/1e-6)**delta
  -0.010 (0.01/1e-6)**delta
   0.000 1.0
   0.010 (0.01/30e-6)**(1.026*alpha)
   0.100 (0.1/30e-6)**(1.026*alpha)
   1.000 (1/30e-6)**(1.026*alpha)
   30.00 gamma*(30/30e-6)**(1/3)
   100.0     gamma*(100/30e-6)**(1/3)
   300.0 difmaxc
   30.e5 difmaxc


  PRESSURE FUNCTION CALCULATIONS
```

```
    #
    # Equation Constants
    #
    bulk0 = MATERIAL BULK MODULUS
    p2    = bulk0*0.15e-2
    #
    # Value Calculations
    # volume strain vs pressure
    0.0 0.0
    0.15e-2 p2
    0.27e-2 p2*1.53
    0.43e-2 p2*2.18
    0.6e-2 p2*2.74
    0.8e-2 p2*3.13
    0.197e-1 p2*5.13
    0.89e-1 p2*21.7
    0.1e1 p2*221.9
    0.1e4 p2*221.9


UNLOAD BULK MODULUS FUNCTION CALCULATIONS
    #
    # Equation Constants
    #
    bulk0 = MATERIAL BULK MODULUS
    #
    # Value Calculations
    # volume strain vs bulk modulus
    0.0 bulk0
    0.15e-2 bulk0
    0.27e-2 min(10*bulk0,
              bulk0*(1+10*(0.27e-2 - 0.15e-2)))
    0.43e-2 min(10*bulk0,
              bulk0*(1+10*(0.43e-2 - 0.15e-2)))
    0.60e-2 min(10*bulk0,
              bulk0*(1+10*(0.6e-2 - 0.15e-2)))
    0.80e-2 min(10*bulk0,
              bulk0*(1+10*(0.8e-2 - 0.15e-2)))
    0.197e-1 min(10*bulk0,
              bulk0*(1+10*(0.197e-1 - 0.15e-2)))
    0.89e-1 min(10*bulk0,
              bulk0*(1+10*(0.89e-1 - 0.15e-2)))
    0.1e1 min(10*bulk0,
              bulk0*(1+10*(0.1e1 - 0.15e-2)))
    0.1e4 min(10*bulk0,
              bulk0*(1+10*(0.1e1 - 0.15e-2)))
```

The following are sample values for a concrete with a compressive strength of 7000 psi (base units inch-pounds):

```
RATE SENSITIVITY FUNCTION SAMPLE VALUES
  -30.e5 9.4873
  -3.0e2 9.4873
  -100.0 6.5781
  -10.00 3.0533
  -1.000 1.0201
  -0.100 1.0190
  -0.010 1.0179
   0.000 1.0
   0.010 1.1310
   0.100 1.1874
   1.000 1.2468
   30.00 1.3399
   100.0    2.0015
   300.0 2.8867
   30.e5 2.8867


PRESSURE FUNCTION SAMPLE VALUES
  0.0 0.0
  0.15e-2 3974
  0.27e-2 6080
  0.43e-2 8664
  0.6e-2 10889
  0.8e-2 12439
  0.197e-1 20387
  0.89e-1 86239
  0.1e1 881861
  0.1e4 881861

UNLOAD BULK MODULUS FUNCTION SAMPLE VALUES
  0.0 2649423
  .15e-2 2649423
  .27e-2 2681216
  .43e-2 2723607
  .60e-2 2768647
  .80e-2 2821636
  .197e-1 3131618
  .89e-1 4967669
  .1e1 26494234
  .1e4 26494234

HARDEN-SOFTEN FUNCTION SAMPLE VALUES
(for most concrete strengths)
(damage parameter d also shown for reference)
  0 0.0     # --> d=0
  8e-06 0.85    # --> d=0.25
```

```
2.4e-05 0.97    # --> d=0.6
4e-05 0.99    # --> d=0.8333333333
5.6e-05 1.0    # --> d=1
7.2e-05 0.99    # --> d=1.125
8.8e-05 0.97    # --> d=1.222222222
.00032 0.5    # --> d=1.70212766
.00052 0.1    # --> d=1.805555556
.00057 0.0    # --> d=1.821086262
1.00056 0.0    # --> d=1.999888013
10.00056 0.0    # --> d=1.9999888
1e+10 0.0    # --> d=2
```

17.6. Kayenta Model

Note, many parameters of this model are undocumented.

```
BEGIN PARAMETERS FOR MODEL KAYENTA
  B0        = <real> b0
  B1        = <real> b1
  B2        = <real> b2
  B3        = <real> b3
  B4        = <real> b4
  G0        = <real> g0
  G1        = <real> g1
  G2        = <real> g2
  G3        = <real> g3
  G4        = <real> g4
  RJS       = <real> rjs
  RKS       = <real> rks
  RKN       = <real> rkn
  A1        = <real> a1
  A2        = <real> a2
  A3        = <real> a3
  A4        = <real> a4
  P0        = <real> p0
  P1        = <real> p1
  P2        = <real> p2
  P3        = <real> p3
  CR        = <real> cr
  RK        = <real> rk
  RN        = <real> rn
  HC        = <real> hc
  CTPSF     = <real> ctpsf
  CUTPS     = <real> cutps
  CUTI1     = <real> cuti1
  T1        = <real> t1
  T2        = <real> t2
  T3        = <real> t3
  T4        = <real> t4
  T5        = <real> t5
  T6        = <real> t6
  T7        = <real> t7
  J3TYPE    = <real> j3type
  A2PF      = <real> a2pf
  A4PF      = <real> a4pf
  CRPF      = <real> crpf
  RKPF      = <real> rkpf
  FAIL0     = <real> fail0
  FAIL1     = <real> fail1
```

```
    FAIL2     = <real> fail2
    FAIL3     = <real> fail3
    FAIL4     = <real> fail4
    FAIL5     = <real> fail5
    FAIL6     = <real> fail6
    FAIL7     = <real> fail7
    FAIL8     = <real> fail8
    FAIL9     = <real> fail9
    PEAKI1I   = <real> peaki1i
    STRENI    = <real> streni
    FSLOPEI   = <real> fslopei
    PEAKI1F   = <real> peaki1f
    STRENF    = <real> strenf
    FSLOPEF   = <real> fslopef
    SOFTENING = <real> softening
    IEOSID    = <real> ieosid
    DILATLIM  = <real> dilatlim
    NU        = <real> nu
    YSLOPEI   = <real> yslopei
    YSLOPEF   = <real> yslopef
    CKN01     = <real> ckn01
    VMAX1     = <real> vmax1
    SPACE1    = <real> space1
    SHRSTIFF1 = <real> shrstiff1
    CKN01     = <real> ckn02
    VMAX1     = <real> vmax2
    SPACE1    = <real> space2
    SHRSTIFF1 = <real> shrstiff2
    CKN01     = <real> ckn03
    VMAX1     = <real> vmax3
    SPACE1    = <real> space3
    SHRSTIFF1 = <real> shrstiff3
  END [PARAMETERS FOR MODEL KAYENTA]
```

Kayenta is an outgrowth of the Brannon-Fossum-Strack isotropic geomaterial model that includes features and fitting functions appropriate to a broad class of materials including rocks, rock-like engineered materials (such as concretes and ceramics), and metals. Fundamentally, Kayenta is a computational framework for generalized plasticity models. As such, it includes a yield surface, but the term "yield" is generalized to include any form of inelastic material response including micro-crack growth and pore collapse. Kayenta supports optional anisotropic elasticity associated with ubiquitous joint sets. Kayenta supports optional deformation-induced anisotropy through kinematic hardening (in which the initially isotropic yield surface is permitted to translate in deviatoric stress space to model Bauschinger effects). The governing equations are otherwise isotropic. Because Kayenta is a unification and generalization of simpler models, it can be run using as few as 2 parameters (for linear elasticity) to as many as 40 material and control parameters in the exceptionally rare case when all features are used. Isotropic damage is modeled through loss of stiffness and strength. If ever you are unsure of the value of a parameter, leave it

unspecified so that Kayenta can use an appropriate default. See [4] for a full description of the model, inputs, and output variables.

The command block for a Kayenta material starts with the line:

```
BEGIN PARAMETERS FOR MODEL KAYENTA
```

and terminates with the line:

```
END [PARAMETERS FOR MODEL KAYENTA]
```

In the above command blocks:

- The following are valid parameters for the Kayenta material model. If ever you are unsure of the value of a parameter, leave it unspecified so that Kayenta can use an appropriate default.
  - The initial elastic bulk modulus is defined with the B0 command line.
  - The high pressure coefficient in nonlinear elastic bulk modulus function is defined with the B1 command line.
  - The curvature parameter in nonlinear elastic bulk modulus function is defined with the B2 command line.
  - The coefficient in nonlinear elastic bulk modulus to allow for plastic softening is defined with the B3 command line.
  - The power in bulk modulus softening is defined with the B4 command line.
  - The initial elastic shear modulus is defined with the G0 command line.
  - The coefficient in shear modulus hardening is defined with the G1 command line.
  - The curvature parameter in shear modulus hardening is defined with the G2 command line.
  - The coefficient in shear modulus softening is defined with the G3 command line.
  - The power in shear modulus softening is defined with the G4 command line.
  - The joint spacing is defined with the RJS command line.
  - The joint shear stiffness is defined with the RKS command line.
  - The joint normal stiffness is defined with the RKN command line.
  - The constant term for meridional profile function of ultimate shear limit surface is defined with the A1 command line.
  - The curvature decay parameter in the meridional profile function is defined with the A2 command line.
  - The parameter in the meridional profile function is defined with the A3 command line.
  - The high-pressure slope parameter in meridional profile function is defined with the A4 command line.
  - One third of the elastic limit pressure parameter at onset of pore collapse is defined with the P0 command line.
  - One third of slope of porosity vs pressure crush curve at elastic limit is defined with the P1 command line.
  - The parameter for hydrostatic crush curve is defined with P2 command line.

- The asymptote of the plastic volumetric strain for hydrostatic crush is defined with the `P3` command line.
- The parameter for porosity affecting shear strength is defined with the `CR` command line.
- The triaxial extension strength to compression strength ratio is defined with the `RK` command line.
- The initial shear yield offset [non negative] is defined with the `RN` command line.
- The kinematic hardening parameter is defined with the `HC` command line.
- The tension cut-off value of I1 is defined with the `CUTI1` command line.
- The tension cut-off value of principal stress is defined with the `CUTPS` command line.
- The relaxation time constant 1 is defined with the `T1` command line.
- The relaxation time constant 2 is defined with the `T2` command line.
- The parameter no longer in use. [set to zero] is defined with the `T3` command line.
- The parameter no longer in use. [set to zero] is defined with the `T4` command line.
- The relaxation time constant 5 (stress) is defined with the `T5` command line.
- The relaxation time constant 6 (time) is defined with the `T6` command line.
- The relaxation time constant 7 (1/stress) is defined with the `T7` command line.
- The type of 3rd deviatoric stress invariant function is defined with the `J3TYPE` command line.
- The potential function parameter 1 (default=A2) is defined with the `A2PF` command line.
- The potential function parameter 2 (default=A4) is defined with the `A4PF` command line.
- The potential function parameter 3 (default=CR) is defined with the `CRPF` command line.
- The potential function parameter 4 (default=RK) is defined with the `RKPF` command line.
- The failed speed is defined with the `FSPEED` command line.
- The peak I1 hydrostatic tension strength is defined with the `PEAKI1I` command line.
- The peak (high pressure) shear strength is defined with the `STRENI` command line.
- The initial slope of limit surface at PEAKI1I is defined with the `FSLOPEI` command line.
- `PEAKI1F` is the same as PEAKI1I, but for failed limit surface.
- `STRENF` is the same as STRENI, but for failed limit surface.
- `FSLOPEF` is the same as FSLOPEI, but for failed limit surface.
- The `SOFTENING` command line allows transition of limit surface from intact description to failed description.
- The amount of time that passes with the stress state at the limit surface before the limit surface collapses (i.e., softens) is defined with the `TFAIL` command line.
- The upper limit on plastic volume strain is defined with the `DILATLIM` command line.

17.7.   Shape Memory Alloy

```
BEGIN PARAMETERS FOR MODEL SHAPE_MEMORY_ALLOY
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Thermoelastic properties of two crystallographic phases
  #
  ELASTIC MODULUS AUSTENITE  = <real> E^A
  POISSON RATIO AUSTENITE    = <real> ν^A
  CTE AUSTENITE              = <real> α^A
  ELASTIC MODULUS MARTENSITE = <real> E^M
  POISSON RATIO MARTENSITE   = <real> ν^M
  CTE MARTENSITE             = <real> α^M
  #
  # Phase diagram parameters
  #
  MARTENSITE START                    = <real> M_s
  MARTENSITE FINISH                   = <real> M_f
  AUSTENITE START                     = <real> A_s
  AUSTENITE FINISH                    = <real> A_f
  STRESS INFLUENCE COEFF MARTENSITE   = <real> C^M
  STRESS INFLUENCE COEFF AUSTENITE    = <real> C^A
  #
  # Transformation strain magnitude parameters
  #
  H_MIN           = <real> H_min
  H_SAT           = <real> H_sat
  KT              = <real> k
  SIGMA_CRITICAL  = <real> σ^crit
  #
  # Calibration parameters
  #
  N1          = <real> n_1
  N2          = <real> n_2
  N3          = <real> n_3
  N4          = <real> n_4
  SIGMA STAR  = <real> σ^*
  T0          = <real> θ_0
  #
```

```
# Initial phase conditions
#
XIO                    = <real> $\xi(t=0)$ (0.0)
PRESTRAIN_DIRECTION = <int> $n^{\mathrm{ps}}$ (0)
PRESTRAIN_MAGNITUDE = <real> $\|\varepsilon_{ij}^{\mathrm{tr}}(t=0)\|$ (0.0)
#
END [PARAMETERS FOR MODEL SHAPE_MEMORY_ALLOY]
```

The shape memory alloy (SMA) model is used to describe the thermomechanical response of intermetallics (e.g. NiTi, NiTiCu, NiTiPd, NiTiPt) that can undergo a reversible, diffusionless, solid-to-solid martensitic transformation. Specifically, the materials have a high-symmetry (typically cubic) austenitic crystallographic structure at high temperature and/or low stress. At lower-temperatures and/or high stress the crystallographic structure is transformed to a lower symmetry (typically orthorhombic or monoclinic) martensitic phase. The change in structure and symmetry may be taken advantage of to produce large inelastic strains of $\approx$ 1-8%. Importantly, this class of materials differentiates itself from TRIP steels in that the transformation is *reversible* and a variety of thermomechanical loading paths have been conceived of to take advantage of this behavior. A notable application of these materials is as an actuator in smart, morphing structures.

Phenomenologically, the macroscopic behavior of SMAs is typically discussed in effective stress-temperature space via a *phase diagram* like in Figure 17-1. The four lines denoted M_s, M_f, A_s, and A_f indicate the martensitic start, martensitic finish, austenitic start, and austenitic finish transformation surfaces. *Forward transformation* (from an austenitic to a martensitic state) is described by the martensitic start and finish surfaces. Specifically, the former refers to the thermomechanical conditions at which transformation will initiate while the latter corresponds to complete transformation. The difference between the two surfaces is associated with internal hardening effects due to microstructure (i.e. texture, back stresses). Transformation from martensite to austenite is referred to as *reverse* and is characterized by the austenitic start and finish surfaces. Detailed discussion of the crystallography and phenomenology may be found in [5, 6][1].

Two responses characteristic of SMAs may also be represented via the phase diagram. These are the *actuation response* and the *pseudoelastic* (often referred to as *superelastic* in the literature) responses. The first (actuation) is indicated by path "*A*" in Figure 17-1. In this case, a mechanical bias load is applied to the SMA and the material is then thermally cycled through forward and reverse transformation. The resulting transformation first produces and then removes the large transformation strains of SMAs and is commonly used for (surprisingly) actuation applications. At higher temperatures ($T > A_f$), mechanical loading may be used induce forward and, upon unloading, reverse transformation as indicated in path "*B*" of Figure 17-1. Through such a cycle, a distinctive flag shape in the stress strain response is observed through which large amounts of energy may be dissipated while producing no permanent deformations. As such, this loading path

---

[1]In the martensitic configuration, the crystallographic structure can either self-accommodate in a *twinned* configuration producing no macroscopic inelastic strain or an internal or external stress field may be used to *detwin* the microstructure thereby producing the desired inelastic strain. For simplicity, this distinction is bypassed in this brief text and the interested reader should consult the referenced works.
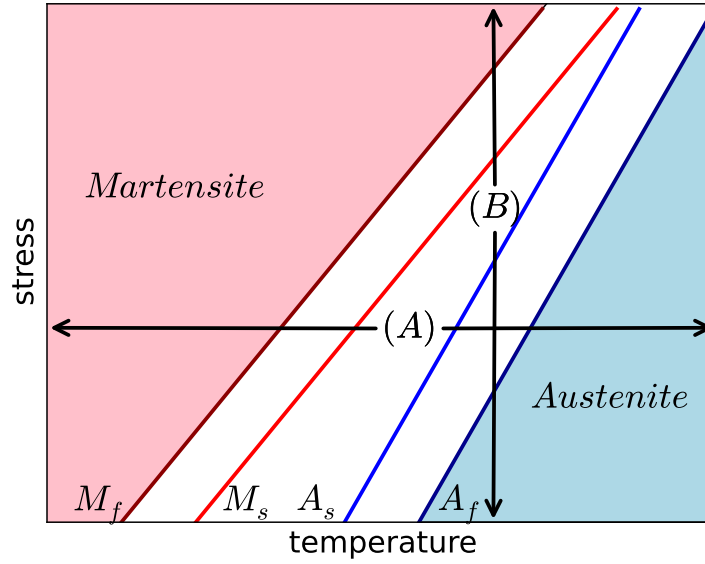
Figure 17-1. Representative phase diagram of shape memory alloys highlighting characteristic loading paths ((A) and (B)), transformation surfaces, and phases.

is often considered for vibration isolation or damping applications.

In LAMÉ, the response of SMAs is described by the phenomenological model of Lagoudas and coworkers [7]. This model was motivated by actuator applications and it describes the inelastic deformation associated with martensitic transformation through two internal state variables – the scalar martensitic volume fraction, ξ, and tensorial transformation strain tensor, $\varepsilon_{ij}^{\mathrm{tr}}$. Before proceeding it should be noted that the *structural* response of SMA specimens and components exhibit a rate dependency associated with the strong thermomechanical coupling of SMAs. Specifically, the transformation process gives off/absorbs large amounts of energy via the *latent heat of transformation*. The rate dependence observed is a result of the characteristic time scale associated with thermal transport of this heat. In pure mechanical analyses (like Sierra/SM), this means quasistatics loadings are typically considered (a strain rate of $\approx 1x10^{-4}$ and/or heating/cooling rate of $\approx 2°C/\mathrm{min}$). Formulations accounting for the full coupling have been developed but require more complex implementations.

To begin, the model assumes an additive decomposition of the total, elastic, thermal, and transformation deformation (strain) rates respectively denoted by D_{ij}, D_{ij}^{e}, D_{ij}^{th} and D_{ij}^{tr} producing a total deformation rate of the form,

$$D_{ij} = D_{ij}^{\mathrm{e}} + D_{ij}^{\mathrm{th}} + D_{ij}^{\mathrm{tr}}. \qquad (17.4)$$

With respect to the thermoelastic deformations, it is noted that the different crystallographic phases have different thermoelastic constants. Previous studies have demonstrated that a rule of mixtures on the compliance and other material properties of the form,

122

$$\mathbb{S}_{ijkl} = \mathbb{S}^A_{ijkl} + \xi \left(\mathbb{S}^M_{ijkl} - \mathbb{S}^A_{ijkl} \right) = \mathbb{S}^A_{ijkl} + \xi \Delta \mathbb{S}_{ijkl}, \tag{17.5}$$

$$\alpha_{ij} = \alpha^A \delta_{ij} + \xi \left(\alpha^M \delta_{ij} - \alpha^A \delta_{ij} \right) = \alpha^A \delta_{ij} + \xi \Delta \alpha \delta_{ij}, \tag{17.6}$$

in which $\mathbb{S}_{ijkl}$ and α_{ij} are the current effective compliance and coefficient of thermal expansion and the superscripts "A" and "M" denote thermoelastic properties in the austenitic and martensitic configuration. The symbol "Δ" is used to indicate the difference in a property between the martensitic and austenitic phases while δ_{ij} is the Kronecker delta. Isotropy is assumed for all these properties and the compliances are determined via the definition of elastic moduli and Poisson's ratio of the two phases – E^A, E^M, v^M, and v^M. The two Poisson ratios are often the same and take typical values for metals ($v^A \approx v^M \approx 0.3$) while the elastic moduli can differ by a factor of more than two. For instance the austenitic modulus of NiTi is typically given as ≈ 70 GPa while the martensitic one is ≈ 30 GPa[2]. Importantly, this difference means that the thermoelastic properties and corresponding deformations vary with transformation. As such, the corresponding rates of deformation are given as,

$$D^{\mathrm{e}}_{ij} = \dot{\xi} \Delta \mathbb{S}_{ijkl} \sigma_{kl} + \mathbb{S}_{ijkl} \, \overset{\circ}{\sigma}_{kl}, \tag{17.7}$$

$$D^{\mathrm{th}}_{ij} = \dot{\xi} \Delta \alpha \delta_{ij} (\theta - \theta_0) + \alpha \delta_{ij} \dot{\theta}, \tag{17.8}$$

where θ and θ_0 are the current and reference temperature and σ_{ij} is the symmetric Cauchy stress. Note, in using the SMA model a temperature field *must* be defined. The stress rate may then be shown to be,

$$\overset{\circ}{\sigma}_{ij} = \mathbb{C}_{ijkl} \left(D_{kl} - \alpha \delta_{kl} \dot{theta} - \dot{\xi} (\Delta \mathbb{S}_{klmn} \sigma_{mn} + \Delta \alpha \delta_{kl} (\theta - \theta_0)) - D^{\mathrm{tr}}_{kl} \right), \tag{17.9}$$

with $\mathbb{C}_{ijkl}$ being the current stiffness tensor defined as $\mathbb{C}_{ijkl} = \mathbb{S}^{-1}_{ijkl}$.

To describe the transformation strain evolution, it is assumed that these deformations evolve with (and only with) the martensitic volume fraction, ξ. The corresponding flow rule is given as,

$$D^{\mathrm{tr}}_{ij} = \dot{\xi} \Lambda_{ij}, \tag{17.10}$$

and Λ_{ij} is the transformation direction tensor assumed to be of the form,

$$\Lambda_{ij} = \begin{cases} H^{\mathrm{cur}} (\bar{\sigma}_{vM}) \frac{3}{2} \frac{s_{ij}}{\bar{\sigma}_{vM}} & \dot{\xi} \geq 0 \\ \frac{\varepsilon^{\mathrm{tr}-rev}_{ij}}{\xi^{rev}} & \dot{\xi} < 0 \end{cases}. \tag{17.11}$$

In (17.11), H^{cur} is the transformation strain magnitude that is dependent on the von Mises effective stress, $\bar{\sigma}_{vM}$, and s_{ij} is the deviatoric stress. With forward transformation defined in this

---

[2]Given the lower symmetry of the martensitic phase the determination of an isotropic elastic modulus can vary with characterization methodology. In this case, the apparent elastic modulus measured from macroscopic thermoelastic tests should be used.

way, it is assumed that deformation is shear-based and follows a J_2 like flow direction. For reverse transformation ($\dot{\xi} < 0$), the postulated form is utilized to ensure complete recovery of transformation strains with martensitic volume fraction. In other words, all transformation strain components are zero-valued at $\xi = 0$. Without enforcing this condition in this way, non-proportional loading paths could be constructed producing a non-zero transformation strain when the material is austenitic. The transformation strain at load reversal, $\varepsilon_{ij}^{\mathrm{tr}-rev}$, and martensitic volume fraction at load reversal, ξ^{rev}, are then tracked (via the implementation) and used for this purpose.

The transformation strain magnitude, H^{cur}, is a function of the von Mises effective stress ($\bar{\sigma}_{vM}$) and is introduced to incorporate detwinning effects without introducing an additional internal state variable complicating the model. Specifically, at low stress values, this function returns a minimum value. If the microstructure is self-accommodated this value will be zero. A decaying exponential is used such that as the stress increases the value of the strain magnitude becomes that of the maximum value incorporating both crystallographic and texture effects. The given functional form is,

$$H^{\mathrm{cur}} = \begin{cases} H_{\mathrm{min}} & \bar{\sigma}_{vM} \leq \sigma_{\mathrm{crit}} \\ H_{\mathrm{min}} + (H_{\mathrm{sat}} - H_{\mathrm{min}})\left(1 - \exp(-k(\bar{\sigma}_{vM} - \sigma_{\mathrm{crit}}))\right) & \bar{\sigma}_{vM} > \sigma_{\mathrm{crit}} \end{cases}, \qquad (17.12)$$

where H_{min}, H_{sat}, k, and σ_{crit} are model parameters giving the minimum transformation strain magnitude, maximum transformation strain magnitude, exponential fitting parameter governing the transition zone, and critical stress values (in some ways analogous to the detwinning stress).

The evolution of martensitic transformation process is governed by a transformation function serving an analogous role to the yield function in plasticity. This function is given by,

$$f\left(\sigma_{ij},\theta,\xi\right) = \pm\phi\left(\sigma_{ij},\theta,\xi\right) - \bar{\sigma}\left(\sigma_{ij}\right), \qquad (17.13)$$

with ϕ begin the thermodynamic driving force for transformation and $\bar{\sigma}$ the critical value. The $\pm$ is used to denote either forward (+) or reverse (−) transformation. This transformation function and the associated forms are derived from continuum thermodynamic considerations and the details of that process are neglected here for brevity but may be found in [7]. The functional forms of these variables are given as,

$$\begin{aligned} \phi\left(\sigma_{ij},\theta,\xi\right) &= \sigma_{ij}\Lambda_{ij} + \frac{1}{2}\sigma_{ij}\Delta\mathbb{S}_{ijkl}\sigma_{kl} + \sigma_{ij}\Delta\alpha\delta_{ij}(\theta - \theta_0) + \rho\Delta s_0\theta - \rho\Delta u_0 - f^t(\xi), \\ \bar{\sigma}\left(\sigma_{ij}\right) &= \sigma_0 + D\sigma_{ij}\Lambda_{ij}, \end{aligned} \qquad (17.14)$$

in which $\rho\Delta s_0$ and $\rho\Delta u_0$ are the differences in reference entropy and internal energy of the two phases, D is a calibration parameter intended to capture variations in dissipation with stress, and $f^t(\xi)$ is the hardening function. With respect to this latter term, empirical observations were used to arrive at a postulated form of,

$$f^t(\xi) = \begin{cases} \frac{1}{2}a_1\left(1 + \xi^{n_1} - (1-\xi)^{n_2}\right) + a_3 & \dot{\xi} \geq 0 \\ \frac{1}{2}a_2\left(1 + \xi^{n_3} - (1-\xi)^{n_4}\right) - a_3 & \dot{\xi} < 0 \end{cases}, \qquad (17.15)$$

124

with a_1, a_2, and a_3 being fitting parameters and n_1, n_2, n_3, and n_4 are exponents fit to match the smooth transformation from elastic to inelastic deformations at the start of forward, end of forward, start of reverse, and end of reverse transformation respectively.

Before proceeding, one final note should be given in regards to calibration. Specifically, some of the model parameters just listed (a_1, a_2, a_3, D, σ_0, $\rho\Delta s_0$ and $\rho\Delta u_0$) are not easily identified or conceptualized in terms of common thermomechanical experiments. Some easily identifiable parameters (M_s, M_f, A_s, and A_f), however, are not evident in the theoretical formulation. Conditions associated with these terms and some physical constraints may be used to determine the model parameters in terms of these more accessible properties. These relations are,

$$\rho\Delta s_0 = \frac{-2\left(C^M C^A\right)\left[H^{\text{cur}}(\sigma) + \sigma\frac{\partial H^{\text{cur}}}{\partial\sigma} + \sigma\left(\frac{1}{E^M} - \frac{1}{E^A}\right)\right]}{C^M + C^A}\Big|_{\sigma=\sigma^*}, \qquad (17.16)$$

$$D = \frac{\left(C^M - C^A\right)\left[H^{\text{cur}}(\sigma) + \sigma\frac{\partial H^{\text{cur}}}{\partial\sigma} + \sigma\left(\frac{1}{E^M} - \frac{1}{E^A}\right)\right]}{\left(C^M + C^A\right)\left[H^{\text{cur}}(\sigma) + \sigma\frac{\partial H^{\text{cur}}}{\partial\sigma}\right]}\Big|_{\sigma=\sigma^*}, \qquad (17.17)$$

$$a_1 = \rho\Delta s_0\left(M_f - M_s\right), \qquad a_2 = \rho\Delta s_0\left(A_s - A_f\right), \qquad (17.18)$$

$$a_3 = -\frac{a_1}{4}\left(1 + \frac{1}{n_1 + 1} - \frac{1}{n_2 + 1}\right) + \frac{a_2}{4}\left(1 + \frac{1}{n_3 + 1} - \frac{1}{n_4 + 1}\right), \qquad (17.19)$$

$$\rho\Delta u_0 = \frac{\rho\Delta s_0}{2}\left(M_s + A_f\right), \qquad \sigma_0 = \frac{\rho\Delta s_0}{2}\left(M_s - A_f\right) - a_3, \qquad (17.20)$$

in which σ^* is the scalar stress measure in which the calibration is performed at. For additional discussion on the characterization of SMAs and calibration of this model, the user is referred to [8, 9].

In the command blocks that define the Shape Memory Alloy model:

- See the Sierra/SM 5.8 User's Guide Section 5.1.5 for more information on elastic constants input. Although the thermoelastic constants of the phases are defined separately, the definition of these constants in this form is necessary for the global solver. Typical values of the phases should be applied.

- The isotropic elastic moduli of the austenitic (E^A) and martensitic phases (E^M) are defined with the `ELASTIC MODULUS AUSTENITE` and `ELASTIC MODULUS MARTENSITE` command lines, respectively. Note, alternative elastic constants (e.g. bulk or shear moduli) may not be used.

- The isotropic Poisson's ratio of the austenitic (ν^A) and martensitic phases (ν^M) are defined with the `POISSON RATIO AUSTENITE` and `POISSON RATIO MARTENSITE` command lines, respectively. Note, alternative elastic constants (e.g. lame constant) may not be used.

- The isotropic coefficient of thermal expansion of the austenitic (α^A) and martensitic phases (α^M) are defined with the `CTE AUSTENITE` and `CTE MARTENSITE` command lines, respectively. Note, given the phase and history dependence of the material thermal expansion, the use of artificial or thermal strain functions may not lead to desired results. The use of these constants in encouraged instead.

- The zero stress, smooth transformation temperatures corresponding to the start and end of forward transformation (martensitic start M_s and finish M_f, respectively) and start and end of reverse transformation (austenitic start A_s and finish A_f, respectively) are given by the (in order) command lines `MARTENSITE START`, `MARTENSITE FINISH`, `AUSTENITE START`, and `AUSTENITE FINISH`.

- The stress influence coefficients giving the slope of the forward and reverse transformation surfaces (C^M and C^A, respectively) are given by the `STRESS INFLUENCE COEFF MARTENSITE` and `STRESS INFLUENCE COEFF AUSTENITE`, respectively.

- The stress dependence of the transformation strain magnitude requires four coefficients. These are the minimum transformation strain magnitude ($H_{\min}$), the saturation (or maximum) magnitude (H_{sat}), exponential fitting coefficient (k), and critical effective stress value below which the magnitude is minimum (σ_{crit}). These parameters are defined via the `H_MIN`, `H_SAT`, `KT`, and `SIGMA_CRITICAL` command lines, respectively.

- The smooth hardening fitting constants n_1, n_2, n_3, and n_4 correspond to the degree of smoothness (essentially how gradual the transformation is) of the martensitic start, martensitic finish, austenitic start, and austenitic finish transformation surfaces. They are given by the `N1`, `N2`, `N3`, and `N4` command lines, respectively, and should take values $0 < n_i \leq 1$.

- The stress level of transformation at which calibration is performed is denoted by σ^* and given by the command line `SIGMA STAR`. For thermally induced transformation this corresponds to the bias stress level while in pseudo-elastic loadings it corresponds to the stress level at which the material is roughly evenly split between martensite and austenite.

- The zero-strain reference temperature is denoted θ_0 and prescribed via the `T0` command line.

- Three optional parameters describing the initial state of the material may be input. These parameters are intended for the case in which the material is initially martensite to allow for initial heating and transformation recovery. The first is the initial martensitic volume fraction, $\xi(t = 0)$, input via the `XI0` command line. If this parameter is not specified the default value is 0.0 representative of an austenitic material. A value between 0.0 and 1.0 may be entered to initialize the material to partially (or fully) martensitic. Corresponding initial transformation strains may be entered via the `PRESTRAIN_DIRECTION` (n^{ps}) and `PRESTRAIN_MAGNITUDE` $\left(\|\varepsilon_{ij}^{\mathrm{tr}}(t = 0)\|\right)$ commands. The first (an integer between one and three) gives the direction of transformation (in global Cartesian space) and the magnitude of the inelastic strain in that direction is given by a fraction (between 0 and 1) of H_{sat} via the second `PRESTRAIN_MAGNITUDE` line. As the transformation strain tensor is deviatoric, the other two directions are specified by preserving that the tensor be trace less. Note, the `PRESTRAIN_DIRECTION` and `PRESTRAIN_MAGNITUDE` cannot be specified without a non-zero `XI0` definition.

- Thermal strains functions and commands in Sierra should not be used in conjunction with the `shape_memory_alloy` model.

Output variables available for this model are listed in Table 17-3.

Table 17-3. State Variables for SHAPE MEMORY ALLOY Model (Section 17.7)

Name	Description
MVF	martensitic volume fraction, ξ
TransStrain	transformation strain tensor, $\varepsilon_{ij}^{\mathrm{tr}}$

17.8.   Linear Elastic

```
BEGIN PARAMETERS FOR MODEL LINEAR_ELASTIC
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
END [PARAMETERS FOR MODEL LINEAR_ELASTIC]
```

The LINEAR_ELASTIC material is used for modeling infinitesimal strain elastic response. Generally this model is used for code verification work when comparing to infinitesimal strain solutions. This differs slightly from the standard ELASTIC model which is formulated for general finite strain.

17.9.   Elastic Three-Dimensional Anisotropic Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_3D_ANISOTROPIC
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> v
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Material coordinates system definition
  #
  COORDINATE SYSTEM               = <string>  coordinate_system_name
  DIRECTION FOR ROTATION          = <real>    1|2|3
  ALPHA                           = <real>    α₁ (degrees)
  SECOND DIRECTION FOR ROTATION   = <real>    1|2|3
  SECOND ALPHA                    = <real>    α₂ (degrees)
  #
  # Required parameters
  #
  STIFFNESS MATRIX 11 = <real> C₁₁
  STIFFNESS MATRIX 22 = <real> C₂₂
  STIFFNESS MATRIX 33 = <real> C₃₃
  STIFFNESS MATRIX 12 = <real> C₁₂
  STIFFNESS MATRIX 13 = <real> C₁₃
  STIFFNESS MATRIX 23 = <real> C₂₃
  STIFFNESS MATRIX 44 = <real> C₄₄
  STIFFNESS MATRIX 55 = <real> C₅₅
  STIFFNESS MATRIX 66 = <real> C₅₅
  STIFFNESS MATRIX 45 = <real> C₄₅
  STIFFNESS MATRIX 46 = <real> C₄₆
  STIFFNESS MATRIX 56 = <real> C₅₆
  STIFFNESS MATRIX 14 = <real> C₁₄
  STIFFNESS MATRIX 15 = <real> C₁₅
  STIFFNESS MATRIX 16 = <real> C₁₆
  STIFFNESS MATRIX 24 = <real> C₂₄
  STIFFNESS MATRIX 25 = <real> C₂₅
  STIFFNESS MATRIX 26 = <real> C₂₆
  STIFFNESS MATRIX 34 = <real> C₃₄
  STIFFNESS MATRIX 35 = <real> C₃₅
  STIFFNESS MATRIX 36 = <real> C₃₆
  #
  # Thermal strain functions
  #
```

```
THERMAL STRAIN 11 FUNCTION = <real> ε₁₁ᵗʰ(θ)
THERMAL STRAIN 22 FUNCTION = <real> ε₂₂ᵗʰ(θ)
THERMAL STRAIN 33 FUNCTION = <real> ε₃₃ᵗʰ(θ)
THERMAL STRAIN 12 FUNCTION = <real> ε₁₂ᵗʰ(θ)
THERMAL STRAIN 23 FUNCTION = <real> ε₂₃ᵗʰ(θ)
THERMAL STRAIN 13 FUNCTION = <real> ε₁₃ᵗʰ(θ)
#
END [PARAMETERS FOR MODEL ELASTIC_3D_ANISOTROPIC]
```

The ELASTIC 3D ANISOTROPIC model is an extension of the ELASTIC model which allows for full anisotropy in both the material stiffness and thermal expansion. Each stiffness component is labeled with i and j indices which correspond to the components of stress and strain vectors in contracted notation,

$$
\begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{23} \\ \sigma_{13} \end{bmatrix} =
\begin{bmatrix}
C_{11} & C_{12} & C_{13} & C_{14} & C_{15} & C_{16} \\
C_{12} & C_{22} & C_{23} & C_{24} & C_{25} & C_{26} \\
C_{13} & C_{23} & C_{33} & C_{34} & C_{35} & C_{36} \\
C_{14} & C_{24} & C_{34} & C_{44} & C_{45} & C_{46} \\
C_{15} & C_{25} & C_{35} & C_{45} & C_{55} & C_{56} \\
C_{16} & C_{26} & C_{36} & C_{46} & C_{56} & C_{66}
\end{bmatrix}
\begin{bmatrix} \epsilon_{11}^{\text{mech}} \\ \epsilon_{22}^{\text{mech}} \\ \epsilon_{33}^{\text{mech}} \\ \epsilon_{12}^{\text{mech}} \\ \epsilon_{23}^{\text{mech}} \\ \epsilon_{13}^{\text{mech}} \end{bmatrix} ,
$$

where the stress and strain components are with respect to principal material directions. The thermal strains are defined in a similar manner,

$$
\epsilon = \epsilon^{\text{mech}} + \epsilon^{\text{th}}, \epsilon^{\text{th}} = [\epsilon_{11}^{\text{th}}(\theta) \, \epsilon_{22}^{\text{th}}(\theta) \, \epsilon_{33}^{\text{th}}(\theta) \, \epsilon_{12}^{\text{th}}(\theta) \, \epsilon_{23}^{\text{th}}(\theta) \, \epsilon_{13}^{\text{th}}(\theta)]^{\text{T}} .
$$

In a finite strain situation, the anisotropic model is formulated in a hypoelastic manner with a constitutive equation of

$$
\dot{\sigma}_{ij} = C_{ijkl} \left(D_{kl} - D_{kl}^{\text{th}} \right) ,
$$

where D_{kl} and D_{kl}^{th} are the total and thermal strain rates, respectively, and the components of the fourth order stiffness tensor C_{ijkl} are related to the contracted notation by

$$
[\mathbb{C}] =
\begin{bmatrix}
C_{1111} & C_{1122} & C_{1133} & C_{1112} & C_{1123} & C_{1113} \\
C_{1122} & C_{2222} & C_{2233} & C_{2212} & C_{2223} & C_{2213} \\
C_{1133} & C_{2233} & C_{3333} & C_{3312} & C_{3323} & C_{3313} \\
C_{1112} & C_{2212} & C_{3312} & C_{1212} & C_{1223} & C_{1213} \\
C_{1123} & C_{2223} & C_{3323} & C_{1223} & C_{2323} & C_{2313} \\
C_{1113} & C_{2213} & C_{3313} & C_{1213} & C_{2313} & C_{1313}
\end{bmatrix} .
$$

17.10. J_2 **Plasticity**

```
BEGIN PARAMETERS FOR MODEL J2_PLASTICITY
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Yield surface parameters
  #
  YIELD STRESS = <real> σ_y
  BETA         = <real> β (1.0)
  #
  # Hardening model
  #
  HARDENING MODEL = LINEAR | POWER_LAW | VOCE | USER_DEFINED |
    FLOW_STRESS | DECOUPLED_FLOW_STRESS | JOHNSON_COOK |
    POWER_LAW_BREAKDOWN
  #
  # Linear hardening
  #
  HARDENING MODULUS = <real> H'
  #
  # Power-law hardening
  #
  HARDENING CONSTANT = <real> A
  HARDENING EXPONENT = <real> n (0.5)
  LUDERS STRAIN      = <real> ε_L (0.0)
  #
  # Voce hardening
  #
  HARDENING MODULUS        = <real> A
  EXPONENTIAL COEFFICIENT = <real> n
  #
  # Johnson-Cook hardening
  #
  HARDENING FUNCTION = <string>hardening_function_name
  RATE CONSTANT      = <real> C
  REFERENCE RATE     = <real> ε̇_0
  #
  # Power law breakdown hardening
  #
```

```
HARDENING FUNCTION = <string>hardening_function_name
RATE COEFFICIENT   = <real> g
RATE EXPONENT      = <real> m
#
# User defined hardening
#
HARDENING FUNCTION = <string>hardening_function_name
#
#
# Following Commands Pertain to Flow_Stress Hardening Model
#
#    -  Isotropic Hardening model
#
ISOTROPIC HARDENING MODEL = LINEAR | POWER_LAW | VOCE |
                            USER_DEFINED
#
# Specifications for Linear, Power-law, and Voce same as above
#
# User defined hardening
#
ISOTROPIC HARDENING FUNCTION = <string>iso_hardening_fun_name
#
#    - Rate dependence
#
RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
   USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
#
# Specifications for Johnson-Cook, Power-law-breakdown
#     same as before EXCEPT no need to specify a
#     hardening function
#
# User defined rate multiplier
#
RATE MULTIPLIER FUNCTION = <string> rate_mult_function_name
#
#    - Temperature dependence
#
TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
   TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
#
#  Johnson-Cook temperature dependence
#
MELTING TEMPERATURE   = <real> $\theta_{melt}$
REFERENCE TEMPERATURE = <real> $\theta_{ref}$
TEMPERATURE EXPONENT  = <real> M
#
#  User-defined temperature dependence
```

```
        TEMPERATURE MULTIPLIER FUNCTION = <string>temp_mult_function_name
#
 #
 # Following Commands Pertain to Decoupled_Flow_Stress Hardening Model
 #
 #    -  Isotropic Hardening model
 #
 ISOTROPIC HARDENING MODEL = LINEAR | POWER_LAW | VOCE | USER_DEFINED
 #
 # Specifications for Linear, Power-law, and Voce same as above
 #
 # User defined hardening
 #
 ISOTROPIC HARDENING FUNCTION = <string>isotropic_hardening_function_name
 #
 #    - Rate dependence
 #
 YIELD RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
             USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
 #
 # Specifications for Johnson-Cook, Power-law-breakdown same as before
 #    EXCEPT no need to specify a hardening function
 #    AND should be preceded by YIELD
 #
 #    As an example for Johnson-Cook yield rate dependence,
 #
 YIELD RATE CONSTANT  = <real> $C^{\mathrm{y}}$
 YIELD REFERENCE RATE = <real> $\dot{\varepsilon}_0^{\mathrm{y}}$
 #
 # User defined rate multiplier
 #
 YIELD RATE MULTIPLIER FUNCTION = <string>yield_rate_mult_function_name
 #
 HARDENING_RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
             USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
 #
 #    Syntax same as for yield parameters but with a HARDENING prefix
 #
 #    - Temperature dependence
 #
 YIELD TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
           TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
 #
 #  Johnson-Cook temperature dependence
 #
 YIELD MELTING TEMPERATURE   = <real> $\theta_{\mathrm{melt}}^{\mathrm{y}}$
 YIELD REFERENCE TEMPERATURE = <real> $\theta_{\mathrm{ref}}^{\mathrm{y}}$
```

```
YIELD TEMPERATURE EXPONENT  = <real> M^y
#
#  User-defined temperature dependence
YIELD TEMPERATURE MULTIPLIER FUNCTION = <string>yield_temp_mult_fun_name
#
HARDENING TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
         TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
#
#    Syntax for hardening constants same as for yield but
#             with HARDENING prefix
#
#
# Optional Failure Definitions
#     Following only need to be defined if intend to use failure model
#
FAILURE MODEL = TEARING_PARAMETER | JOHNSON_COOK_FAILURE | WILKINS
                 | MODULAR_FAILURE | MODULAR_BCJ_FAILURE

CRITICAL FAILURE PARAMETER = <real> d_crit
#
# TEARING_PARAMETER Failure model definitions
#
TEARING PARAMETER EXPONENT  = m
#
# JOHNSON_COOK_FAILURE Failure model definitions
#
JOHNSON COOK D1        = <real> D_1
JOHNSON COOK D2        = <real> D_2
JOHNSON COOK D3        = <real> D_3
JOHNSON COOK D4        = <real> D_4
JOHNSON COOK D5        = <real> D_5
#
#Following Johnson-Cook parameters can only be defined once.  As such, only
#   needed if not previously defined via Johnson-Cook multipliers
#   w/ flow-stress hardening.  Does need to be defined
#   w/ Decoupled Flow Stress
#
REFERENCE RATE        = <real> ε̇_0
REFERENCE TEMPERATURE = <real> T_ref
MELTING TEMPERATURE   = <real> T_melt
#
# WILKINS Failure model definitions
#
WILKINS ALPHA    = <real> α
WILKINS BETA     = <real> β
WILKINS PRESSURE = <real> B
#
```

```
#  MODULAR_FAILURE Failure model definitions
#
PRESSURE MULTIPLIER          = PRESSURE_INDEPENDENT | WILKINS
                               | USER_DEFINED (PRESSURE_INDEPENDENT)
LODE ANGLE MULTIPLIER        = LODE_ANGLE_INDEPENDENT |
                               WILKINS (LODE_ANGLE_INDEPENDENT)
TRIAXIALITY MULTIPLIER       = TRIAXIALITY_INDEPENDENT | JOHNSON_COOK
                               | USER_DEFINED (TRIAXIALITY_INDEPENDENT)
RATE FAIL MULTIPLIER         = RATE_INDEPENDENT | JOHNSON_COOK
                               | USER_DEFINED (RATE_INDEPENDENT)
TEMPERATURE FAIL MULTIPLIER = TEMPERATURE_INDEPENDENT | JOHNSON_COOK
                               | USER_DEFINED (TEMPERATURE_INDEPENDENT)
#
# Individual multiplier definitions
#
PRESSURE MULTIPLIER = WILKINS
WILKINS ALPHA        = <real> α
WILKINS PRESSURE     = <real> B
#
PRESSURE MULTIPLIER = USER_DEFINED
PRESSURE MULTIPLIER FUNCTION = <string> pressure_multiplier_fun_name
#
LODE ANGLE MULTIPLIER = WILKINS
WILKINS BETA          = <real> β
#
TRIAXIALITY MULTIPLIER = JOHNSON_COOK
JOHNSON COOK D1        = <real> D₁
JOHNSON COOK D2        = <real> D₂
JOHNSON COOK D3        = <real> D₃
#
TRIAXIALITY MULTIPLIER = USER_DEFINED
TRIAXIALITY MULTIPLIER FUNCTION = <string> triaxiality_multiplier_fun_name
#
RATE FAIL MULTIPLIER = JOHNSON_COOK
JOHNSON COOK D4       = <real> D₄
# REFERENCE RATE should only be added if not previously defined
REFERENCE RATE        = <real> ε̇₀
#
RATE FAIL MULTIPLIER = USER_DEFINED
RATE FAIL MULTIPLIER FUNCTION = <string> rate_fail_multiplier_fun_name
#
TEMPERATURE FAIL MULTIPLIER = JOHNSON_COOK
JOHNSON COOK D5              = <real> D₅
# JC Temperatures should only be defined if not previously given
REFERENCE TEMPERATURE       = <real> T_ref
MELTING TEMPERATURE         = <real> T_melt
#
```

```
TEMPERATURE FAIL MULTIPLIER          = USER_DEFINED
TEMPERATURE FAIL MULTIPLIER FUNCTION = <string> temp_multiplier_fun_name
#
# MODULAR BCJ_FAILURE Failure model definitions
#
INITIAL DAMAGE    = <real> $\phi_{t=0}$
INITIAL VOID SIZE = <real> $v_{v\ 0}$
DAMAGE BETA       = <real> $\beta$ (0.5)
GROWTH MODEL      = COCKS_ASHBY | NO_GROWTH (NO_GROWTH)
NUCLEATION MODEL  = HORSTEMEYER_GOKHALE | CHU_NEEDLEMAN_STRAIN
                  | NO_NUCLEATION (NO_NUCLEATION)
#
GROWTH RATE FAIL MULTIPLIER         = JOHNSON_COOK | USER_DEFINED
                                    | RATE_INDEPENDENT
                                    (RATE_INDEPENDENT)
GROWTH TEMPERATURE FAIL MULTIPLIER = JOHNSON_COOK | USER_DEFINED
                                    | TEMPERATURE_INDEPENDENT
                                    (TEMPERATURE_INDEPENDENT)
#
NUCLEATION RATE FAIL MULTIPLIER        = JOHNSON_COOK | USER_DEFINED
                                       | RATE_INDEPENDENT
                                       (RATE_INDEPENDENT)
NUCLEATION TEMPERATURE FAIL MULTIPLIER = JOHNSON_COOK | USER_DEFINED
                                       | TEMPERATURE_INDEPENDENT
                                       (TEMPERATURE_INDEPENDENT)
#
# Definitions for individual growth and nucleation models
#
GROWTH MODEL = COCKS_ASHBY
DAMAGE EXPONENT = <real> $m$ (0.5)
#
NUCLEATION MODEL      = HORSTEMEYER_GOKHALE
NUCLEATION PARAMETER1 = <real> $N_1$ (0.0)
NUCLEATION PARAMETER2 = <real> $N_2$ (0.0)
NUCLEATION PARAMETER3 = <real> $N_3$ (0.0)
#
NUCLEATION MODEL        = CHU_NEEDLEMAN_STRAIN
NUCLEATION AMPLITUDE    = <real> $\psi$
MEAN NUCLEATION STRAIN  = <real> $\varepsilon_N$
NUCLEATION STRAIN STD DEV = <real> $s$
#
# Definitions for rate and temperature fail multiplier
#   Note: only showing definitions for growth.
#    Nucleation terms are the same just with NUCLEATION instead
#    of GROWTH
#
GROWTH RATE FAIL MULTIPLIER = JOHNSON_COOK
```

```
GROWTH JOHNSON COOK D4        = <real> D₄ᵍ
GROWTH REFERENCE RATE         = <real> ε̇₀ᵍ
#
GROWTH RATE FAIL MULTIPLIER            = USER_DEFINED
GROWTH RATE FAIL MULTIPLIER FUNCTION = <string> growth_rate_fail_mult_func
#
GROWTH TEMPERATURE FAIL MULTIPLIER = JOHNSON_COOK
GROWTH JOHNSON COOK D5        = <real> D₅ᵍ
GROWTH REFERENCE TEMPERATURE  = <real> T_refᵍ
GROWTH MELTING TEMPERATURE    = <real> T_meltᵍ
#
GROWTH TEMPERATURE FAIL MULTIPLIER          = USER_DEFINED
GROWTH TEMPERATURE FAIL MULTIPLIER FUNCTION = <string> temp_fail_mult_func
#
#
# Optional Adiabatic Heating/Thermal Softening  Definitions
#     Following only need to be defined if intend to use failure model
#
THERMAL SOFTENING MODEL = ADIABATIC | COUPLED
#
SPECIFIC HEAT = <real> c_p #not needed for COUPLED
BETA_TQ       = <real> β^TQ
END [PARAMETERS FOR MODEL J2_PLASTICITY]
```

The J_2 plasticity model is a generic implementation of a von Mises yield surface with kinematic and isotropic hardening features. Unlike other models (*e.g.* Elastic-Plastic, Elastic-Plastic Power Law) more flexible, general hardening forms are implemented enabling different isotropic hardening descriptions and some rate and/or temperature dependence.

As is common to other plasticity models in LAMÉ, the J_2 plasticity model uses a hypoelastic formulation. As such, the total rate of deformation is additively decomposed into an elastic and plastic part such that

$$D_{ij} = D_{ij}^{\mathrm{e}} + D_{ij}^{\mathrm{p}}. \tag{17.21}$$

The objective stress rate, depending only on the elastic deformation, may then be written as,

$$\overset{\circ}{\sigma}_{ij} = \mathbb{C}_{ijkl} D_{kl}^{\mathrm{e}}, \tag{17.22}$$

where $\mathbb{C}_{ijkl}$ is the fourth-order elastic, isotropic stiffness tensor.

The yield surface for the J_2 plasticity model, f, may be written,

$$f\left(\sigma_{ij}, \alpha_{ij}, \bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p, \theta\right) = \phi\left(\sigma_{ij}, \alpha_{ij}\right) - \bar{\sigma}\left(\bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p, \theta\right), \tag{17.23}$$

137

in which α_{ij}, $\bar{\varepsilon}^p$, $\dot{\bar{\varepsilon}}^p$, and θ are the kinematic backstress, equivalent plastic strain, equivalent plastic strain rate, and absolute temperature, respectively, while ϕ and $\bar{\sigma}$ are the effective stress and a generic form of the flow stress. Broadly speaking, the effective stress describes the shape of the yield surface and kinematic effects while the flow stress gives the size of the current yield surface. It should also be noted that in writing the yield surface in this way, the dependence on the state variables is split between the effective stress and flow stress functions.

For J_2 plasticity, the effective stress is given as,

$$\phi^2\left(\sigma_{ij}, \alpha_{ij}\right) = \frac{3}{2}\left(s_{ij} - \alpha_{ij}\right)\left(s_{ij} - \alpha_{ij}\right), \tag{17.24}$$

with s_{ij} being the deviatoric stress defined as $s_{ij} = \sigma_{ij} - (1/3)\sigma_{kk}\delta_{ij}$. For the flow stress, a general representation of the form,

$$\bar{\sigma}\left(\bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p, \theta\right) = \sigma_y \hat{\sigma}_y\left(\dot{\bar{\varepsilon}}^p\right)\breve{\sigma}_y\left(\theta\right) + K\left(\bar{\varepsilon}^p\right)\hat{\sigma}_h\left(\dot{\bar{\varepsilon}}^p\right)\breve{\sigma}_h\left(\theta\right), \tag{17.25}$$

is allowed. In this fashion, the effects of rate ($\hat{\sigma}_{y,h}$) and temperature ($\breve{\sigma}_{y,h}$) dependence on yield (σ_y) and isotropic hardening ($K(\bar{\varepsilon}^p)$) are decomposed. Separate temperature and rate dependencies may be be specified for yield (subscript "y") and hardening ("h"). This assumption is an extension of the multiplicative decomposition of the Johnson-Cook model [10, 11]. It should be noted that not all effects need to be included and the default parameterization of the hardening classes is such that the response is rate and temperature independent. The following section on plastic hardening will go into more detail on possible choices for functional representations.

An associated flow rule is utilized such that the plastic rate of deformation is normal to the yield surface and is given by,

$$D_{ij}^p = \dot{\gamma}\frac{\partial \phi}{\partial \sigma_{ij}} = \dot{\gamma}\frac{3}{2\phi}s_{ij}, \tag{17.26}$$

where $\dot{\gamma}$ is the consistency multiplier enforcing $f = 0$ during plastic deformation. Given the form of f, it can also be shown that $\dot{\gamma} = \dot{\bar{\varepsilon}}^p$.

Additional discussion on options for failure models and adiabatic heating may be found in [12, 13] and [14], respectively.

In the command blocks that define the J_2 plasticity model:

- See the Sierra/SM 5.8 User's Guide Section 5.1.5 for more information on elastic constants input.

- The reference nominal yield stress, $\bar{\sigma}$, is defined with the `YIELD STRESS` command line.

- The beta parameter defines if hardening is isotropic. Consult the Sierra/SM 5.8 User's Guide chapter on Material Models for more information on the beta parameter.

- The type of hardening law is defined with the `HARDENING MODEL` command line, other hardening commands then define the specific shape of that hardening curve.

- The hardening modulus for a linear hardening model is defined with the HARDENING MODULUS command line.

- The hardening constant for a power law hardening model is defined with the HARDENING CONSTANT command line.

- The hardening exponent for a power law hardening model is defined with the HARDENING EXPONENT command line.

- The Lüders strain for a power law hardening model is defined with the LUDERS STRAIN command line.

- The hardening function for a user defined hardening model is defined with the HARDENING FUNCTION command line.

- The shape of the spline for the spline based hardening is defined by the CUBIC SPLINE TYPE, CARDINAL PARAMETER, KNOT EQPS, and KNOT STRESS command lines.

- The isotropic hardening model for the flow stress hardening model is defined with the ISOTROPIC HARDENING MODEL command line.

- The function name of a user-defined isotropic hardening model is defined via the ISOTROPIC HARDENING FUNCTION command line.

- The optional rate multiplier for the flow stress hardening model is defined with the RATE MULTIPLIER command line.

- The optional temperature multiplier for the flow stress hardening model is defined via the TEMPERATURE MULTIPLIER command line.

- The function name of a user-defined temperature multiplier is defined with the TEMPERATURE MULTIPLIER FUNCTION command line.

- For a Johnson-Cook temperature multiplier, the melting temperature, θ_{melt}, is defined via the MELTING TEMPERATURE command line.

- For a Johnson-Cook temperature multiplier, the reference temperature, θ_{ref}, is defined via the REFERENCE TEMPERATURE command line.

- For a Johnson-Cook temperature multiplier, the temperature exponent, M, is defined via the TEMPERATURE EXPONENT command line.

- The optional rate multiplier for the yield stress for the decoupled flow stress hardening model is defined with the YIELD RATE MULTIPLIER command line.

- The optional rate multiplier for the hardening for the decoupled flow stress hardening model is defined with the HARDENING RATE MULTIPLIER command line.

- The optional temperature multiplier for the yield stress for the decoupled flow stress hardening model is defined with the YIELD TEMPERATURE MULTIPLIER command line.

- The optional temperature multiplier for the hardening for the decoupled flow stress hardening model is defined via the HARDENING TEMPERATURE MULTIPLIER command line.

Output variables available for this model are listed in Table 17-4.

Table 17-4. State Variables for J2 PLASTICITY Model (Section 17.10)

Name	Description
EQPS	equivalent plastic strain, $\bar{\varepsilon}^p$
EQDOT	equivalent plastic strain rate, $\dot{\bar{\varepsilon}}^p$
SEFF	effective stress, ϕ
TENSILE_EQPS	tensile equivalent plastic strain, $\bar{\varepsilon}_t^p$
DAMAGE	damage, ϕ
VOID_COUNT	void count, η
VOID_SIZE	void size, υ
DAMAGE_DOT	damage rate, $\dot{\phi}$
VOID_COUNT_DOT	void count rate, $\dot{\eta}$
PLASTIC_WORK_HEAT_RATE	plastic work heat rate, $\dot{Q}^p$

17.11.   Karafillis Boyce Plasticity Model

```
BEGIN PARAMETERS FOR MODEL KARAFILLIS_BOYCE_PLASTICITY
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Yield surface parameters
  #
  YIELD STRESS = <real> σ_y
  A            = <real> a (4.0)
  C            = <real> c (0.0)
  COEFF        = <real> C (2.0/3.0)
  ALPHA 1      = <real>  c_1 (1.0)
  ALPHA 2      = <real>  c_2 (1.0)
  GAMMA 1      = <real>  c_1 (1.5)
  GAMMA 2      = <real>  c_2 (1.5)
  GAMMA 3      = <real>  c_3 (1.5)
  #
  # Hardening model
  #
  HARDENING MODEL = LINEAR | POWER_LAW | USER_DEFINED |
    CUBIC_HERMITE_SPLINE
  #
  # Linear hardening
  #
  HARDENING MODULUS = <real> H'
  #
  # Power law hardening
  #
  HARDENING CONSTANT = <real> A
  HARDENING EXPONENT = <real> n (0.5)
  #
  # User defined hardening
  #
  HARDENING FUNCTION = <string>hardening_function_name
  #
  # Spline based hardening curve
  #
  CUBIC SPLINE TYPE  = <string>
  CARDINAL PARAMETER = <real> val
```

```
    KNOT EQPS             = <real_list> vals
    KNOT STRESS           = <real_list> vals
    #
    # Material coordinates system definition
    #
    COORDINATE SYSTEM                = <string> coordinate_system_name
    DIRECTION FOR ROTATION           = <real> 1|2|3
    ALPHA                            = <real> α₁ (degrees)
    SECOND DIRECTION FOR ROTATION = <real> 1|2|3
    SECOND ALPHA                     = <real> α₂ (degrees)
  END [PARAMETERS FOR MODEL KARAFILLIS_BOYCE_PLASTICITY]
```

The Karafillis and Boyce model [15] is an anisotropic plasticity model. The stress is transformed, based on the anisotropy, and the transformed stress is used in the yield function. The transformed stress, using Voigt notation in the material coordinate system, is given by

$$\mathbf{s}' = \mathbf{C} : \boldsymbol{\sigma}$$

$$[\mathbf{C}] = C \begin{bmatrix} 1 & \beta_1 & \beta_2 & 0 & 0 & 0 \\ \beta_1 & \alpha_1 & \beta_3 & 0 & 0 & 0 \\ \beta_2 & \beta_3 & \alpha_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \gamma_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \gamma_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \gamma_3 \end{bmatrix}$$

where the terms β_k are

$$\beta_1 = \frac{\alpha_2 - \alpha_1 - 1}{2}$$

$$\beta_2 = \frac{\alpha_1 - \alpha_2 - 1}{2}$$

$$\beta_3 = \frac{1 - \alpha_1 - \alpha_2}{2}$$

The response is isotropic if $\alpha_1 = \alpha_2 = 1$, $\gamma_1 = \gamma_2 = \gamma_3 = 1.5$, and $C = 2/3$.

The principal stresses of the transformed stress, $\mathbf{s}'$, are used in the yield function

$$\phi = \{(1-c)\phi_1 + c\phi_2\}^{1/a}$$

$$\phi_1 = \frac{1}{2}\left(|s_1 - s_2|^a + |s_2 - s_3|^a + |s_3 - s_1|^a\right)$$

$$\phi_2 = \frac{3^a}{2^a + 2}\left(|s_1|^a + |s_2|^a + |s_3|^a\right)$$

The exponent, a, is similar to the exponent in the Hosford plasticity model and the constant, c (not to be confused with C above), is a parameter that provides a mixture of two yield functions.

In the command blocks that define the Hosford plasticity model:

- Consult the Sierra/SM 5.8 User's Guide Chapter 5 for more information on elastic constants input.
- The reference nominal yield stress, $\bar{\sigma}$, is defined with the `YIELD STRESS` command line.
- The exponent for the yield surface description, a, is defined with the `A` command line.
- The coefficient C in the stress transformation is defined with the `COEFF` command line.
- The term α_1 in the stress transformation is defined with the `ALPHA 1` command line.
- The term α_2 in the stress transformation is defined with the `ALPHA 2` command line.
- The term γ_1 in the stress transformation is defined with the `GAMMA 1` command line.
- The term γ_2 in the stress transformation is defined with the `GAMMA 2` command line.
- The term γ_3 in the stress transformation is defined with the `GAMMA 3` command line.
- The type of hardening law is defined with the `HARDENING MODEL` command line, other hardening commands then define the specific shape of that hardening curve.
- The hardening modulus for a linear hardening model is defined with the `HARDENING MODULUS` command line.
- The hardening constant for a power law hardening model is defined with the `HARDENING CONSTANT` command line.
- The hardening exponent for a power law hardening model is defined with the `HARDENING EXPONENT` command line.
- The hardening function for a user defined hardening model is defined with the `HARDENING FUNCTION` command line.
- The shape of the spline for the spline based hardening is defined by the `CUBIC SPLINE TYPE`, `CARDINAL PARAMETER`, `KNOT EQPS`, and `KNOT STRESS` command lines.

Output variables available for this model are listed in Table 17-5.

Table 17-5. State Variables for KARAFILLIS_BOYCE_PLASTICITY Model

Index	Name	Variable Description
1	EQPS	equivalent plastic strain, $\bar{\varepsilon}^{p}$

17.12.   Cazacu Plasticity Model

```
BEGIN PARAMETERS FOR MODEL CAZACU_PLASTICITY
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Yield surface parameters
  #
  YIELD STRESS = <real> σ_y
  A            = <real> a (1.0)
  #
  # tension/compression asymmetry
  #
  RATIO        = <real> r
  #
  # Hardening model
  #
  HARDENING MODEL = LINEAR | POWER_LAW | VOCE | USER_DEFINED |
    FLOW_STRESS | DECOUPLED_FLOW_STRESS | JOHNSON_COOK |
    POWER_LAW_BREAKDOWN
  #
  # Linear hardening
  #
  HARDENING MODULUS = <real> H′
  #
  # Power-law hardening
  #
  HARDENING CONSTANT = <real> A
  HARDENING EXPONENT = <real> n (0.5)
  LUDERS STRAIN      = <real> ε_L  (0.0)
  #
  # Voce hardening
  #
  HARDENING MODULUS        = <real> A
  EXPONENTIAL COEFFICIENT = <real> n
  #
  # Johnson-Cook hardening
  #
  HARDENING FUNCTION = <string>hardening_function_name
  RATE CONSTANT      = <real> C
```

```
REFERENCE RATE       = <real> $\dot{\varepsilon}_0$
#
# Power law breakdown hardening
#
HARDENING FUNCTION = <string>hardening_function_name
RATE COEFFICIENT    = <real> $g$
RATE EXPONENT       = <real> $m$
#
# User defined hardening
#
HARDENING FUNCTION = <string>hardening_function_name
#
#
# Following Commands Pertain to Flow_Stress Hardening Model
#
#    -  Isotropic Hardening model
#
ISOTROPIC HARDENING MODEL = LINEAR | POWER_LAW | VOCE |
                            USER_DEFINED
#
# Specifications for Linear, Power-law, and Voce same as above
#
# User defined hardening
#
ISOTROPIC HARDENING FUNCTION = <string>iso_hardening_fun_name
#
#    - Rate dependence
#
RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
   USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
#
# Specifications for Johnson-Cook, Power-law-breakdown
#    same as before EXCEPT no need to specify a
#    hardening function
#
# User defined rate multiplier
#
RATE MULTIPLIER FUNCTION = <string> rate_mult_function_name
#
#    - Temperature dependence
#
TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
   TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
#
#  Johnson-Cook temperature dependence
#
MELTING TEMPERATURE   = <real> $\theta_{melt}$
```

146

```
REFERENCE TEMPERATURE = <real> $\theta_{ref}$
TEMPERATURE EXPONENT  = <real> $M$
 #
 #  User-defined temperature dependence
 TEMPERATURE MULTIPLIER FUNCTION = <string>temp_mult_function_name
#
 #
 # Following Commands Pertain to Decoupled_Flow_Stress Hardening Model
 #
 #    -  Isotropic Hardening model
 #
 ISOTROPIC HARDENING MODEL = LINEAR | POWER_LAW | VOCE | USER_DEFINED
 #
 # Specifications for Linear, Power-law, and Voce same as above
 #
 # User defined hardening
 #
 ISOTROPIC HARDENING FUNCTION = <string>isotropic_hardening_function_name
 #
 #    - Rate dependence
 #
 YIELD RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
              USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
 #
 # Specifications for Johnson-Cook, Power-law-breakdown same as before
 #     EXCEPT no need to specify a hardening function
 #     AND should be preceded by YIELD
 #
 #     As an example for Johnson-Cook yield rate dependence,
 #
 YIELD RATE CONSTANT  = <real> $C^y$
 YIELD REFERENCE RATE = <real> $\dot{\varepsilon}_0^y$
 #
 # User defined rate multiplier
 #
 YIELD RATE MULTIPLIER FUNCTION = <string>yield_rate_mult_function_name
 #
 HARDENING_RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
              USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
 #
 #     Syntax same as for yield parameters but with a HARDENING prefix
 #
 #     - Temperature dependence
 #
 YIELD TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
           TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
 #
```

```
   #   Johnson-Cook temperature dependence
   #
   YIELD MELTING TEMPERATURE    = <real> θ_melt^Y
   YIELD REFERENCE TEMPERATURE = <real> θ_ref^Y
   YIELD TEMPERATURE EXPONENT   = <real> M^Y
   #
   #   User-defined temperature dependence
   YIELD TEMPERATURE MULTIPLIER FUNCTION = <string>yield_temp_mult_fun_name
   #
   HARDENING TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
            TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
   #
   #    Syntax for hardening constants same as for yield but
   #            with HARDENING prefix
  #
  END  [PARAMETERS FOR MODEL CAZACU_PLASTICITY]
```

The Cazacu plasticity model is an isotropic plasticity model that accounts for tension/compression asymmetry in yield.. This model is used primarily for modeling the plastic deformation of HCP metals which can show significant tension/compression asymmetry. As is common to other plasticity models in Lamé, the Cazacu plasticity model uses a hypoelastic formulation. As such, the total rate of deformation is additively decomposed into an elastic and plastic part such that

$$D_{ij} = D_{ij}^e + D_{ij}^p. \tag{17.27}$$

The objective stress rate, depending only on the elastic deformation, may then be written as,

$$\overset{\circ}{\sigma}_{ij} = \mathbb{C}_{ijkl} D_{kl}^e, \tag{17.28}$$

where $\mathbb{C}_{ijkl}$ is the fourth-order elastic, isotropic stiffness tensor.

The yield surface for the Cazacu plasticity model, f, may be written,

$$f\left(\sigma_{ij}, \bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p\right) = \phi\left(\sigma_{ij}\right) - \bar{\sigma}\left(\bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p\right), \tag{17.29}$$

in which $\bar{\varepsilon}^p$ and $\dot{\bar{\varepsilon}}^p$ are the equivalent plastic strain and equivalent plastic strain rate respectively, while ϕ and $\bar{\sigma}$ are generically the effective stress and flow stress. Broadly speaking, the flow stress gives the size of the current yield surface. For the Cazacu plasticity model, the effective stress is given as,

$$\phi\left(\sigma_{ij}\right) = \left\{ h(a,k)\left[(|s_1| - ks_1)^a + (|s_2| - ks_2)^a + (|s_3| - ks_3)^a \right] \right\}^{1/a} \tag{17.30}$$

148

with s_i being the principal deviatoric stresses, k and a are model parameters describing asymmetry in tension/compression along with the shape of the yield surface. The function $h(a, k)$ is a normalizing factor that is set so that the effective stress for a uniaxial stress state is σ

$$h = \frac{3^a}{((2(1-k))^a + 2(1+k)^a)} \tag{17.31}$$

The parameter k is calculated from the tension/compression asymmetry. The ratio of the yield stress in tension to the yield stress in compression is

$$r = \sigma_y^T / \sigma_y^C \tag{17.32}$$

The value of k is

$$k = \frac{1 - h(r)}{1 + h(r)} \quad ; \quad h(r) = \left[\frac{2^a - 2r^a}{(2r)^a - 2} \right]^{1/a} \tag{17.33}$$

An associated flow rule is utilized such that the plastic rate of deformation is normal to the yield surface and is given by,

$$\dot{D}_{ij}^p = \dot{\gamma} \frac{\partial \phi}{\partial \sigma_{ij}}, \tag{17.34}$$

where $\dot{\gamma}$ is the consistency multiplier enforcing $f = 0$ during plastic deformation. Given the form of f, it can also be shown that $\dot{\gamma} = \dot{\bar{\varepsilon}}^p$, i.e. the consistency parameter is equal to the rate of the equivalent plastic strain.

In the command blocks that define the Cazacu plasticity model:

- See the Sierra/SM 5.8 User's Guide Section 5.1.5 for more information on elastic constants input.

- The reference nominal yield stress, $\bar{\sigma}$, is defined with the YIELD STRESS command line.

- The type of hardening law is defined with the HARDENING MODEL command line, other hardening commands then define the specific shape of that hardening curve.

- The hardening modulus for a linear hardening model is defined with the HARDENING MODULUS command line.

- The hardening constant for a power law hardening model is defined with the HARDENING CONSTANT command line.

- The hardening exponent for a power law hardening model is defined with the HARDENING EXPONENT command line.

- The Lüders strain for a power law hardening model is defined with the LUDERS STRAIN command line.

- The hardening function for a user defined hardening model is defined with the HARDENING FUNCTION command line.

149

- The shape of the spline for the spline based hardening is defined by the CUBIC SPLINE TYPE, CARDINAL PARAMETER, KNOT EQPS, and KNOT STRESS command lines.

- The isotropic hardening model for the flow stress hardening model is defined with the ISOTROPIC HARDENING MODEL command line.

- The function name of a user-defined isotropic hardening model is defined via the ISOTROPIC HARDENING FUNCTION command line.

- The optional rate multiplier for the flow stress hardening model is defined with the RATE MULTIPLIER command line.

- The optional temperature multiplier for the flow stress hardening model is defined via the TEMPERATURE MULTIPLIER command line.

- The function name of a user-defined temperature multiplier is defined with the TEMPERATURE MULTIPLIER FUNCTION command line.

- For a Johnson-Cook temperature multiplier, the melting temperature, θ_{melt}, is defined via the MELTING TEMPERATURE command line.

- For a Johnson-Cook temperature multiplier, the reference temperature, θ_{ref}, is defined via the REFERENCE TEMPERATURE command line.

- For a Johnson-Cook temperature multiplier, the temperature exponent, M, is defined via the TEMPERATURE EXPONENT command line.

- The optional rate multiplier for the yield stress for the decoupled flow stress hardening model is defined with the YIELD RATE MULTIPLIER command line.

- The optional rate multiplier for the hardening for the decoupled flow stress hardening model is defined with the HARDENING RATE MULTIPLIER command line.

- The optional temperature multiplier for the yield stress for the decoupled flow stress hardening model is defined with the YIELD TEMPERATURE MULTIPLIER command line.

- The optional temperature multiplier for the hardening for the decoupled flow stress hardening model is defined via the HARDENING TEMPERATURE MULTIPLIER command line.

Output variables available for this model are listed in Table 17-6.

Table 17-6. State Variables for CAZACU PLASTICITY Model (Section 17.12)

Name	Description
EQPS	equivalent plastic strain, $\bar{\varepsilon}^p$
EQDOT	equivalent plastic strain rate, $\dot{\bar{\varepsilon}}^p$
SEFF	effective stress, ϕ

17.13.   Cazacu Orthotropic Plasticity Model

```
BEGIN PARAMETERS FOR MODEL CAZACU_ORTHOTROPIC_PLASTICITY
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Yield surface parameters
  #
  YIELD STRESS = <real> σ_y
  A            = <real> a (4.0)
  #
  # tension/compression asymmetry
  #
  KP           = <real> k'
  KPP          = <real> k''
  #
  # orthotroppic parameters
  #
  CP11         = <real> c'_{11}
  CP22         = <real> c'_{22}
  CP33         = <real> c'_{33}
  CP12         = <real> c'_{12}
  CP23         = <real> c'_{23}
  CP31         = <real> c'_{31}
  CP44         = <real> c'_{44}
  CP55         = <real> c'_{55}
  CP66         = <real> c'_{66}
  CPP11        = <real> c''_{11}
  CPP22        = <real> c''_{22}
  CPP33        = <real> c''_{33}
  CPP12        = <real> c''_{12}
  CPP23        = <real> c''_{23}
  CPP31        = <real> c''_{31}
  CPP44        = <real> c''_{44}
  CPP55        = <real> c''_{55}
  CPP66        = <real> c''_{66}
  #
  # Hardening model
  #
  HARDENING MODEL = LINEAR | POWER_LAW | VOCE | USER_DEFINED |
```

151

```
  FLOW_STRESS | DECOUPLED_FLOW_STRESS | JOHNSON_COOK |
  POWER_LAW_BREAKDOWN
#
# Linear hardening
#
HARDENING MODULUS = <real> H'
#
# Power-law hardening
#
HARDENING CONSTANT = <real> A
HARDENING EXPONENT = <real> n (0.5)
LUDERS STRAIN      = <real> εL (0.0)
#
# Voce hardening
#
HARDENING MODULUS        = <real> A
EXPONENTIAL COEFFICIENT = <real> n
#
# Johnson-Cook hardening
#
HARDENING FUNCTION = <string>hardening_function_name
RATE CONSTANT      = <real> C
REFERENCE RATE     = <real> ε̇0
#
# Power law breakdown hardening
#
HARDENING FUNCTION = <string>hardening_function_name
RATE COEFFICIENT   = <real> g
RATE EXPONENT      = <real> m
#
# User defined hardening
#
HARDENING FUNCTION = <string>hardening_function_name
#
#
# Following Commands Pertain to Flow_Stress Hardening Model
#
#    -  Isotropic Hardening model
#
ISOTROPIC HARDENING MODEL = LINEAR | POWER_LAW | VOCE |
                            USER_DEFINED
#
# Specifications for Linear, Power-law, and Voce same as above
#
# User defined hardening
#
ISOTROPIC HARDENING FUNCTION = <string>iso_hardening_fun_name
```

```
#
#      - Rate dependence
#
RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
    USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
#
# Specifications for Johnson-Cook, Power-law-breakdown
#      same as before EXCEPT no need to specify a
#      hardening function
#
# User defined rate multiplier
#
RATE MULTIPLIER FUNCTION = <string> rate_mult_function_name
#
#      - Temperature dependence
#
TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
    TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
#
#   Johnson-Cook temperature dependence
#
MELTING TEMPERATURE   = <real> $\theta_{melt}$
REFERENCE TEMPERATURE = <real> $\theta_{ref}$
TEMPERATURE EXPONENT  = <real> $M$
#
#   User-defined temperature dependence
TEMPERATURE MULTIPLIER FUNCTION = <string>temp_mult_function_name
#
 #
# Following Commands Pertain to Decoupled_Flow_Stress Hardening Model
#
#    -  Isotropic Hardening model
#
ISOTROPIC HARDENING MODEL = LINEAR | POWER_LAW | VOCE | USER_DEFINED
#
# Specifications for Linear, Power-law, and Voce same as above
#
# User defined hardening
#
ISOTROPIC HARDENING FUNCTION = <string>isotropic_hardening_function_name
#
#      - Rate dependence
#
YIELD RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
            USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
#
# Specifications for Johnson-Cook, Power-law-breakdown same as before
```

```
#       EXCEPT no need to specify a hardening function
#       AND should be preceded by YIELD
#
#       As an example for Johnson-Cook yield rate dependence,
#
YIELD RATE CONSTANT  = <real> C^y
YIELD REFERENCE RATE = <real> ε̇_0^y
#
# User defined rate multiplier
#
YIELD RATE MULTIPLIER FUNCTION = <string>yield_rate_mult_function_name
#
HARDENING_RATE MULTIPLIER = JOHNSON_COOK | POWER_LAW_BREAKDOWN |
            USER_DEFINED | RATE_INDEPENDENT (RATE_INDEPENDENT)
#
#       Syntax same as for yield parameters but with a HARDENING prefix
#
#       - Temperature dependence
#
YIELD TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
          TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
#
#  Johnson-Cook temperature dependence
#
YIELD MELTING TEMPERATURE   = <real> θ_melt^y
YIELD REFERENCE TEMPERATURE = <real> θ_ref^y
YIELD TEMPERATURE EXPONENT  = <real> M^y
#
#  User-defined temperature dependence
YIELD TEMPERATURE MULTIPLIER FUNCTION = <string>yield_temp_mult_fun_name
#
HARDENING TEMPERATURE MULTIPLIER = JOHNSON_COOK | USER_DEFINED |
          TEMPERATURE_INDEPENDENT (TEMPERATURE_INDEPENDENT)
#
#   Syntax for hardening constants same as for yield but
#            with HARDENING prefix
#
END [PARAMETERS FOR MODEL CAZACU_ORTHOTROPIC_PLASTICITY]
```

The orthotropic Cazacu plasticity model is an extension of the tension/compression asymmetry model of Cazacu to account for orthotropic response. As is common to other plasticity models in Lamé, the Cazacu plasticity model uses a hypoelastic formulation. As such, the total rate of deformation is additively decomposed into an elastic and plastic part such that

$$D_{ij} = D_{ij}^{e} + D_{ij}^{p}. \tag{17.35}$$

154

The objective stress rate, depending only on the elastic deformation, may then be written as,

$$\overset{\circ}{\sigma}_{ij} = \mathbb{C}_{ijkl} D^e_{kl}, \tag{17.36}$$

where $\mathbb{C}_{ijkl}$ is the fourth-order elastic, isotropic stiffness tensor.

The yield surface for the orthotropic Cazacu plasticity model, f, may be written,

$$f\left(\sigma_{ij}, \bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p\right) = \phi\left(\sigma_{ij}\right) - \bar{\sigma}\left(\bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p\right), \tag{17.37}$$

in which $\bar{\varepsilon}^p$ and $\dot{\bar{\varepsilon}}^p$ are the equivalent plastic strain and equivalent plastic strain rate respectively, while ϕ and $\bar{\sigma}$ are generically the effective stress and flow stress. Broadly speaking, the flow stress gives the size of the current yield surface. For the orthotropic Cazacu plasticity model, the effective stress is given as,

$$\phi^2\left(\sigma_{ij}\right) = \left\{ h(a, k', k'', c'_{ij}, c''_{ij}) \left[\left(|s'_1| - k' s'_1\right)^a + \left(|s'_2| - k' s'_2\right)^a + \left(|s'_3| - k' s'_3\right)^a \right.\right.$$

$$\tag{17.38}$$

$$\left.\left. + \left(|s''_1| - k'' s''_1\right)^a + \left(|s''_2| - k'' s''_2\right)^a + \left(|s''_3| - k'' s''_3\right)^a \right] \right\}^{1/a}$$

with s'_i and s''_i being the principal transformed stresses, and k', k'' and a are model parameters describing the asymmetry in tension/compression along with the general shape of the yield surface. The transformed stresses, which account for the anisotropy, are given by

$$s'_{ij} = C'_{ijkl} s_{kl} \quad ; \quad s''_{ij} = C''_{ijkl} s_{kl} \tag{17.39}$$

In matrix notation in the material coordinate system, these transformations are

$$\begin{Bmatrix} s'_{11} \\ s'_{22} \\ s'_{33} \\ s'_{12} \\ s'_{23} \\ s'_{31} \end{Bmatrix} = \begin{bmatrix} c'_{11} & c'_{12} & c'_{31} & 0 & 0 & 0 \\ c'_{12} & c'_{22} & c'_{23} & 0 & 0 & 0 \\ c'_{31} & c'_{23} & c'_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & c'_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c'_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & c'_{66} \end{bmatrix} \begin{Bmatrix} s_{11} \\ s_{22} \\ s_{33} \\ s_{12} \\ s_{23} \\ s_{31} \end{Bmatrix} \tag{17.40}$$

and

$$\begin{Bmatrix} s''_{11} \\ s''_{22} \\ s''_{33} \\ s''_{12} \\ s''_{23} \\ s''_{31} \end{Bmatrix} = \begin{bmatrix} c''_{11} & c''_{12} & c''_{31} & 0 & 0 & 0 \\ c''_{12} & c''_{22} & c''_{23} & 0 & 0 & 0 \\ c''_{31} & c''_{23} & c''_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & c''_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & c''_{55} & 0 \\ 0 & 0 & 0 & 0 & 0 & c''_{66} \end{bmatrix} \begin{Bmatrix} s_{11} \\ s_{22} \\ s_{33} \\ s_{12} \\ s_{23} \\ s_{31} \end{Bmatrix} \tag{17.41}$$

The c'_{ij} and c''_{ij} are model parameters governing the anisotropy. The normalizing coefficient, h, depends on the model parameters and is computed so that the effective stress for a uniaxial stress state is σ.

An associated flow rule is utilized such that the plastic rate of deformation is normal to the yield surface and is given by,

$$\dot{D}^{\mathrm{p}}_{ij} = \dot{\gamma}\frac{\partial \phi}{\partial \sigma_{ij}}, \tag{17.42}$$

where $\dot{\gamma}$ is the consistency multiplier enforcing $f = 0$ during plastic deformation. Given the form of f, it can also be shown that $\dot{\gamma} = \dot{\bar{\varepsilon}}^p$, i.e. the consistency parameter is equal to the rate of the equivalent plastic strain.

In the command blocks that define the orthotropic Cazacu plasticity model:

- See the Sierra/SM 5.8 User's Guide Section 5.1.5 for more information on elastic constants input.

- The reference nominal yield stress, $\bar{\sigma}$, is defined with the `YIELD STRESS` command line.

- The type of hardening law is defined with the `HARDENING MODEL` command line, other hardening commands then define the specific shape of that hardening curve.

- The hardening modulus for a linear hardening model is defined with the `HARDENING MODULUS` command line.

- The hardening constant for a power law hardening model is defined with the `HARDENING CONSTANT` command line.

- The hardening exponent for a power law hardening model is defined with the `HARDENING EXPONENT` command line.

- The Lüders strain for a power law hardening model is defined with the `LUDERS STRAIN` command line.

- The hardening function for a user defined hardening model is defined with the `HARDENING FUNCTION` command line.

- The shape of the spline for the spline based hardening is defined by the `CUBIC SPLINE TYPE`, `CARDINAL PARAMETER`, `KNOT EQPS`, and `KNOT STRESS` command lines.

- The isotropic hardening model for the flow stress hardening model is defined with the `ISOTROPIC HARDENING MODEL` command line.

- The function name of a user-defined isotropic hardening model is defined via the `ISOTROPIC HARDENING FUNCTION` command line.

- The optional rate multiplier for the flow stress hardening model is defined with the `RATE MULTIPLIER` command line.

- The optional temperature multiplier for the flow stress hardening model is defined via the `TEMPERATURE MULTIPLIER` command line.

- The function name of a user-defined temperature multiplier is defined with the TEMPERATURE MULTIPLIER FUNCTION command line.

- For a Johnson-Cook temperature multiplier, the melting temperature, θ_{melt}, is defined via the MELTING TEMPERATURE command line.

- For a Johnson-Cook temperature multiplier, the reference temperature, θ_{ref}, is defined via the REFERENCE TEMPERATURE command line.

- For a Johnson-Cook temperature multiplier, the temperature exponent, M, is defined via the TEMPERATURE EXPONENT command line.

- The optional rate multiplier for the yield stress for the decoupled flow stress hardening model is defined with the YIELD RATE MULTIPLIER command line.

- The optional rate multiplier for the hardening for the decoupled flow stress hardening model is defined with the HARDENING RATE MULTIPLIER command line.

- The optional temperature multiplier for the yield stress for the decoupled flow stress hardening model is defined with the YIELD TEMPERATURE MULTIPLIER command line.

- The optional temperature multiplier for the hardening for the decoupled flow stress hardening model is defined via the HARDENING TEMPERATURE MULTIPLIER command line.

Output variables available for this model are listed in Table 17-7.

Table 17-7. State Variables for CAZACU ORTHOTROPIC PLASTICITY Model (Section 17.13)

Name	Description
EQPS	equivalent plastic strain, $\bar{\varepsilon}^p$
EQDOT	equivalent plastic strain rate, $\dot{\bar{\varepsilon}}^p$
SEFF	effective stress, ϕ

17.14. Skorohod-Olevsky Viscous Sintering (SOVS)

```
BEGIN PARAMETERS FOR MODEL SOVS
  #
  #  Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  #  Initial relative density
  #
  RHO0      = <real> ρ_0
  #
  #  Normalized shear viscosity relative density dependence
  #
  A1        = <real> a_1
  B1        = <real> b_1
  #
  #  Normalized bulk viscosity relative density dependence
  #
  A2        = <real> a_2
  B2        = <real> b_2
  C2        = <real> c_2
  #
  #  Effective sinter stress relative density dependence
  #
  SIGMA_S0 = <real> σ_{s0}
  A3        = <real> a_3
  B3        = <real> b_3
  #
  #  Skeleton shear viscosity temperature dependence
  #
  A4        = <real> a_4
  B4        = <real> b_4
  C4        = <real> c_4
  T0        = <real> θ_0
  #
  #  Numerical integration parameters
  #
  BETA      = <real> β (0.5)
  TOL       = <real> tol (1.0E6)
END [PARAMETERS FOR MODEL SOVS]
```

158

The Skorohod-Olevsky viscous sintering (SOVS) model is a continuum scale model for the evolution of porosity and deformation of ceramic materials through sintering. This implementation is intended to capture geometric evolution of a body through such a process thereby enabling the design and manufacture of complex 3D components and/or structures.

Here, the model is a linear-viscous form of the non-linear viscous incompressible model of Olevsky [16] based on the concepts of plastic porous bodies. The specific implementation used here is that of Argüello and coworkers [17]. Like other inelastic models, an additive split in the rate of deformation,

$$D_{ij} = D_{ij}^{\text{e}} + D_{ij}^{\text{in}}, \tag{17.43}$$

is used in which the elastic constitutive relation may be written as,

$$\overset{\circ}{\sigma}_{ij} = \mathbb{C}_{ijkl} D_{kl}^{\text{e}}. \tag{17.44}$$

For the inelastic response, a constitutive relation may be derived via thermodynamic analysis and dissipation considerations (see [16]) producing an inelastic (viscous) strain rate, D_{ij}^{in}, of the form,

$$D_{ij}^{\text{in}} = \frac{\sigma'_{ij}}{2\eta_0(\theta)\phi(\rho)} + \frac{\sigma_{kk}/3 - \sigma_s(\rho)}{6\eta_0(\theta)\psi(\rho)}\delta_{ij}, \tag{17.45}$$

with η_0, ϕ, σ_s, and ψ, being the shear viscosity of the fully dense skeleton, normalized shear viscosity, sintering stress, and normalized bulk viscosity. A split between the contributions of the deviatoric, σ'_{ij}, and volumetric, $\sigma_{kk}/3$, components of the Cauchy stress, $\sigma_{ij} = \sigma'_{ij} + (1/3)\sigma_{kk}\delta_{ij}$, is also utilized. Note, in (17.45) explicit dependencies of the various terms on relative density, ρ, and absolute temperature, θ, are indicated although for simplicity will be neglected in the remainder. Importantly, the relative density, ρ, is defined as,

$$\rho = \frac{\rho_t}{\rho_T}, \tag{17.46}$$

where ρ_T is the theoretical density and ρ_t is the current material density at time t. Conservation of mass may be invoked to show the time rate of change of the relative density, $\dot{\rho}$, is simply,

$$\dot{\rho} = -\rho D_{kk}^{\text{in}}. \tag{17.47}$$

Based on previously performed kinetic Monte-Carlo analysis, Argüello and coworkers [17] used the following density dependent relations,

$$\phi(\rho) = a_1 \rho^{b_1}, \tag{17.48}$$

$$\psi(\rho) = a_2 \frac{\rho^{b_2}}{(1-\rho)^{c_2}}, \tag{17.49}$$

$$\sigma_s(\rho) = \sigma_{s0} \bar{\sigma}_s(\rho), \tag{17.50}$$

$$\bar{\sigma}_s(\rho) = a_3 \rho^{b_3}. \tag{17.51}$$

In (17.50) it can be observed that the sintering stress is decomposed into two parts: (*i*) the local sintering stress, σ_{s0}, and (*ii*) the relative density dependent normalized effective sintering stress, $\bar{\sigma}_s$. With respect to the latter, the functional representation is given in (17.51). For the former, the value may be approximated as,

$$\sigma_{s0} = \frac{3\alpha}{r_0}, \tag{17.52}$$

with α being the surface tension and r_0 the average grain size. In the current implementation, only the local sintering stress is input to the model as the surface tension and grain size are unneeded elsewhere.

The temperature dependence of the shear viscosity of the fully dense skeleton was proposed in a previous effort by Olevsky *et al.* [18] and is given as,

$$\eta_0(\theta) = a_4 \left(\frac{\theta}{\theta_0} \right)^2 + b_4 \frac{\theta}{\theta_0} + c_4, \tag{17.53}$$

in which θ_0 is a reference temperature. During the efforts of Argüello *et al.* [17], alternative forms of this dependence based on either an Arrhenius relationship or the introduction of an additional variable for grain growth were put forth. Notably, the latter showed good agreement with experimental measurements although at the time only a 1D form was considered as alterations to the numerical scheme were needed for the 3D implementation. For the current model, the previous quadratic form is used as that was the expression validated against experiments.

For details on the SOVS model, please see [17].

In the command blocks that define the SOVS model:

- See the Sierra/SM 5.8 User's Guide Section 5.1.5 for more information on elastic constants input.

- The initial relative density, ρ_0, $(0 \leq \rho_0 \leq 1)$ is defined with the `RHO0` command line.

- The modulus of the relative density dependence of the normalized shear viscosity, a_1, is defined with the `A1` command line.

- The exponent of the relative density dependence of the normalized shear viscosity, b_1, is defined with the `B1` command line.

- The modulus of the relative density dependence of the normalized bulk viscosity, a_2, is defined with the A2 command line.

- The exponent of the relative density dependence of the normalized bulk viscosity, b_2, is defined with the B2 command line.

- The exponent of the relative porosity dependence of the normalized bulk viscosity, c_2, is defined with the C2 command line.

- The local sintering stress, σ_{s0}, is defined with the SIGMA_S0 command line.

- The modulus of the relative density dependence of the normalized effective sintering stress, a_3, is defined with the A3 command line.

- The exponent of the relative density dependence of the normalized effective sintering stress, b_3, is defined with the B3 command line.

- The quadratic constant of the skeleton shear viscosity on normalized temperature, a_4, is defined with the A4 command line.

- The linear constant of the skeleton shear viscosity on normalized temperature, b_4, is defined with the B4 command line.

- The constant of the skeleton shear viscosity on normalized temperature, c_4, is defined with the C4 command line.

- The reference temperature used with the skeleton shear viscosity, θ_0, is defined with the T0 command line.

- The type of integration is controlled by the integration selection parameter, β ($0 \leq \beta \leq 1$), is defined with the BETA command line. A value of one corresponds to implicit constitutive integration while zero is specified for explicit integration. The default value is 0.5.

- The tolerance of convergence for the non-linear problem associated with constitutive model integration, tol, may be specified by the TOL command line. A default value of 1.0×10^6 is used if no value is specified.

- It is recommended that the default values of constitutive integration parameters (β, tol) be used. Alternative selections remain an unverified development option.

Output variables available for this model are listed in Table 17-8.

Table 17-8. State Variables for SOVS Model (Section 17.14)

Name	Description
RHO	relative density, ρ
RHO_DOT	time rate of change of density, $\dot{\rho}$
EPSILON_IN_DOT	inelastic strain rate, $\dot{\varepsilon}_{ij}^{\text{in}}$
ITERATIONS	constitutive integration convergence iterations, iter

17.15. Hydra Plasticity

17.15.1. Summary

The hydra plasticity model has the following features:

- Isotropic elastic behavior, with elastic constants that may optionally depend on temperature,

- A Hill yield surface that may undergo isotropic hardening,

- Tabular definition of the material's hardening behavior, with dependence on equivalent plastic strain and optional dependence on temperature, and/or equivalent plastic strain rate,

- Option inclusion of material failure,

- Tabular definition of the material's failure strain, with dependence on stress triaxiality and optional dependence on temperature, equivalent plastic strain rate, and/or Lode angle (via a Lode parameter),

- Linear material strength and stiffness degradation following failure initiation that is based on the fracture energy for the material and is normalized to the element characteristic length to reduce mesh dependencies,

- User specified value of decay at which an integration point is flagged for removal from the analysis, and

- Optional inclusion of heating caused by plastic deformation of the material.

17.15.2. User Guide

```
BEGIN PARAMETERS FOR MODEL HYDRA_PLASTICITY
  #
  # Elastic Constants
  #
  YOUNGS MODULUS      = <real> E
  YOUNGS FUNCTION     = <string> youngs_func_name f_E(T_t)
  POISSONS RATIO      = <real> ν
  POISSONS FUNCTION   = <string> poissons_func_name f_ν(T_t)
  #
  # Material Coordinates System Definition
  #
  COORDINATE SYSTEM                = <string> coordinate_system_name
  DIRECTION FOR ROTATION           = <real> 1|2|3
  ALPHA                            = <real> α_1 (degrees)
  SECOND DIRECTION FOR ROTATION = <real> 1|2|3
  SECOND ALPHA                     = <real> α_2 (degrees)
  #
  # Yield Surface Parameters
  #
  R11              = <real> R11
```

```
   R22                = <real> R22
   R33                = <real> R33
   R12                = <real> R12
   R23                = <real> R23
   R31                = <real> R31
   #
   # Hardening Parameters
   #
   HARDENING FUNCTION = <string> hardening_func_name $f_H\left(\bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p, T_t\right)$
   PLASTIC STRAIN RATE LOG FLAG = <bool> true or false
   #
   # Plastic Heating Parameters
   #
   SPECIFIC HEAT           = <real> $C$
   SPECIFIC HEAT FUNCTION  = <string> specific_heat_func_name $f_C(T_t)$
   INELASTIC HEAT FRACTION = <real> $\eta$
   ADIABATIC ANALYSIS FLAG = <bool> true or false
   #
   # Failure Parameters
   #
   FAILURE FUNCTION                = <string> failure_func_name $f_F\left(\varkappa, \dot{\bar{\varepsilon}}^p, \Theta_p, T_t\right)$
   FRACTURE ENERGY                 = <real> $G_f$
   ELEMENT REMOVAL DECAY VALUE = <real> $\varphi_r$
   FAILURE ANALYSIS FLAG       = <bool> true or false
END [PARAMETERS FOR MODEL HYDRA_PLASTICITY]
```

- The elastic constants, E and ν, are defined by the YOUNGS MODULUS and POISSONS RATIO command lines, respectively. Both of these values must be defined by the user, there are no defaults.

- The temperature scaling factor functions for E and ν are defined by the YOUNGS MODULUS FUNCTION and POISSONS RATIO FUNCTION command lines, respectively. The youngs_func_name and poissons_func_name reference functions defined by separate FUNCTION command blocks. These functions should define temperature dependent scaling factors that when applied to E and ν (defined by the YOUNGS MODULUS and POISSONS RATIO command lines) will produce temperature adjusted E and ν values. Definition of these functions is optional. If not defined by the user a constant temperature scaling factor of 1.0 will be used.

- The hydra plasticity material model uses an element specific coordinate frame. The coordinate frame used is specified by the COORDINATE SYSTEM command line, where the coordinate_system_name references a coordinate system defined by a separate COORDINATE SYSTEM command block. For the hydra plasticity model, this coordinate system can be either a rectangular or a cylindrical coordinate system. The COORDINATE SYSTEM defines a local R, S, and T frame at each element. This initial coordinate system can be additionally rotated to give the final material directions at each material point.

The first rotation of the initial coordinate system is defined using the `DIRECTION FOR ROTATION` and `ALPHA` command lines. The axis for rotation of the initial coordinate system is specified by the `DIRECTION FOR ROTATION` command line, `1` corresponds to the initial coordinate system local `R` axis, `2` corresponds to the initial coordinate system local `S` axis, and `3` corresponds to the initial coordinate system local `T` axis. The right hand rule angle of rotation about this axis is given by `ALPHA`. This rotation yields an intermediate coordinate system.

A secondary rotation of the intermediate coordinate system may be defined using the `SECOND DIRECTION FOR ROTATION` and `SECOND ALPHA` command lines. The axis for rotation of the intermediate coordinate system is specified by the `SECOND DIRECTION FOR ROTATION` command line, `1` corresponds to the intermediate coordinate system local `R` axis, `2` corresponds to the intermediate coordinate system local `S` axis, and `3` corresponds to the intermediate coordinate system local `T` axis. The right hand rule angle of rotation about this axis is given by `SECOND ALPHA`. The resulting coordinate system gives the final `R`, `S`, and `T` coordinate frame for the material directions.

At a minimum, the `COORDINATE SYSTEM` command line must be specified. The `DIRECTION FOR ROTATION`, `ALPHA`, `SECOND DIRECTION FOR ROTATION`, and `SECOND ALPHA` command lines are optional, and default to `1`, `0.0`, `2`, and `0.0`, respectively (effectively specifying no additional rotations of the coordinate system specified by the `COORDINATE SYSTEM` command line).

- The ratio of the normal yield stress to the reference yield stress ($\bar{\sigma}_r$) in the $\bar{\mathbf{e}}_1\bar{\mathbf{e}}_1$ material direction is defined with the `R11` command line. The default is 1.0.

- The ratio of the normal yield stress to the reference yield stress ($\bar{\sigma}_r$) in the $\bar{\mathbf{e}}_2\bar{\mathbf{e}}_2$ material direction is defined with the `R22` command line. The default is 1.0.

- The ratio of the normal yield stress to the reference yield stress ($\bar{\sigma}_r$) in the $\bar{\mathbf{e}}_3\bar{\mathbf{e}}_3$ material direction is defined with the `R33` command line. The default is 1.0.

- The ratio of the shear yield stress to the reference shear yield stress in the $\bar{\mathbf{e}}_1\bar{\mathbf{e}}_2$ material direction is defined with the `R12` command line. The default is 1.0. Note that the reference shear stress is equal to $\frac{1}{\sqrt{3}}\bar{\sigma}_r$.

- The ratio of the shear yield stress to the reference shear yield stress in the $\bar{\mathbf{e}}_2\bar{\mathbf{e}}_3$ material direction is defined with the `R23` command line. The default is 1.0. Note that the reference shear stress is equal to $\frac{1}{\sqrt{3}}\bar{\sigma}_r$.

- The ratio of the shear yield stress to the reference shear yield stress in the $\bar{\mathbf{e}}_3\bar{\mathbf{e}}_1$ material direction is defined with the `R31` command line. The default is 1.0. Note that the reference shear stress is equal to $\frac{1}{\sqrt{3}}\bar{\sigma}_r$.

- The hardening function is defined with the `HARDENING FUNCTION` command line. The `hardening_func_name` references a function defined by a separate `FUNCTION` command block. The hardening function must be defined using a piecewise multivariate function (`TYPE = PIECEWISE MULTIVARIATE`). The piecewise multivariate function allows the user to define the behavior of a single dependent variable with respect to one or more independent variables. In this case, hardening (which defines the reference yield stress) can

be defined with dependencies on equivalent plastic strain $\bar{\varepsilon}^p$, and optionally on equivalent plastic strain rate $\dot{\bar{\varepsilon}}^p$, and temperature T_t. The hardening function must be defined by the user and must include dependence of the reference yield stress on equivalent plastic strain. All other dependencies are optional.

Shown below is an example of the format required to define a hardening function using a piecewise multivariate function:

```
begin function multivariate_hardening_function
  type = piecewise multivariate
  column titles plastic_strain plastic_strain_rate temperature \#
    yield_stress
  begin values
    0.00  -3.0   100.0   1.52294e+09
    0.10  -3.0   100.0   1.76488e+09
    0.20  -3.0   100.0   1.78225e+09
    0.30  -3.0   100.0   1.79298e+09
    0.40  -3.0   100.0   1.80086e+09
    0.50  -3.0   100.0   1.80713e+09
                .
                .
                .

  end
end
```

The `column titles` line contains the dependent variable `yield_stress` in the last column and the independent variables `plastic_strain`, `plastic_strain_rate`, and `temperature` in the first three columns. The specific column name shown must be used if dependency on the variable is desired. The piecewise multivariate function requires that the independent variables create a fully populated grid where points in each dimension are evenly spaced. For performance reasons, the independent variables that are expected to vary the most in the analyses should be placed in the left columns, and variables that are expected to vary the least in the right columns.

- The flag `PLASTIC STRAIN RATE LOG FLAG` is used to specify whether the data for $\dot{\bar{\varepsilon}}^p$ provided in the hardening function is $\log_{10}$ or not. If the command line `PLASTIC STRAIN RATE LOG FLAG` is set equal to *true*, the values provided in the piecewise multivariate function defining the hardening for the material are $\log_{10}\left(\dot{\bar{\varepsilon}}^p\right)$. This is the case in the example provided above, where the equivalent plastic strain rate value given as -3.0 corresponds to an actual strain rate of 0.001 1/s. The default setting is *false*.

- The value for the specific heat, C, which is the amount of heat required to change the temperature of a unit mass of the material by one degree, is defined with the `SPECIFIC HEAT` command line. The specific heat must be specified by the user, there is no default value.

- The temperature scaling factor function for C is defined by the `SPECIFIC HEAT FUNCTION` command line. The `specific_heat_func_name` references a function defined by a separate `FUNCTION` command block. The function should define temperature dependent scaling factors that when applied to C (defined by the `SPECIFIC HEAT` command line) will produce the temperature adjusted C value. Definition

of this function is optional. If not defined by the user, a constant temperature scaling factor of 1.0 will be used.

- The value for the inelastic heat fraction, η, is defined with the INELASTIC HEAT FRACTION command line. The inelastic heat fraction defines the fraction of plastic work that acts to heat the material. The default value is 1.0.

- The flag ADIABATIC ANALYSIS FLAG is used to specify whether an adiabatic analysis should be performed. If the command line ADIABATIC ANALYSIS FLAG is set equal to *true*, the material model will calculate a change in temperature in the material due to plastic work, and add the plastic work temperature change to the externally defined temperature, before determining temperature dependent material properties. If the command line ADIABATIC ANALYSIS FLAG is set equal to *false*, the material model will still calculate the plastic work performed on the material, but will make no adjustment to the externally defined temperature before determining temperature dependent material properties. The material model will continue to calculate the plastic work heat increment and heat flux, making them available through state variables for coupled analysis. The default setting is *false*.

- The failure function is defined with the FAILURE FUNCTION command line. The failure_func_name references a function defined by a separate FUNCTION command block. The failure function, like the hardening function, must be defined using a piecewise multivariate function (TYPE = PIECEWISE MULTIVARIATE). The failure function defines the failure strain, with dependencies on stress triaxiality $\varkappa$, and optionally on equivalent plastic strain rate $\dot{\bar{\varepsilon}}^p$, temperature T_t, and Lode parameter Θ_p (where Θ_p is calculated as $\cos(3\Theta)$ and Θ is the Lode angle). The failure function need only be defined when FAILURE ANALYSIS FLAG is set equal to *true*. When defined, the failure function must include dependence of the failure strain on stress triaxiality. All other dependencies are optional.

Shown below is an example of the format required to define a failure function using a piecewise multivariate function:

```
begin function multivariate_failure_function
  type = piecewise multivariate
  column titles stress_triaxiality plastic_strain_rate temperature \#
    lode_parameter failure_strain
  begin values
    -0.333  -3.0   100.0  -1.00    102.816
     0.000  -3.0   100.0  -1.00    0.288
     0.333  -3.0   100.0  -1.00    0.431
     0.666  -3.0   100.0  -1.00    0.395
     0.999  -3.0   100.0  -1.00    0.275
                .
                .
                .
  end
end
```

The column titles line contains the dependent variable failure_strain in the last column and the independent variables stress_triaxiality, plastic_strain_rate,

`temperature`, and `lode_parameter` in the first four columns. The specific column name shown must be used if dependency on the variable is desired. The piecewise multivariate function requires that the independent variables create a fully populated grid where points in each dimension are evenly spaced. For performance reasons, the independent variables that are expected to vary the most in the analyses should be placed in the left columns, and variables that are expected to vary the least in the right columns.

- The fracture energy for the material, G_f, the energy required to form a unit area of crack surface, is defined with the `FRACTURE ENERGY` command line. This default value is 1.0.

- The decay value at which an element or integration point is flagged for removal from the analysis, φ_r, is specified by the `ELEMENT REMOVAL DECAY VALUE` command line. The default value is 0.001.

- The flag `FAILURE ANALYSIS FLAG` is used to specify whether the failure modeling capabilities of the material model should be enabled. If the command line `FAILURE ANALYSIS FLAG` is set equal to *true*, the material model will include failure. If the command line `FAILURE ANALYSIS FLAG` is set equal to *false*, the failure capabilities of the material model will be disabled. In the latter case, the user need not define a failure function, fracture energy, or element removal decay value. The default setting is *false*.

Output variables available for this model are listed in Table 17-9.

17.15.3. Theory

Strain Decomposition

The hydra plasticity model makes use of an additive decomposition of the strain tensor into elastic and inelastic (plastic) components. The rate form of the additive decomposition is given by:

$$\dot{\varepsilon} = \dot{\varepsilon}^e + \dot{\varepsilon}^p \tag{17.54}$$

where $\dot{\varepsilon}$ is the total strain rate that consists of an elastic ($\dot{\varepsilon}^e$) and plastic ($\dot{\varepsilon}^p$) component.

Elasticity

In the elastic regime, the true stress σ is related to the total strain ε, by Hooke's law:

$$\sigma = \mathbb{C}(T_t)\varepsilon \tag{17.55}$$

where $\mathbb{C}(T_t)$ is the fourth order elastic moduli tensor that may optionally depend on the material temperature T_t. Isotropic elasticity is assumed, so the elastic moduli tensor is fully defined by two elastic constants. In the hydra plasticity model the two elastic constants are constrained to be Young's modulus and Poisson's ratio. These constants each may optionally depend on temperature T_t.

Table 17-9. State Variables for HYDRA PLASTICITY Model (Section 17.15)

Name	Description
ITERATIONS	Number of iterations required for convergence of the Newton algorithm.
SEFF	Effective Stress $\phi(\sigma)$, note may $\neq \sigma_e = \sqrt{3J_2}$
SBAR	Yield stress based on the current size of the yield surface. $\bar{\sigma}$
EQPS	Equivalent Plastic Strain $\bar{\varepsilon}^p = \int_0^t \dot{\bar{\varepsilon}}^p \, \mathrm{d}t$
EQPS_RATE	Equivalent Plastic Strain Rate $\dot{\bar{\varepsilon}}^p = \sqrt{\frac{2}{3}\dot{\varepsilon}^p : \dot{\varepsilon}^p}$
HEAT_FLUX	Heat Flux $r^p = \eta(\sigma : \dot{\varepsilon}^p)$
HEAT_INCREMENT	Heat Increment Δr^p
TEMP_TOTAL	Total temperature including change in temperature due to plastic heating. $T_t = T + \Delta T_p$, where T is the externally defined temperature.
EQPS_DT	Total accumulated change in temperature due to plastic heating. ΔT_p
FAIL_STRESS	Effective stress at the time of failure initiation. $\sigma_e^f = \sqrt{3J_2^f}$
UFAIL	Displacement over which the integration point's strength and stiffness are reduced to zero following failure initiation. $u^f = \frac{2G_f}{\sigma_e^f}$
DAMAGE	Damage, equal to 0 at failure initiation, increasing to 1 at full material strength and stiffness degradation. D
PRESSURE	Pressure $p = -\frac{1}{3}\mathrm{tr}(\sigma)$
SECOND_STRESS_INVARIANT	Second Stress Invariant $J_2 = \frac{1}{2}\left[\mathrm{tr}\left(\sigma^2\right) - \frac{1}{3}\mathrm{tr}(\sigma)^2\right]$
THIRD_STRESS_INVARIANT	Third Stress Invariant $J_3 = \frac{1}{3}\left[\mathrm{tr}\left(\sigma^3\right) - \mathrm{tr}\left(\sigma^2\right)\mathrm{tr}(\sigma) + \frac{2}{9}\mathrm{tr}(\sigma)^3\right]$
STRESS_TRIAXIALITY	Stress Triaxiality $\varkappa = -\frac{p}{\sigma_e}$
LODE_PARAMETER	Lode Parameter $\Theta_p = \cos(3\Theta)$, where $\Theta = \left(\frac{J_3}{2}\right)\left(\frac{3}{J_2}\right)^{1.5}$ is the Lode angle.

Yield Surface

The hydra plasticity model makes use of the Hill yield surface. The Hill yield surface may be orthotropic, and assumes orthogonal principal material directions. The yield surface is defined as follows,

$$\phi^2\left(\sigma_{ij}\right) = F\left(\sigma_{22} - \sigma_{33}\right)^2 + G\left(\sigma_{33} - \sigma_{11}\right)^2 + H\left(\sigma_{11} - \sigma_{22}\right)^2$$

$$+ 2L\sigma_{23}^2 + 2M\sigma_{31}^2 + 2N\sigma_{12}^2$$

(17.56)

where the coefficients F, G, H, L, M, and N are given by the following.

$$F = \frac{(\bar{\sigma}_r)^2}{2}\left[\frac{1}{\left(\sigma_{22}^y\right)^2} + \frac{1}{\left(\sigma_{33}^y\right)^2} - \frac{1}{\left(\sigma_{11}^y\right)^2}\right] \quad ; \quad L = \frac{(\bar{\sigma})^2}{2}\left[\frac{1}{\left(\tau_{23}^y\right)^2}\right]$$

$$G = \frac{(\bar{\sigma}_r)^2}{2}\left[\frac{1}{\left(\sigma_{33}^y\right)^2} + \frac{1}{\left(\sigma_{11}^y\right)^2} - \frac{1}{\left(\sigma_{22}^y\right)^2}\right] \quad ; \quad M = \frac{(\bar{\sigma}_r)^2}{2}\left[\frac{1}{\left(\tau_{31}^y\right)^2}\right] \quad (17.57)$$

$$H = \frac{(\bar{\sigma}_r)^2}{2}\left[\frac{1}{\left(\sigma_{11}^y\right)^2} + \frac{1}{\left(\sigma_{22}^y\right)^2} - \frac{1}{\left(\sigma_{33}^y\right)^2}\right] \quad ; \quad N = \frac{(\bar{\sigma}_r)^2}{2}\left[\frac{1}{\left(\tau_{12}^y\right)^2}\right]$$

In Equation 17.57, σ_{11}^y, σ_{22}^y, σ_{33}^y, τ_{12}^y, τ_{23}^y, and τ_{31}^y represent the three normal and three shear yield stresses in the three material coordinate directions, and $\bar{\sigma}_r$ a reference yield stress that is defined by the hardening function. The six independent yield stresses, are specified by the user through the definition of the following six yield ratios.

$$R_{11} = \frac{\sigma_{11}^y}{\bar{\sigma}} \quad ; \quad R_{12} = \sqrt{3}\frac{\tau_{12}^y}{\bar{\sigma}}$$

$$R_{22} = \frac{\sigma_{22}^y}{\bar{\sigma}} \quad ; \quad R_{23} = \sqrt{3}\frac{\tau_{23}^y}{\bar{\sigma}} \quad (17.58)$$

$$R_{33} = \frac{\sigma_{33}^y}{\bar{\sigma}} \quad ; \quad R_{31} = \sqrt{3}\frac{\tau_{31}^y}{\bar{\sigma}}$$

If all of the above ratios are set equal to 1.0 ($R_{ij} = 1$) the Mises yield surface is recovered.

Hardening

The hydra plasticity model assumes isotropic hardening. Hardening of the material is defined by a hardening function.

169

$$\bar{\sigma}_r = f_H\left(\bar{\varepsilon}^p, \dot{\bar{\varepsilon}}^p, T_t\right) \tag{17.59}$$

The hardening function defines the reference yield stress ($\bar{\sigma}_r$) for the material. The hardening function has dependence on the equivalent plastic strain ($\bar{\varepsilon}^p$), and optionally may depend on the equivalent plastic strain rate ($\dot{\bar{\varepsilon}}^p$) and/or temperature ($T_t$). The equivalent plastic strain is defined as follows:

$$\bar{\varepsilon}^p = \int_0^t \dot{\bar{\varepsilon}}^p \mathrm{d}t \tag{17.60}$$

where $\dot{\bar{\varepsilon}}^p$ is the equivalent plastic strain rate calculated from the plastic strain rates ($\dot{\varepsilon}^p$) as follows.

$$\dot{\bar{\varepsilon}}^p = \sqrt{\frac{2}{3}\dot{\varepsilon}^p : \dot{\varepsilon}^p} \tag{17.61}$$

Failure

Failure in the hydra plasticity model is comprised of two distinct phases. The first, pre-failure initiation phase, involves the calculation of a failure initiation metric that is used to determine when the failure initiation criterion has been met. The second, post-failure initiation phase, involves the accumulation of damage in the material (with increasing plastic deformation) and the degradation of the material's strength and stiffness.

The failure initiation criterion is strain based. The criterion is satisfied when the failure measure (fm) equation is satisfied.

$$fm = \int \frac{\mathrm{d}\bar{\varepsilon}^p}{\bar{\varepsilon}_f^p\left(\varkappa, \dot{\bar{\varepsilon}}^p, \Theta_p, T_t\right)} \geq 1.0 \tag{17.62}$$

The failure strain $\bar{\varepsilon}_f^p\left(\varkappa, \dot{\bar{\varepsilon}}^p, \Theta_p, T_t\right)$ is provided by the failure function f_F, and is dependent on stress triaxiality ($\varkappa$), and optionally on plastic strain rate ($\dot{\bar{\varepsilon}}^p$), Lode angle ($\Theta$, through the Lode parameter, Θ_p), and/or temperature (T_t). Stress triaxiality is calculated as follows:

$$\varkappa = -\frac{p}{\sigma_e} \tag{17.63}$$

where p is the pressure and σ_e is the effective stress. The pressure is given by the following,

$$p = -\frac{1}{3}\mathrm{tr}\left(\sigma\right) \tag{17.64}$$

and the effective stress by the following.

170

$$\sigma_e = \sqrt{3J_2} \tag{17.65}$$

In the above equation J_2 is the second stress invariant and is calculated from the stress σ as follows.

$$J_2 = \frac{1}{2}\left[\operatorname{tr}\left(\sigma^2\right) - \frac{1}{3}\operatorname{tr}(\sigma)^2\right] \tag{17.66}$$

The Lode parameter (Θ_p) is a function of the Lode angle (Θ),

$$\Theta = \left(\frac{J_3}{2}\right)\left(\frac{3}{J_2}\right)^{1.5} \tag{17.67}$$

where J_3 is the third stress invariant calculated from the stress σ as follows.

$$J_3 = \frac{1}{3}\left[\operatorname{tr}\left(\sigma^3\right) - \operatorname{tr}\left(\sigma^2\right)\operatorname{tr}(\sigma) + \frac{2}{9}\operatorname{tr}(\sigma)^3\right] \tag{17.68}$$

The Lode parameter is then calculated as follows.

$$\Theta_p = \cos(3\Theta) \tag{17.69}$$

Note that the Lode parameter can vary from -1.0 (triaxial compression) to 1.0 (triaxial tension), with a value of 0.0 representing the pure shear stress state.

Once the failure initiation criterion has been satisfied, increasing plastic deformation of the material results in the accumulation of damage (D). The rate of damage accumulation is governed by the following rate equation:

$$\dot{D} = \frac{L\dot{\bar{\varepsilon}}_f^p}{u^f} \tag{17.70}$$

where $\dot{\bar{\varepsilon}}_f^p$ is the equivalent plastic strain rate associated with plastic deformation of the material occurring after the initiation of failure, L is the characteristic length associated with the material integration point (taken as $\sqrt[3]{V_{ipt}}$, where V_{ipt} is the volume of material associated with the material integration point), and u^f is the failure displacement. The failure displacement is calculated based on the state of the material at the time of failure initiation as follows:

$$u^f = \frac{2G_f}{\sigma_e^f} \tag{17.71}$$

where G_f is the material's fracture energy and σ_e^f is the effective stress at the instant of failure initiation. The instantaneous damage value is therefore given by the following.

171

$$D = \int_0^t \dot{D} \mathrm{d}t \qquad (17.72)$$

It is important to note that the use of the characteristic length (L) associated with the material integration point in Equation 17.70 is an approximate way to remove some of the mesh sensitivity effects associated with material softening. The implementation attempts to get the amount of energy dissipated by the failure process to match the fracture energy for the material, regardless of the mesh size selected.

As damage accumulates in the material, both the strength and stiffness of the material are degraded. This degradation takes the following form,

$$\bar{\sigma}_r{}^D = (1 - D)\sigma_e^f = \varphi\sigma_e^f \qquad (17.73)$$

$$\mathbb{C}^D = (1 - D)\mathbb{C} = \varphi\mathbb{C} \qquad (17.74)$$

where $\varphi = (1 - D)$ defines the decay value for the damaged material, and $\bar{\sigma}_r{}^D$ and $\mathbb{C}^D$ define the damaged yield strength and elastic moduli tensor, respectively. The material strength and stiffness are continually degraded with increasing plastic deformation until the material is fully damage ($D = 1$) or the decay value (φ) has reached a critical decay value specified by the user (φ_r), at which point the material point is flagged for removal from the analysis. It is important to note that the pressure stresses are only degraded when the pressure is tensile (negative pressures). When the pressure is compressive, the pressure stresses are not degraded, which results in the material behaving increasingly like an incompressible fluid as damage is accumulated.

Plastic Heating

The hydra plasticity model includes the ability to calculate changes in the material's temperature due to heating resulting from plastic deformation of the material. The heat flux (r^p) per unit volume associated with a given plastic strain rate ($\dot{\varepsilon}^p$) is calculated as:

$$r^p = \eta\sigma\dot{\varepsilon}^p \qquad (17.75)$$

where η is a user defined inelastic heat fraction and σ is the instantaneous stress. Given the specific heat $C(T_t)$ for the material, which may be a function of the temperature (T_t), the heat equation to be solved at each material integration point is,

$$\dot{T}_t C(T_t)\rho = r^p \qquad (17.76)$$

where T_t is the temperature and $\dot{T}_t$ is the rate of change of the temperature.

17.15.4. Implementation

Trial Stress

The hydra plasticity model uses a predictor-corrector algorithm for integrating the constitutive model. Since the elastic constants of the hydra plasticity model may depend on temperature, the fourth order elastic moduli tensor $\mathbb{C}$ is first updated to account for the current (step n) temperature (T_t^n). If `ADIABATIC ANALYSIS FLAG` is set equal to *true*, the current temperature is calculated as follows:

$$T_t^{(n)} = T^{(n)} + \Delta T_p^{(n-1)} \tag{17.77}$$

where $T^{(n)}$ is the current time step (n) externally defined temperature, and $T_p^{(n-1)}$ is the accumulated temperature change in the material resulting from plastic work up through the previous time step ($n-1$). If `ADIABATIC ANALYSIS FLAG` is set equal to *false*, the current temperature is taken directly as the externally defined temperature.

$$T_t^{(n)} = T^{(n)} \tag{17.78}$$

Given the current step strain rate $\dot{\varepsilon}^{(n)}$, time increment $\Delta t^{(n)}$, and temperature adjusted elastic moduli tensor $\mathbb{C}\left(T_t^{(n)}\right)$, a trial stress is calculated assuming an elastic response:

$$\sigma_{tr}^{(n)} = \sigma^{(n-1)} + \mathbb{C}\left(T_t^{(n)}\right)\Delta t^{(n)}\dot{\varepsilon}^{(n)} \tag{17.79}$$

where $\sigma^{(n-1)}$ is the converged stress state from the previous time step. If the trial stress lies inside the yield surface (i.e. if $\phi(\sigma_{tr}^{(n)}) \leq \bar{\sigma}(\bar{\varepsilon}^{p^{(n)}}, \dot{\bar{\varepsilon}}^{p^{(n)}}, T_t^{(n)})$), then the step is elastic, the plastic strain increment and plastic strain rate are zero, and the stress is given by the trial stress ($\sigma_{tr}^{(n)}$). If the trial stress is outside of the yield surface (i.e. if $\phi(\sigma_{tr}^{(n)}) > \bar{\sigma}(\bar{\varepsilon}^{p^{(n)}}, \dot{\bar{\varepsilon}}^{p^{(n)}}, T_t^{(n)})$) then the step will include plastic deformation of the material.

Return Mapping Algorithm

When the elastic trial stress falls outside of the yield surface, the model uses an iterative algorithm to determine the increment of plastic strain that occurs during the time step and the final stress state in the material. During the solution process, normality is enforced:

$$R = \Delta\varepsilon^{p^{(n)}} - \Delta\gamma^{(n)}\frac{\partial\phi}{\partial\sigma} = 0 \tag{17.80}$$

and the final stress state for the increment (n) is required to resides on the yield surface.

$$f = \phi\left(\sigma^{(n)}\right) - \bar{\sigma}\left(\bar{\varepsilon}^{p^{(n)}}, \dot{\bar{\varepsilon}}^{p^{(n)}}, T_t^{(n)}\right) = 0 \tag{17.81}$$

173

A Newton-Raphson search algorithm is implemented. Using $\Delta\gamma^{(k)}$ (which conveniently is equal to $\Delta\bar{\varepsilon}^{p^{(k)}}$) and σ^k as the solution variables, the iterative algorithm is as follows:

$$\Delta\gamma^{(k+1)} = \Delta\gamma^{(k)} + \Delta(\Delta\gamma)^{(k)}$$

$$\sigma^{(k+1)} = \sigma^{(k)} + \Delta\sigma^{(k)} \tag{17.82}$$

$$\Delta\varepsilon^{p^{(k)}} = \mathbb{C}\left(T_t^{(n,k=0)}\right)^{-1}\left(\sigma_{tr}^{(n)} - \sigma^{(k)}\right)$$

where $\Delta\gamma^{(0)} = 0$ and $\sigma^{(0)} = \sigma_{tr}$. The increment in plastic strain is given by:

$$\Delta(\Delta\gamma)^{(k)} = \frac{f^{(k)} - R^{(k)}\mathscr{L}^{(k)}\dfrac{\partial\phi^{(k)}}{\partial\sigma}}{\dfrac{\partial\phi^{(k)}}{\partial\sigma}\mathscr{L}^{(k)}\dfrac{\partial\phi^{(k)}}{\partial\sigma} + H'^{(k)}} \tag{17.83}$$

where $f^{(k)}$ and $R^{(k)}$ are defined for the kth increment using Equation 17.81 and 17.80 as follows:

$$f^{(k)} = \phi\left(\sigma^{(k)}\right) - \bar{\sigma}\left(\bar{\varepsilon}^{p^{(k)}}, \dot{\bar{\varepsilon}}^{p^{(k)}}, T_t^{(k)}\right)$$

$$\tag{17.84}$$

$$R^{(k)} = \Delta\varepsilon^{p^{(k)}} - \Delta\gamma^{(k)}\frac{\partial\phi}{\partial\sigma}^{(k)}$$

and the elastoplastic tangent ($\mathscr{L}^{(k)}$) for the kth increment, as follows.

$$\mathscr{L}^{(k)} = \mathbb{C}\left(T_t^{(n,k=0)}\right)^{-1} + \Delta\gamma\frac{\partial^2\phi^{(k)}}{\partial\sigma\partial\sigma} \tag{17.85}$$

In Equation 17.83 the slope of the hardening curve ($H'^{(k)}$) is required. Because the hardening behavior in the hydra plasticity model may depend on a number of variables defined in a multivariate function, the calculation of $H'^{(k)}$ is performed using a finite difference approximation. This is handle by incrementing the current equivalent plastic strain ($\bar{\varepsilon}^{p^{(k)}}$) and updating the other parameters ($T_t^{(k)}$ and $\dot{\bar{\varepsilon}}^{p^{(k)}}$) upon which the hardening function depends based on the incremented plastic strain, before the hardening function is evaluated to obtain the values used in the finite difference calculation. This approach ensures that the hardening slope appropriately captures not only the effect of the change in plastic strain, but also the change in material temperature and strain rate generated by a change in the plastic strain. Equations 17.83 and 17.85 also require the first and second partial derivatives of $\phi^{(k)}$. Since ϕ is known, the values are calculated at each step using the current stress state ($\sigma^{(k)}$) and predetermined analytical

expressions (not described here). Finally, the search direction in stress space ($\Delta\sigma^{(k)}$) utilized in Equation 17.82 is given by the following.

$$\Delta\sigma^{(k)} = -\mathscr{L}^{(k)}\left(R^{(k)} + \Delta(\Delta\gamma)^{(k)}\frac{\partial\phi^{(k)}}{\partial\sigma}\right) \tag{17.86}$$

It has been observed that the Newton-Raphson algorithm described above does not always converge when non von Mises yield surfaces are employed. To improve the robustness of the return mapping algorithm, a line search is performed during each Newton-Raphson (k) iteration. This algorithm takes the following form.

$$\Delta\gamma^{(j)} = \Delta\gamma^{(k)} + \alpha^{(j)}\Delta(\Delta\gamma)^{(k)}$$

$$\sigma^{(j)} = \sigma^{(k)} + \alpha^{(j)}\Delta\sigma^{(k)} \tag{17.87}$$

where $\alpha^{(j)} \in (0,1]$ is the line search parameter that is determined iteratively using a simple algorithm not described here. $\alpha^{(0)} = 1$ is used to initialize the line search. Note that if $\alpha^{(j)} = 1$ then the exact Newton-Raphson algorithm search direction and magnitude is recovered. The $\Delta\gamma^{(k+1)}$ and $\sigma^{(k+1)}$ values are then given by the converged ($\alpha^{(j=\text{converged})}$) lines search values.

$$\Delta\gamma^{(k+1)} = \Delta\gamma^{(k)} + \alpha^{(j=\text{converged})}\Delta(\Delta\gamma)^{(k)}$$

$$\sigma^{(k+1)} = \sigma^{(k)} + \alpha^{(j=\text{converged})}\Delta\sigma^{(k)} \tag{17.88}$$

During each Newton-Raphson (k) and line search (j) iteration, the variables that affect the material hardening and failure parameters are updated, and the hardening and failure functions re-evaluated. These parameters include the plastic strain ($\bar{\varepsilon}^p$), plastic strain rate ($\dot{\bar{\varepsilon}}^p$), temperature ($T_t$), stress triaxiality ($\varkappa$), and lode parameter ($\Theta_p$). These parameters are calculated as follows, where m is equal to either k or j depending on whether the process is occurring during the Newton-Raphson (k outer) loop or the line search (j inner) loop of the return mapping algorithm.

$$\bar{\varepsilon}^{p(m)} = \Delta\gamma^{(m)} \tag{17.89}$$

$$\dot{\bar{\varepsilon}}^{p(m)} = \frac{\Delta\gamma^{(m)}}{\Delta t^{(n)}} \tag{17.90}$$

$$T_t^{(m)} = T^{(n)} + \Delta T_p^{(m)} \tag{17.91}$$

175

$$\varkappa^{(m)} = -\frac{p^{(m)}}{\sigma_e^{(m)}} \tag{17.92}$$

$$\Theta_p^{(m)} = \cos\left(3\Theta^{(m)}\right) \tag{17.93}$$

In the above equations, the pressure $(p^{(m)})$ is calculated as follows,

$$p^{(m)} = -\frac{1}{3}\mathrm{tr}\left(\sigma^{(m)}\right) \tag{17.94}$$

the effective stress is calculated as follows,

$$\sigma_e^{(m)} = \sqrt{3J_2^{(m)}} \tag{17.95}$$

and the lode angle is calculated as follows.

$$\Theta^{(m)} = \left(\frac{J_3^{(m)}}{2}\right)\left(\frac{3}{J_2^{(m)}}\right)^{1.5} \tag{17.96}$$

$J_2^{(m)}$ and $J_3^{(m)}$ are the second and third stress invariants, respectively, that are calculated as follows.

$$J_2^{(m)} = \frac{1}{2}\left[\mathrm{tr}\left(\sigma^{(m)^2}\right) - \frac{1}{3}\mathrm{tr}\left(\sigma^{(m)}\right)^2\right] \tag{17.97}$$

$$J_3^{(m)} = \frac{1}{3}\left[\mathrm{tr}\left(\sigma^{(m)^3}\right) - \mathrm{tr}\left(\sigma^{(m)^2}\right)\mathrm{tr}\left(\sigma^{(m)}\right) + \frac{2}{9}\mathrm{tr}\left(\sigma^{(m)}\right)^3\right] \tag{17.98}$$

Plastic Heating

The hydra plasticity model includes the ability to calculate heating of the material resulting from its plastic deformation. The heat flux increment $(r^{p^{(n)}})$ per unit volume for a given time step n is calculated as:

$$r^{p^{(n)}} = \eta\frac{\left(\sigma^{(n)} + \sigma^{(n-1)}\right)}{2} : \Delta\varepsilon^{p^{(n)}} \tag{17.99}$$

where η is a user defined inelastic heat fraction, $\sigma^{(n)}$ is the current time step's stress, $\sigma^{(n-1)}$ is the previous time step's stress, and $\Delta\varepsilon^{p^{(n)}}$ is the current time step's increment in plastic strain. The temperature change added to the integration point resulting from plastic heating $(\Delta T_p^{(n)})$ is given by the following:

$$\Delta T_p^{(n)} = \Delta T_p^{(n-1)} + \Delta \left(\Delta T_p \right)^{(n)} \tag{17.100}$$

where $\Delta T_p^{(n-1)}$ is the accumulated temperature change added to the integration point resulting from plastic heating up through the previous time step, and $\Delta \left(\Delta T_p \right)^{(n)}$ is the additional increment in temperature change resulting from the plastic deformation of the material during the current time step, which is calculated as follows.

$$\Delta \left(\Delta T_p \right)^{(n)} = \frac{r^{p(n)}}{\rho C \left(\frac{T_t^{(n)} + T_t^{(n-1)}}{2} \right)} \tag{17.101}$$

In Equation 17.101 ρ is the material density. Because the specific heat may be temperature dependent, this equation is iteratively solved, with each successive iteration providing an updated $T_t^{(n)}$.

Failure

Material failure is initiated in the model when the failure measure reaches or exceeds 1.0 ($fm \geq 1.0$). The failure measure is calculated as follows:

$$fm^{(n)} = fm^{(n-1)} + \Delta fm^{(n)} \tag{17.102}$$

The increment in the failure measure associated with the current time step ($\Delta fm^{(n)}$) is calculated as follows:

$$\Delta fm^{(n)} = \frac{\Delta \bar{\varepsilon}^{p(n)}}{\bar{\varepsilon}_f^p \left(\varkappa^{(n)}, \dot{\bar{\varepsilon}}^{p(n)}, \Theta_p^{(n)}, T_t^{(n)} \right)} \tag{17.103}$$

where $\Delta \bar{\varepsilon}^{p(n)}$ is the current step's increment in equivalent plastic strain, and $\bar{\varepsilon}_f^p \left(\varkappa^{(n)}, \dot{\bar{\varepsilon}}_{p(n)}, \Theta_p^{(n)}, T_t^{(n)} \right)$ is the failure strain for the current increment, provided by the failure function.

During the time step in which the failure measure just reaches or exceeds the failure initiation criterion, the failure stress (σ_e^f) and failure displacement (u^f) are determined. This is done iteratively during the line search portion of the return mapping algorithm, and consists of a process where the line search parameter ($\alpha^{(j)}$) is successively modified until the failure measure just equals 1.0. The value returned by the hardening function when this occurs is the stress utilized as the failure stress (σ_e^f) and the failure displacement is calculated as follows.

$$u^f = \frac{2G_f}{\sigma_e^f} \tag{17.104}$$

177

where G_f is the material's fracture energy.

Once the failure initiation criterion has been satisfied, the yield strength of the material remains constant and is set equal to the failure stress. In addition, increasing plastic deformation of the material results in the accumulation of damage (D). Damage is calculated as follows:

$$D^{(n)} = D^{(n-1)} + \frac{L\Delta\bar{\varepsilon}_f^{p^{(n)}}}{u^f}$$

(17.105)

where $D^{(n-1)}$ is the damage accumulated in the material through the previous time step, $\Delta\bar{\varepsilon}_f^{p^{(n)}}$ is the change in the equivalent plastic strain associated with the current time step (including only plastic strains that have occurred after the failure initiation criterion has been satisfied), and L is the characteristic length associated with the material integration point (taken as $\sqrt[3]{V_{ipt}}$, where V_{ipt} is the volume of material associated with the material integration point). As damage accumulates in the material, both the strength, and indirectly, the stiffness of the material are degraded. This degradation is accomplished by returning the damage adjusted stresses (σ_D) at the conclusion of the material integration step, as follows.

$$\sigma_D^{(n)} = (1 - D)\sigma^{(n)} = \varphi\sigma^{(n)}$$

(17.106)

where $\varphi = (1 - D)$ defines the decay value for the damaged material. This effectively results in a response where the material strength and stiffness are degraded. It is important to note that the pressure stresses are only degraded when the pressure is tensile (negative pressures). When the pressure is compressive, the pressure stress components of $\sigma^{(n)}$ are not degraded as indicated in Equation 17.106. The material strength and stiffness are continually degraded with increasing plastic deformation until the material is fully damage ($D = 1$) or the decay value (φ) has reached a critical decay value specified by the user (φ_r), at which point the material point is flagged for removal from the analysis.

17.15.5. Verification

The hydra plasticity model was verified through comparison of results from the material model with results obtained from the general metal plasticity model in Abaqus/Explicit (Version 6.14). A series of six element tests were designed to verify all of the hydra plasticity model's capabilities.

Description of Six Element Test Cases

Five test cases in total were performed. For each of the five test cases, six elements were loaded using six different prescribed velocities - the goal being to test the hydra plasticity model performs as expected for a variety of different stress states. These elements are not connected in any way, so each of the six element test cases are actually six single element tests performed at the same time. Elements 1, 2, and 3 were loaded to cause a tension stress state in the σ_{xx}, σ_{yy}, and σ_{zz} components respectively. Elements 4, 5, and 6 were loaded to cause shear dominated stress

178

states - though not pure shear - with the dominant component of stress in the τ_{xx}, τ_{yy}, and τ_{zz} and component respectively. Figure 17-2 shows the six elements in an undeformed (top) and deformed (bottom) configuration. The deformed configuration shown is chosen from an arbitrary state in time prior to the failure of any of the elements and is intended only to illustrate the types of loadings applied.
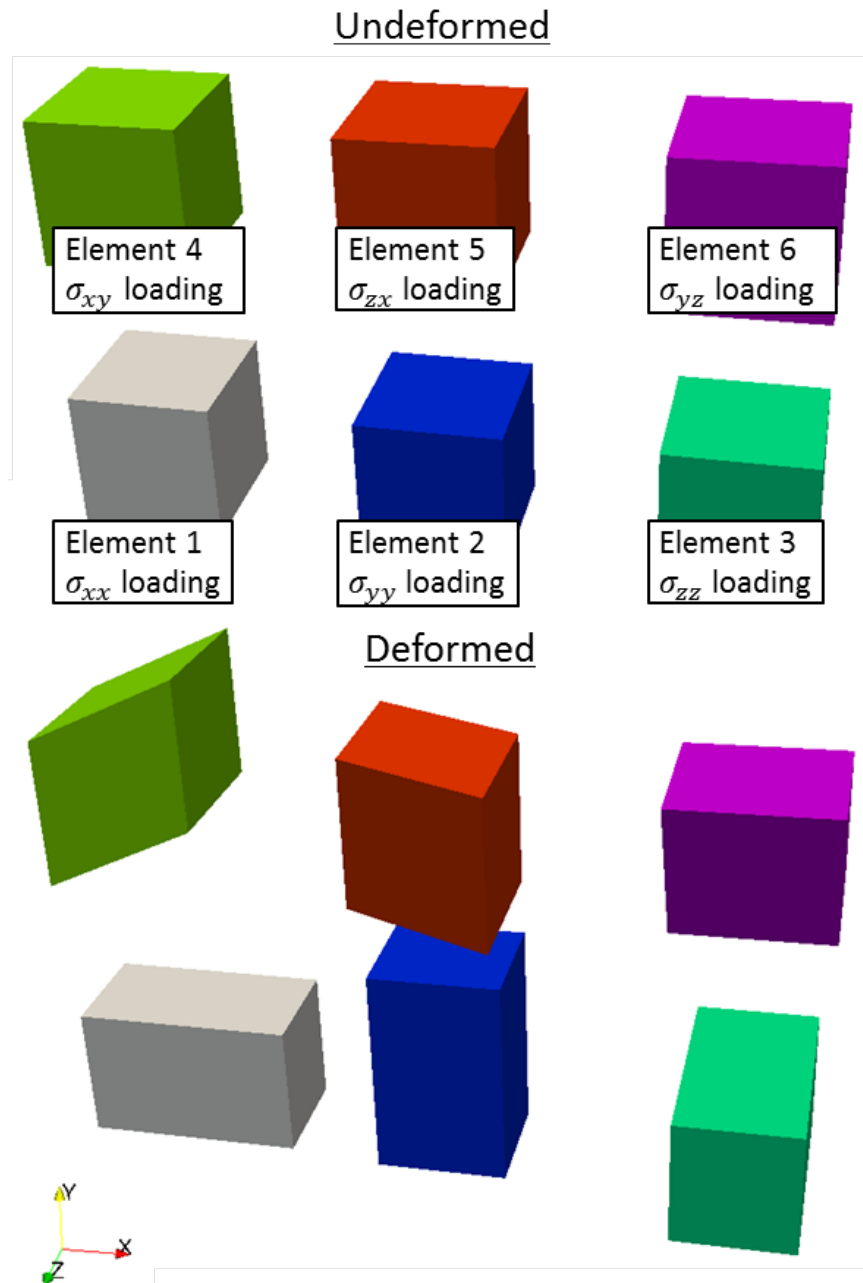

Undeformed

Element 4
σ_{xy} loading

Element 5
σ_{zx} loading

Element 6
σ_{yz} loading

Element 1
σ_{xx} loading

Element 2
σ_{yy} loading

Element 3
σ_{zz} loading

Deformed

Figure 17-2. Undeformed (top) and deformed (bottom) shapes for the six element tests.

A test case matrix was developed to verify the capabilities of the hydra plasticity model over a range of inputs. Table 17-10 below shows the five test cases developed. Test cases 1 through 4 test

different combinations of low (300 K) and high (700 K) temperatures and load and high load rates (resulting in a range of strain rates tested). Test cases 1 through 4 all have Hill ratios that vary between 0.90 and 1.20. Test case 5 tests the adiabatic heating capability of the hydra plasticity model. Because Abaqus does not allow adiabatic heating for a Hill surface all of the Hill ratios were set to 1.0 in the hydra plasticity model to create a Mises yield surface.

Table 17-10. Hydra plasticity test case matrix

Test Case	Temp (K)	load rate (m/s)	Failure Included	Adiabatic Heating Included	Hill Ratios (R11, R22, R33, R12, R23, R31)
1	300	1	Yes	No	1.00, 1.15, 0.90, 1.00, 1.20, 0.95
2	700	1	Yes	No	1.00, 1.15, 0.90, 1.00, 1.20, 0.95
3	300	10	Yes	No	1.00, 1.15, 0.90, 1.00, 1.20, 0.95
4	700	10	Yes	No	1.00, 1.15, 0.90, 1.00, 1.20, 0.95
5	300	1	Yes	Yes	all 1.00 = Mises

A set of hardening data was defined for all five test cases. A python script was used to generate the same data formatted for the hydra plasticity model and an Abaqus general metal plasticity model. The hardening data used for the test cases was dependent on equivalent plastic strain, temperature, and equivalent plastic strain rate. Figure 17-3 shows the data points used to define yield stress as a function equivalent plastic strain at various temperatures and equivalent plastic strain rates. Both the hydra plasticity model and the Abaqus general metal plasticity model use linear interpolation between most of the data points to obtain the yield stress (the exception being the equivalent plastic strain rate direction which for the test cases include the hydra plasticity model had the PLASTIC STRAIN RATE LOG FLAG set to true and Abaqus by default uses log interpolation between for strain rate data).

A set of failure data was also defined for all five test cases. Again, a python script was used to generate the same data formatted for the hydra plasticity model and an Abaqus general metal plasticity model. The failure data used for the test cases was dependent on stress triaxiality, temperature, equivalent plastic strain rate, and Lode angle (via the Lode parameter). Figure 17-4 shows the data points used to define failure strain as a function of stress triaxiality at various temperatures, equivalent plastic strain rates, and Lode parameters. Again, both the hydra plasticity model and the Abaqus general metal plasticity model use linear interpolation between the data points to obtain the failure strain.

Comparison of Results

The five test cases were each run in Sierra using the hydra plasticity model, and Abaqus using the general plasticity model. The results of the five test cases are presented in Figures 4 through 8. Each of these figures presents the equivalent plastic strain in an element versus the unrotated stress in the element in the dominant loading direction. For example, for element 1 which is loaded in the σ_{xx} component the y-axis represents the unrotated stress σ_{xx}; for element 4 which is dominantly loaded in the component the y-axis represents the unrotated stress τ_{xx}; etc.
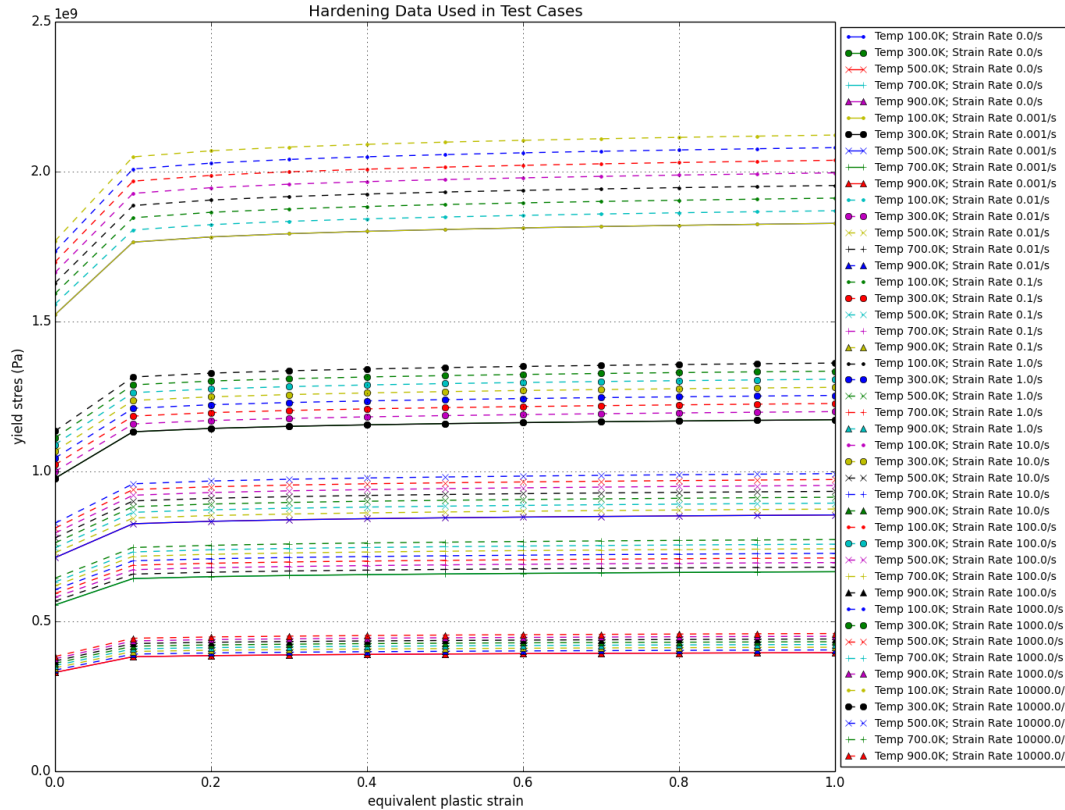
180

Figure 17-3. Hardening data used in all five test cases.

Figure 17-5 shows the results from test case 1, where the temperature was set to 300 K and the loading rate was 1 m/s. The figure shows that there is near perfect agreement between Sierra and Abaqus for all of the tension loaded elements (elements 1, 2, and 3). As expected, the tension loaded elements all fail at the same equivalent plastic strain since they all use the same failure data. The agreement between Sierra and Abaqus is very close for the shear dominated loading elements (elements 4, 5, and 6) with less agreement as the plastic strain increases.

Figure 17-6 shows the results from test case 2, where the temperature was set to 700 K and the loading rate was 1 m/s. Again, the figure shows that there is near perfect agreement between Sierra and Abaqus for all of the tension loaded elements (elements 1, 2, and 3). And again, as expected the tension loaded elements all fail at the same equivalent plastic strain since they all use the same failure data. Also, the agreement between Sierra and Abaqus is very close for the shear dominated loading elements (elements 4, 5, and 6) with less agreement as the plastic strain increases.

Figure 17-7 shows the results from test case 3, where the temperature was set to 300 K and the loading rate was 10 m/s. The figure shows that there is very close agreement between Sierra and Abaqus for all of the tension loaded elements (elements 1, 2, and 3). However, there is some minor disagreement between the two sets at very low plastic strains. And again, as expected the
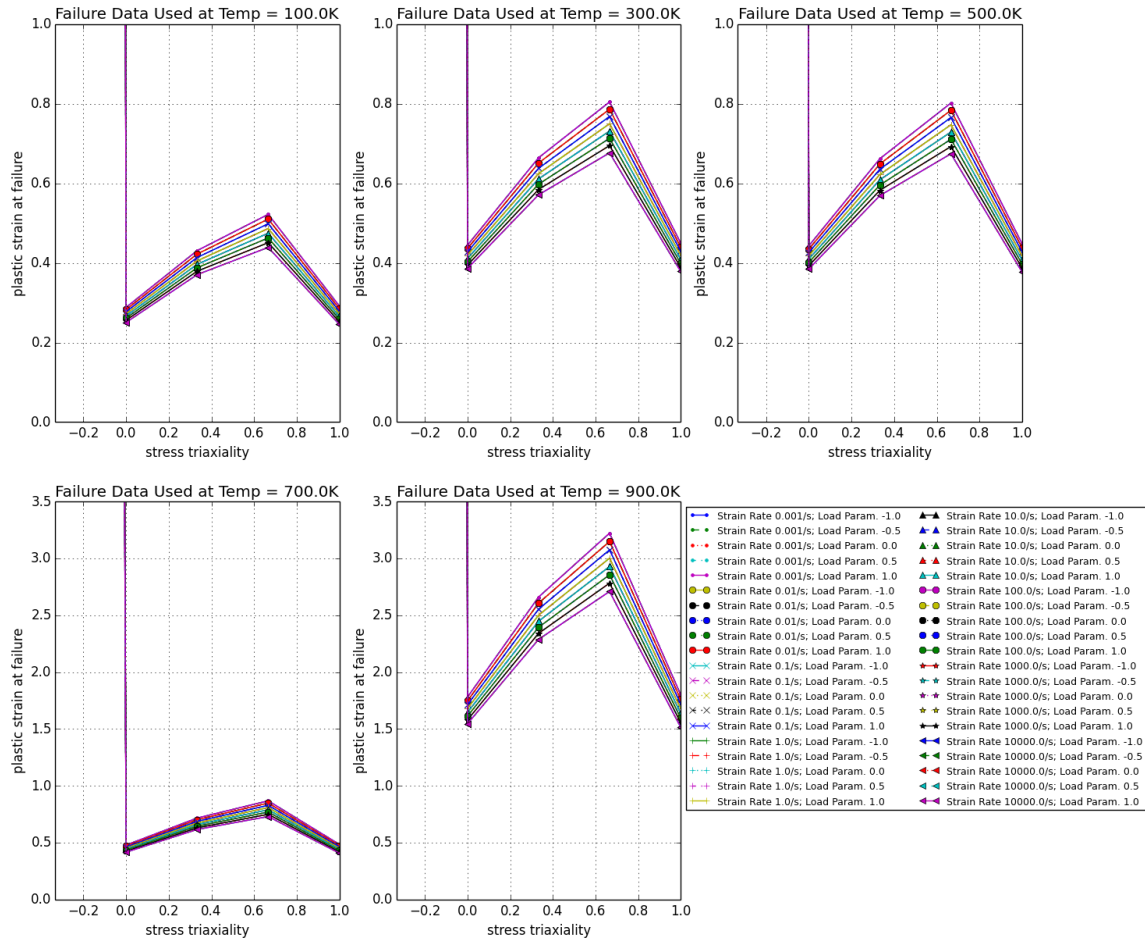
181

Figure 17-4. Failure data used in all five test cases.

tension loaded elements all fail at the same equivalent plastic strain since they all use the same failure data. Also, the agreement between Sierra and Abaqus is very close for the shear dominated loading elements (elements 4, 5, and 6) with less agreement as the plastic strain increases.

Figure 17-8 shows the results from test case 4, where the temperature was set to 700 K and the loading rate was 10 m/s. Again, the figure shows that there is good agreement between Sierra and Abaqus for all of the tension loaded elements (elements 1, 2, and 3). However, there is more discrepancy between the two sets of data at low plastic strains. And again, as expected the tension loaded elements all fail at the same equivalent plastic strain since they all use the same failure data. Also, the agreement between Sierra and Abaqus is very close for the shear dominated loading elements (elements 4, 5, and 6) with less agreement as the plastic strain increases.

Figure 17-9 shows the results from test case 5, the test case that includes adiabatic heating and the Hill ratios are all set to 1.0 (which is equivalent to the Mises surface). For this case the
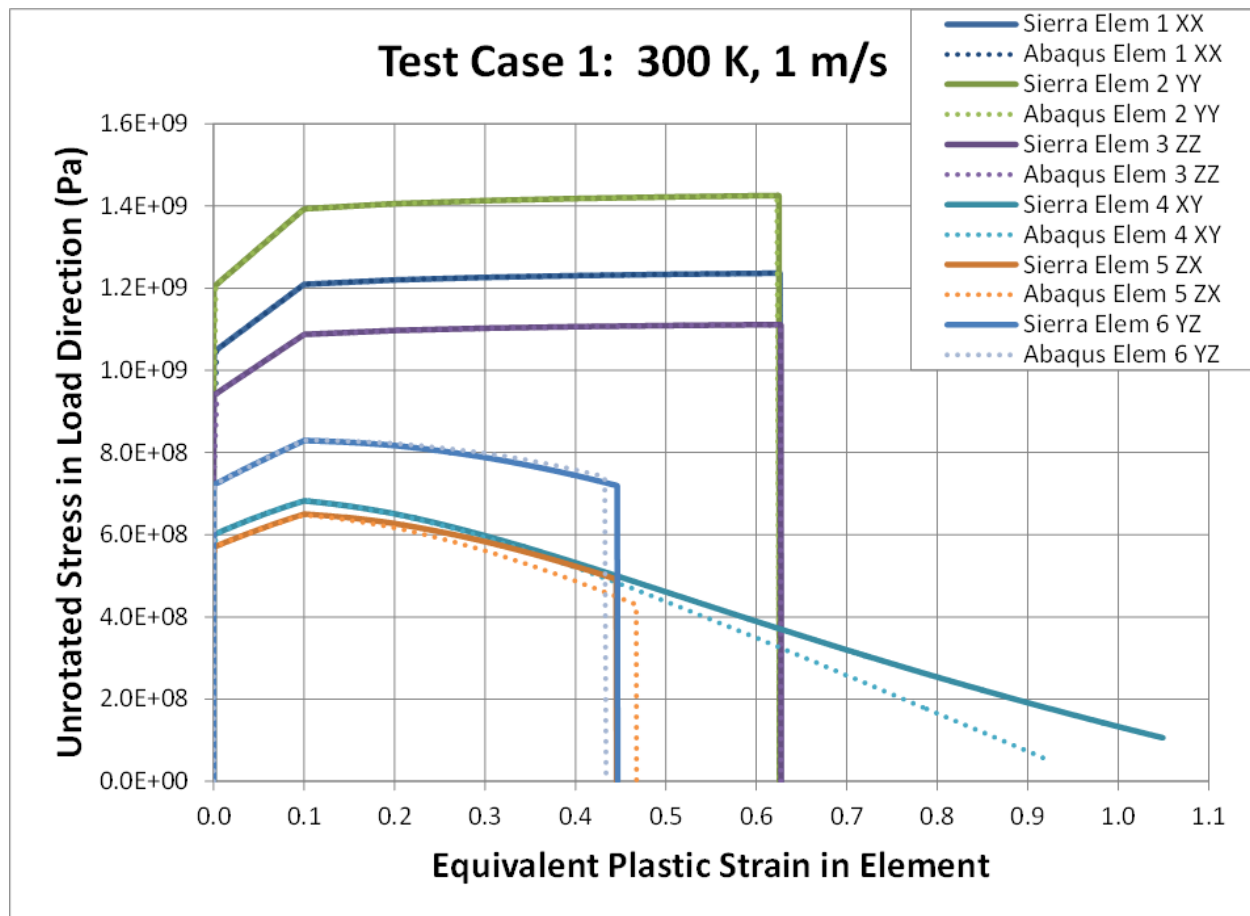
Figure 17-5. Comparison of results from test case 1.

temperature was set to 300 K and the loading rate was 1 m/s. Because the Hill ratios are all the same the results from the three tension elements all lie right on top of each other in the figure; likewise the results from the three shear elements all lie right on top of each other. The figure shows that there is very good agreement between Sierra and Abaqus for all of the tension loaded elements (elements 1, 2, and 3). As expected, the tension loaded elements all fail at the same equivalent plastic strain since they all use the same failure data. The agreement between Sierra and Abaqus is very close for the shear dominated loading elements (elements 4, 5, and 6) with less agreement as the plastic strain increases.

Conclusions

The hydra plasticity model was verified by comparing the hydra plasticity model to a similar model in Abaqus. Five sets of six single element tests that exercised a range of features in the hydra plasticity model were used for the verification. Nearly perfect agreement between the hydra plasticity model and the Abaqus model was found in the tensile loaded elements. At higher temperatures, some minor differences between the two models was seen at low plastic strains. Good agreement was found between the two models in the shear loaded elements; however, there
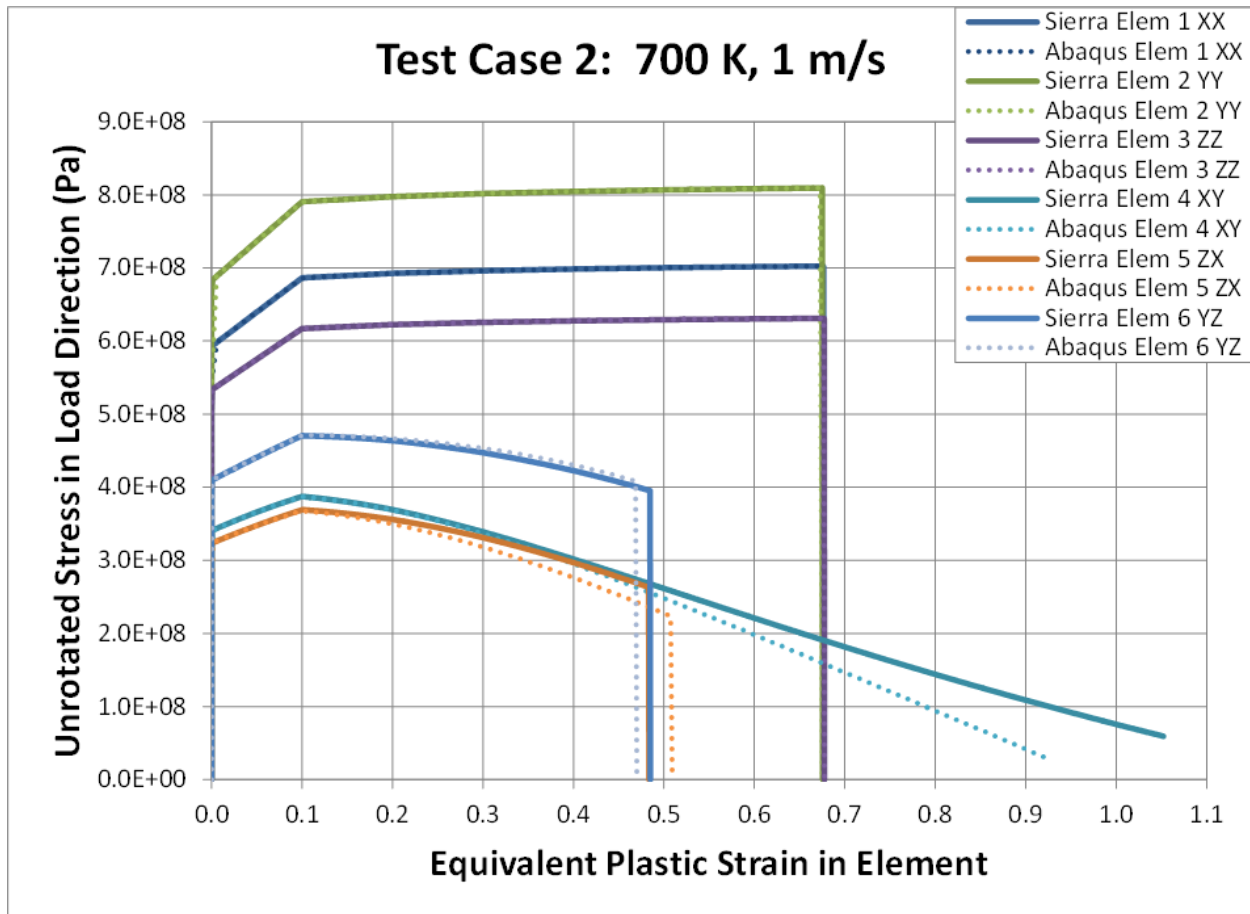
183

Figure 17-6. Comparison of results from test case 2.

are non-trivial differences at larger strains. While every attempt was made to make the models in Sierra using hydra plasticity and the models Abaqus/Explicit using the generalized plasticity model to be the same there may be some differences in non-material model algorithms or options that would result in discrepancies.

While the above tests are comprehensive, additional verification may benefit the model. Additional verification should focus on the hydra plasticity model's ability to recovering the data input by the hardening and failure functions under simple loading conditions.
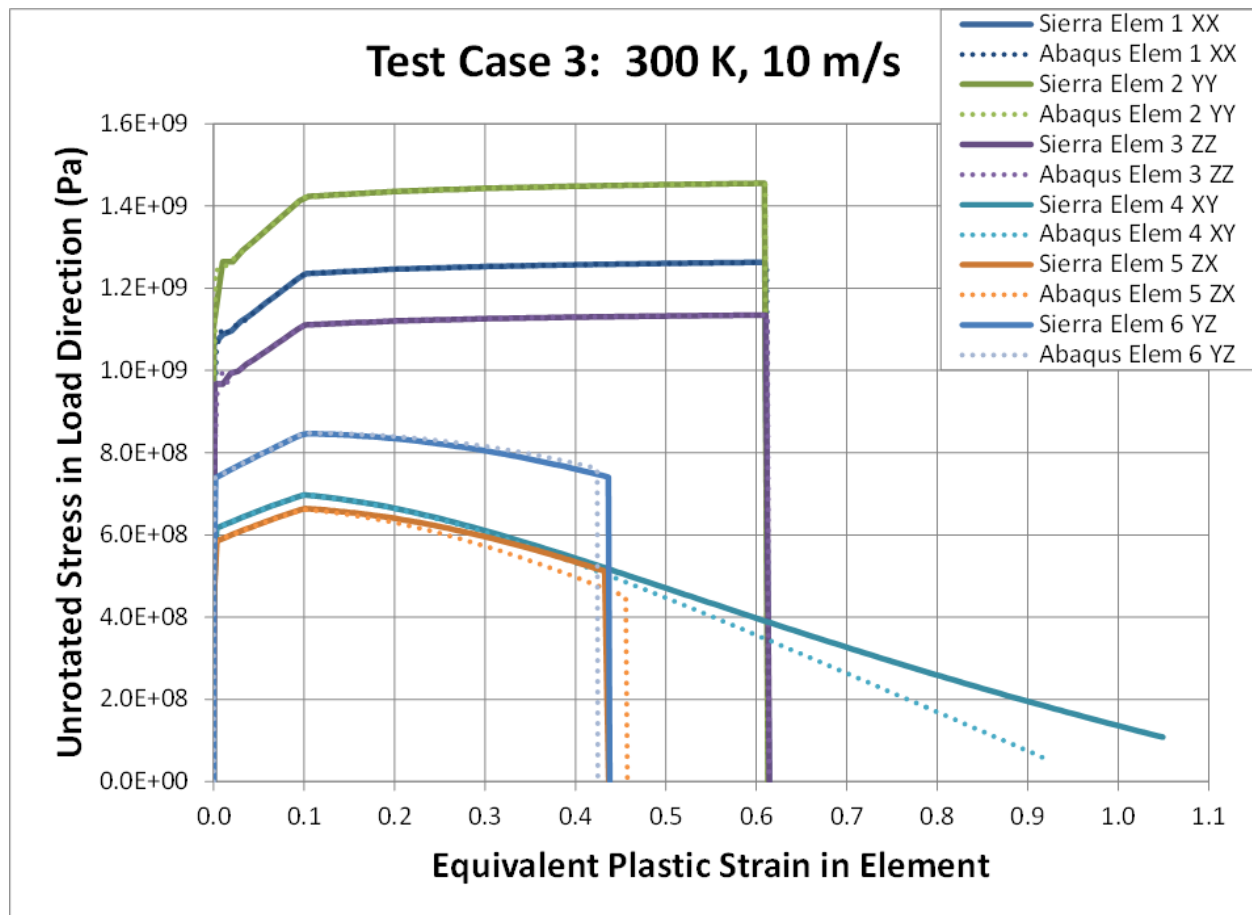
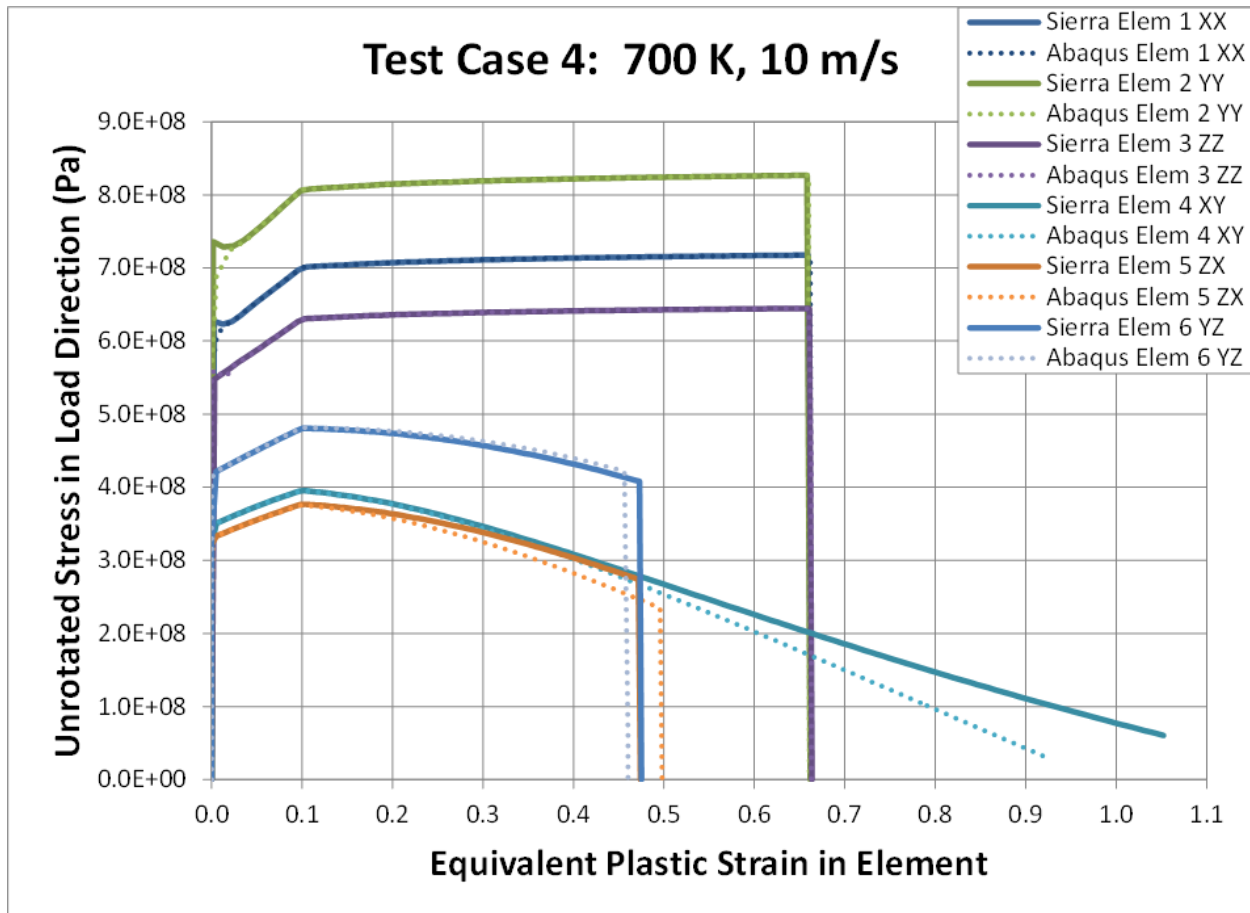**Figure 17-7. Comparison of results from test case 3.**

Figure 17-8. Comparison of results from test case 4.
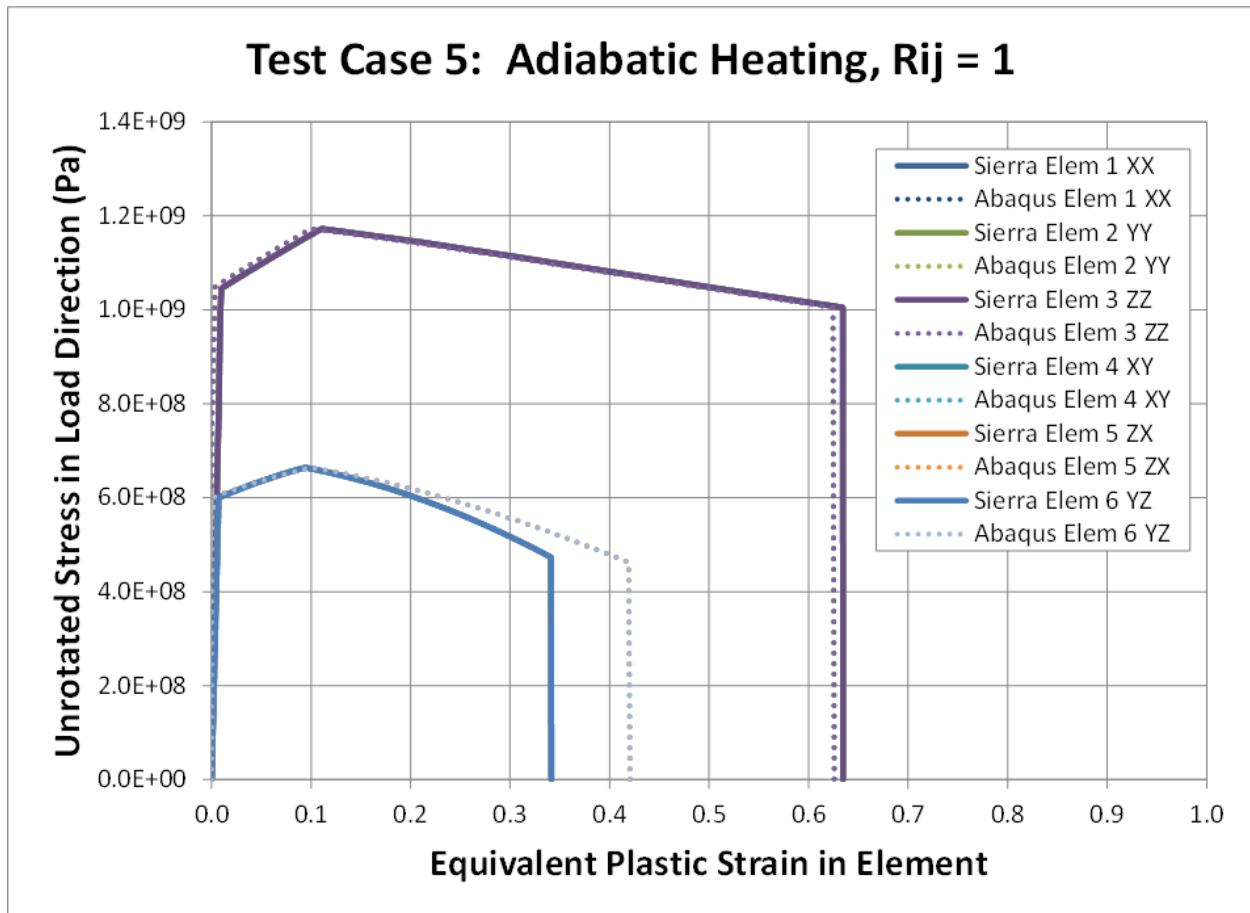
Figure 17-9. Comparison of results from test case 5.

17.16.   Honeycomb Model

```
BEGIN PARAMETERS FOR MODEL HONEYCOMB
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Orthotropic response
  #
  MODULUS_TTTT = <real> E⁰_TTTT
  MODULUS_LLLL = <real> E⁰_LLLL
  MODULUS_WWWW = <real> E⁰_WWWW
  MODULUS_TTLL = <real> E⁰_TTLL
  MODULUS_TTWW = <real> E⁰_TTWW
  MODULUS_LLWW = <real> E⁰_LLWW
  MODULUS_TLTL = <real> E⁰_TLTL
  MODULUS_LWLW = <real> E⁰_LWLW
  MODULUS_WTWT = <real> E⁰_WTWT
  #
  # Material orientation
  #
  TX = <real> t_x
  TY = <real> t_y
  TZ = <real> t_z
  LX = <real> l_x
  LY = <real> l_y
  LZ = <real> l_z
  #
  # Yield behavior
  #
  YIELD_STRESS = <real> σ_y
  A1           = <real> A₁
  B1           = <real> B₁
  C1           = <real> C₁
  A2           = <real> A₂
  B2           = <real> B₂
  C2           = <real> C₂
  A3           = <real> A₃
  B3           = <real> B₃
  C3           = <real> C₃
```

```
    TS  = <real> T_s
    LS  = <real> L_s
    WS  = <real> W_s
    TLS = <real> TL_s
    LWS = <real> LW_s
    WTS = <real> WT_s

    ESTL = <real>
    ESTW = <real>
    ESLW = <real>
    ESLT = <real>
    ESWT = <real>
    ESWL = <real>

    MODULUS_FUNCTION = <string>
    RATE_FUNCTION    = <string>
    T_FUNCTION       = <string>
    L_FUNCTION       = <string>
    W_FUNCTION       = <string>
    TL_FUNCTION      = <string>
    LW_FUNCTION      = <string>
    WT_FUNCTION      = <string>
    TTP_FUNCTION     = <string>
    LLP_FUNCTION     = <string>
    WWP_FUNCTION     = <string>
    TLTLP_FUNCTION   = <string>
    LWLWP_FUNCTION   = <string>
    WTWTP_FUNCTION   = <string>
    TTLP_FUNCTION    = <string>
    TTWP_FUNCTION    = <string>
  END [PARAMETERS FOR MODEL HONEYCOMB]
```

The honeycomb constitutive model is used to model the energy absorbing capabilities of aluminum honeycomb. There are three orthogonal material directions for the model: T, L, and W. The t-direction is generally considered as the "strong" direction, the W-direction is the "weak" direction, and the L-direction has an intermediate strength. This convention, however, does not necessarily need to be followed when defining material inputs.

$$
\begin{Bmatrix} \dot{\sigma}_{TT} \\ \dot{\sigma}_{LL} \\ \dot{\sigma}_{WW} \\ \dot{\sigma}_{TL} \\ \dot{\sigma}_{LW} \\ \dot{\sigma}_{WT} \end{Bmatrix} = \begin{bmatrix} E_{TTTT} & E_{TTLL} & E_{TTWW} & 0 & 0 & 0 \\ E_{TTLL} & E_{LLLL} & E_{LLWW} & 0 & 0 & 0 \\ E_{TTWW} & E_{LLWW} & E_{WWWW} & 0 & 0 & 0 \\ 0 & 0 & 0 & E_{TLTL} & 0 & 0 \\ 0 & 0 & 0 & 0 & E_{LWLW} & 0 \\ 0 & 0 & 0 & 0 & 0 & E_{WTWT} \end{bmatrix} \begin{Bmatrix} \dot{d}_{TT} \\ \dot{d}_{LL} \\ \dot{d}_{WW} \\ \dot{d}_{TL} \\ \dot{d}_{LW} \\ \dot{d}_{WT} \end{Bmatrix} \tag{17.107}
$$

Output variables available for this model are listed in Table 17-11.

Table 17-11. State Variables for HONEYCOMB Model

Index	Name	Variable Description
1	CRUSH	minimum volume ratio
2	EQDOT	effective strain rate
3	RMULT	rate multiplier
5	ITER	iterations
6	EVOL	volumetric strain

17.17. Viscoplastic Foam

```
BEGIN PARAMETERS FOR MODEL VISCOPLASTIC_FOAM
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ

  FLOW RATE      = <real> h
  POWER EXPONENT = <real> n
  BETA           = <real> β₀  (1.0)
  PHI            = <real> φ₀

  SHEAR STRENGTH  = <real> A₀  (1.0)
  SHEAR HARDENING = <real> A₁  (0.0)
  SHEAR EXPONENT  = <real> A₂  (1.0)
  HYDRO STRENGTH  = <real> B₀  (1.0)
  HYDRO HARDENING = <real> B₁  (0.0)
  HYDRO EXPONENT  = <real> B₂  (1.0)

  YOUNGS FUNCTION    = <string> hₑ(θ)
  POISSONS FUNCTION  = <string> hᵥ(θ)
  SS FUNCTION        = <string> h_{A₀}(θ)
  SH FUNCTION        = <string> h_{A₁}(θ)
  HS FUNCTION        = <string> h_{B₀}(θ)
  HH FUNCTION        = <string> h_{B₁}(θ)
  RATE FUNCTION      = <string> hₕ(θ)
  EXPONENT FUNCTION  = <string> hₙ(θ)
  STIFFNESS FUNCTION = <string> fₑ(φ)

  SHEAR HARDENING FUNCTION = <string> fₐ(φ)
  HYDRO HARDENING FUNCTION = <string> f_b(φ)
  BETA FUNCTION            = <string> f_β(φ)
END [PARAMETERS FOR MODEL VISCOPLASTIC_FOAM]
```

The viscoplastic foam model is used to model the rate (and temperature) dependent crushing of foams [19]. It is based on an additive split of the rate of deformation into an elastic and plastic portion

$$D_{ij} = D^{\text{e}}_{ij} + D^{\text{p}}_{ij}. \tag{17.108}$$

The stress in the material is due strictly to the elastic portion of the rate of deformation so that

$$\overset{\circ}{\sigma}_{ij} = \mathbb{C}_{ijkl} D_{kl}^{e}, \tag{17.109}$$

where $\mathbb{C}_{ijkl}$ are the components of the fourth-order, isotropic elasticity tensor. The stress rate is arbitrary, as long as it is objective. Two objective stress rates are commonly used: the Jaumann rate and the Green-McInnis rate. For problems with fixed principal axes of deformation, these two rates give the same answers. For problems where the principal axes of deformation rotate during deformation, the two rates can give different answers.

To describe the rate-dependent response, an over-stress-type yield function is used. Specifically, the rate-independent foam plasticity yield function

$$f = \frac{\bar{\sigma}^2}{a^2} + \frac{p^2}{b^2} - 1 \tag{17.110}$$

is rearranged such that,

$$f = \sigma^* - a, \tag{17.111}$$

where σ^* is the effective stress given by

$$\sigma^* = \sqrt{\bar{\sigma}^2 + \frac{a^2}{b^2} p^2}. \tag{17.112}$$

In (17.112), $\bar{\sigma}$ is the von Mises effective stress ($\bar{\sigma} = \sqrt{\frac{3}{2} s_{ij} s_{ij}}$) and p is the pressure resulting from a stress decomposition of the form,

$$s_{ij} = \sigma_{ij} - p\delta_{ij}. \tag{17.113}$$

Furthermore, a and b are state variables that are functions of the absolute temperature, θ, and maximum solid volume fraction, ϕ, and are defined as

$$a(\theta, \phi) = \left(A_0(\theta) + A_1(\theta) \phi^{A_2} \right) f_a(\phi) \tag{17.114}$$

$$b(\theta, \phi) = \left(B_0(\theta) + B_1(\theta) \phi^{B_2} \right) f_b(\phi). \tag{17.115}$$

In (17.114) and (17.115), the functional forms are chosen to provide flexibility in terms of representation. To this end, f_a and f_b default to constant values of one and the expression reduce to legacy analytical functions. This both provides for backward compatibility with previous user input decks while allowing for user defined definitions. Further, if $A_1(\theta) = 0$, $a(\theta, \phi) = A_0(\theta) f_a(\phi)$; a multiplicative decomposition of dependencies.

The temperature dependent material properties in the preceding relations are all defined as, $A_0(\theta) = A_0 h_{A_0}(\theta)$ where A_0 is the reference material parameter and $h_{A_0}(\theta)$ is the relative value as

a function of temperature. In addition to the a and b state variables, the Young's modulus and Poisson's ratio are also functions of the absolute temperature. The latter may be written as $v(\theta) = v h_v(\theta)$ while the former is also dependent on the maximum volume fraction of solid material and is given as $E(\theta, \phi) = E h_E(\theta) f_E(\phi)$.

The maximum volume fraction of solid material, ϕ, is given by

$$\phi = \max_{t>0} \tilde{\phi}(t) \tag{17.116}$$

where $\tilde{\phi}(t)$ is the current volume fraction of solid material and is defined as,

$$\tilde{\phi}(t) = \frac{\phi_0}{\exp\left(\varepsilon_v^{\mathrm{p}}\right)} \tag{17.117}$$

with ϕ_0 being the initial solid volume fraction and $\varepsilon_v^{\mathrm{p}}$ is

$$\varepsilon_v^{\mathrm{p}} = \int_0^t D_{kk}^{\mathrm{p}} dt. \tag{17.118}$$

During inelastic deformation ($f > 0$), the corresponding rate of plastic deformation is given in a Perzyna-type form as,

$$D_{ij}^{\mathrm{p}} = \begin{cases} \exp\left(h(\theta)\right) \left(\dfrac{\sigma^*}{a} - 1\right)^{n(\theta)} g_{ij} & \text{if} \quad f > 0 \\[4mm] 0 & \text{if} \quad f \leq 0 \end{cases} \tag{17.119}$$

where $h(\theta)$ and $n(\theta)$ are the flow rate and power exponent respectively. The inelastic flow direction, g_{ij}, is given as a linear combination of the associated (with respect to (17.110)), g_{ij}^{a}, and radial, g_{ij}^{r},

$$g_{ij} = \frac{(1-\beta) g_{ij}^{\mathrm{a}} + \beta g_{ij}^{\mathrm{r}}}{|(1-\beta) g_{kl}^{\mathrm{a}} + \beta g_{kl}^{\mathrm{r}}|} \tag{17.120}$$

The directions g_{ij}^{a} and g_{ij}^{r} are given by

$$g_{ij}^{\mathrm{a}} = \frac{\frac{\partial f}{\partial \sigma_{ij}}}{\left|\frac{\partial f}{\partial \sigma_{kl}}\right|} = \frac{\frac{3}{a^2} s_{ij} - \frac{2}{3b^2} p \delta_{ij}}{\left|\frac{3}{a^2} s_{ij} - \frac{2}{3b^2} p \delta_{ij}\right|}, \tag{17.121}$$

$$g_{ij}^{\mathrm{r}} = \frac{\sigma_{ij}}{|\sigma_{kl}|} = \frac{\sigma_{ij}}{\sqrt{\sigma_{kl} \sigma_{kl}}}, \tag{17.122}$$

respectively. In this model, the flow rule weight, β, may be specified as either a constant value or as a function of the maximum solid volume fraction ($\beta = \beta_0 f_\beta(\phi)$).

In the above command blocks:

- Since the model requires functions to describe the temperature dependence of the elastic modulus and Poisson's ratio, it is recommended that one inputs these properties at some reference temperature. However, any two of the elastic constants can be used for input. Consult the Sierra/SM 5.8 User's Guide Section 5.1.5 for more information on elastic constants input.

- The reference value for the flow rate, h, is defined with the FLOW RATE command line.

- The reference value of the over-stress exponent, n, is defined with the POWER EXPONENT command line.

- The user-defined scalar scaling between associated and radial flow, β, is defined with the BETA command line.

- The initial volume fraction of solid material, ϕ_0, is defined with the PHI command line.

- The reference value for the shear strength, A_0, is defined with the SHEAR STRENGTH command line.

- The reference value for the shear hardening modulus, A_1, is defined with the SHEAR HARDENING command line.

- The shear hardening exponent, A_2, is defined with the SHEAR EXPONENT command line.

- The reference value for the hydrostatic strength, B_0, is defined with the HYDRO STRENGTH command line.

- The reference value for the hydrostatic hardening modulus, B_1, is defined with the HYDRO HARDENING command line.

- The hydrostatic hardening exponent, B_2, is defined with the HYDRO EXPONENT command line.

- The user-defined and normalized function that gives the elastic modulus as a function of temperature, $h_E(\theta)$, is defined with the YOUNGS FUNCTION command line.

- The user-defined and normalized function that gives the Poisson's ratio as a function of temperature, $h_\nu(\theta)$, is defined with the POISSONS FUNCTION command line.

- The user-defined and normalized function that gives the shear strength as a function of temperature, $h_{A_0}(\theta)$, is defined with the SS FUNCTION command line.

- The user-defined and normalized function that gives the shear hardening modulus as a function of temperature, $h_{A_1}(\theta)$, is defined with the SH FUNCTION command line.

- The user-defined and normalized function that gives the hydrostatic strength as a function of temperature, $h_{B_0}(\theta)$, is defined with the HS FUNCTION command line.

- The user-defined and normalized function that gives the hydrostatic hardening modulus as a function of temperature, $h_{B_1}(\theta)$, is defined with the HH FUNCTION command line.

- The user-defined and normalized function that gives the flow rate as a function of temperature, $h_h(\theta)$, is defined with the RATE FUNCTION command line.

- The user-defined and normalized function that gives the over-stress exponent as a function

of temperature, $h_n(\theta)$, is defined with the EXPONENT FUNCTION command line.

- The user-defined and normalized function that gives the elastic modulus as a function of maximum solid volume fraction, $f_E(\phi)$, is defined with the STIFFNESS FUNCTION command line.

- The optional user-defined function that gives the shear strength as a function of the maximum solid volume fraction, $a(\phi)$, is defined with the SHEAR HARDENING FUNCTION command line. Note, if this function is defined the SHEAR STRENGTH, SHEAR HARDENING, and SHEAR EXPONENT values should not be specified.

- The optional user-defended function that gives the hydrostatic strength as a function of the maximum solid volume fraction, $b(\phi)$, is defined with the HYDRO HARDENING FUNCTION command line. Note, if this function is defined the HYDRO STRENGTH, HYDRO HARDENING, and HYDRO EXPONENT values should not be specified.

- The optional user-defined function that gives the scaling between associated and radial flow as a function of maximum solid volume fraction, $\beta(\phi)$, is defined with the BETA FUNCTION command line. Note, if this function is defined the BETA value should not be specified.

Output variables available for this model are listed in Table 17-12.

Table 17-12. State Variables for VISCOPLASTIC FOAM Model 17.17

Name	Description
ITER	number of sub-increments
EPVOL	inelastic volumetric strain, $\varepsilon_v^{\mathrm{p}}$
EDOT	effective inelastic strain rate, $\dot{\bar{\varepsilon}}^p$
PHI	maximum volume fraction of solid material, ϕ
FA	shear strength, a
FB	hydrostatic strength, b
STIF	elastic stiffness as a function of ϕ

17.18.   Foam Damage

```
BEGIN PARAMETERS FOR MODEL FOAM_DAMAGE
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Yield behavior
  #
  PHI                = <real> φ₀
  FLOW RATE          = <real> h
  POWER EXPONENT     = <real> n
  TENSILE STRENGTH   = <real> c
  ADAM               = <real> a_dam
  BDAM               = <real> b_dam
  #
  # Functions
  #
  YOUNGS FUNCTION            = <string> h_E(θ)
  POISSONS FUNCTION          = <string> h_ν(θ)
  RATE FUNCTION              = <string> h_h(θ)
  EXPONENT FUNCTION          = <string> h_n(θ)
  SHEAR HARDENING FUNCTION   = <string> a(φ)
  HYDRO HARDENING FUNCTION   = <string> b(φ)
  BETA FUNCTION              = <string> β(φ)
  YOUNGS PHI FUNCTION        = <string> f_E(φ)
  POISSONS PHI FUNCTION      = <string> f_ν(φ)
  DAMAGE FUNCTION            = <string> w(ε_dam)
END [PARAMETERS FOR FOAM_DAMAGE]
```

The foam damage model was developed at Sandia National Laboratories to model the behavior of rigid polyurethane foams under a variety of loading conditions [20]. For instance, temperature, rate, and tension-compression dependencies are all built into this model. This model, leverages previous efforts and experience with other foam models. Consult the Sierra/SM 5.8 User's Guide chapter on Material Models for additional details. Like those past efforts, this model utilizes an additive decomposition of the strain rates into elastic and inelastic parts,

$$D_{ij} = D_{ij}^{e} + D_{ij}^{in}. \tag{17.123}$$

It is also assumed that the elastic response is linear and isotropic such that the stress rate for

isothermal conditions is given by the following equation

$$\overset{\circ}{\sigma}_{ij} = \mathbb{C}_{ijkl}D_{kl}^{\mathrm{e}} = \mathbb{C}_{ijkl}\left(D_{kl} - D_{kl}^{\mathrm{in}}\right), \tag{17.124}$$

with $\mathbb{C}_{ijkl}$ being the fourth-order, isotropic elasticity tensor. The specific stress rate considered is arbitrary as long as it is object. Two common rates satisfying that constraint are the Jaumann and Green-McInnis rates.

The initial yield surface is assumed to be an ellipsoid about the hydrostat and is described by the function

$$f = \frac{\bar{\sigma}^2}{a^2} + \frac{p^2}{b^2} - 1 = 0, \tag{17.125}$$

where a and b are state variables that define the current deviatoric and volumetric strengths, respectively, of the foam. The von Mises effective stress, $\bar{\sigma}$ is a scalar measure of the deviatoric stress given by

$$\bar{\sigma} = \sqrt{\frac{3}{2}s_{ij}s_{ij}}, \tag{17.126}$$

while p is the pressure, or mean stress, and is defined as

$$p = \frac{1}{3}\sigma_{kk}, \tag{17.127}$$

with σ_{ij} and s_{ij} being the components of the Cauchy and deviatoric stress. This latter tensor may be written as,

$$s_{ij} = \sigma_{ij} - p\delta_{ij}, \tag{17.128}$$

where δ_{ij} are the components of the identity tensor - $\delta_{ij} = 1$ if $i = j$, $\delta_{ij} = 0$ if $i \neq j$.

For this model, the yield function (17.125) is re-written as

$$f = \sigma^* - a = 0 \tag{17.129}$$

with the effective stress, σ^*, being a function of the von Mises effective stress, $\bar{\sigma}$, and the pressure, p, as follows

$$\sigma^* = \sqrt{\bar{\sigma}^2 + \frac{a^2}{b^2}p^2}. \tag{17.130}$$

Next, using a Perzyna-type formulation, the following expression for the inelastic strain rate, D_{ij}^{in}, is developed

$$
D_{ij}^{\text{in}} =
\begin{cases}
\dot{\bar{\varepsilon}}^p g_{ij} = e^h \left(\dfrac{\sigma^*}{a} - 1 \right)^n g_{ij} & \text{if } \dfrac{\sigma^*}{a} - 1 > 0 \\
0 & \text{if } \dfrac{\sigma^*}{a} - 1 \leq 0,
\end{cases}
\tag{17.131}
$$

where g_{ij} are the components of a symmetric, second-order tensor that defines the orientation of the inelastic flow. This type of model is sometimes referred to as an over-stress model because the inelastic rate is a function of the over-stress - the distance outside the yield surface. For associated flow, g_{ij} is simply normal to the yield surface and is given by

$$
g_{ij}^{\text{a}} = \frac{\dfrac{\partial f}{\partial \sigma_{ij}}}{\left| \dfrac{\partial f}{\partial \sigma_{kl}} \right|} = \frac{\dfrac{3}{a^2} s_{ij} + \dfrac{2}{3b^2} p \delta_{ij}}{\left| \dfrac{3}{a^2} s_{kl} + \dfrac{2}{3b^2} p \delta_{kl} \right|}.
\tag{17.132}
$$

When lower density foams are subjected to a simple load path like uniaxial compression, the inelastic flow direction at moderate strains appears nearly uniaxial. In other words, the flow direction is given by the normalized stress tensor as follows

$$
g_{ij}^{\text{r}} = \frac{\sigma_{ij}}{|\sigma_{kl}|}.
\tag{17.133}
$$

This type of flow is called radial flow. The foam damage model has another parameter, β, which allows for the flow direction to be prescribed as a linear combination of associated and radial flow such that,

$$
g_{ij} = \frac{(1-\beta) g_{ij}^{\text{a}} + \beta g_{ij}^{\text{r}}}{\left| (1-\beta) g_{kl}^{\text{a}} + \beta g_{kl}^{\text{r}} \right|}.
\tag{17.134}
$$

Rigid polyurethane foams have little ductility when they are subjected to tensile stress. For this loading case, the materials behave more like brittle materials and even for uniaxial compression the foams often show cracking at large strains.

The damage surfaces for the foam damage model are simply three orthogonal planes with the normals given by the positive principal stress axes. The damage surfaces are given by the following equation

$$
f_{\text{dam}}^i = \hat{\sigma}^i - c(1-w), \quad ; \quad i = 1,2,3
\tag{17.135}
$$

198

where $\hat{\sigma}^i$ is a principal stress, c is the initial tensile strength which is a material parameter, and w is a scalar measure of the damage. As damage occurs, the damage surface will collapse toward the origin and the foam will lose tensile strength. The foam will, however, still have compressive strength.

Damage is taken to be a positive, monotonically increasing function of the damage strain, ε_{dam}, and the damage strain is a function of the maximum principal strain, ε_{max}, and the plastic volume strain, ε_v^p, such that

$$w = w(\varepsilon_{\text{dam}}) \quad ; \quad \varepsilon_{\text{dam}} = a_{\text{dam}}\varepsilon_{\text{max}} + b_{\text{dam}}\varepsilon_v^p, \tag{17.136}$$

with the material parameters a_{dam} and b_{dam} controlling the rate at which damage is generated in tension and compression, respectively. The model does not allow healing, so the damage never decreases even if the damage strain decreases.

To fully capture temperature, strain rate, and lock-up effects, several material parameters are defined as functions of temperature, θ, and/or some measure of the amount of compaction, e.g. the maximum volume fraction of the solid material obtained during any prior loading, ϕ. For instance,

$$E(\theta,\phi) = E\,h_E(\theta)\,f_E(\phi),$$

$$\tag{17.137}$$

$$\nu(\theta,\phi) = \nu\,h_\nu(\theta)\,f_\nu(\phi),$$

and the natural logarithm of the reference flow rate, h, and the power law exponent, n are also functions of temperature

$$h(\theta) = h\,h_h(\theta)$$

$$\tag{17.138}$$

$$n(\theta) = n\,h_n(\theta).$$

The current deviatoric and volumetric strengths are hardening functions of the maximum volume fraction of the solid material obtained during any prior loading, ϕ, as is the parameter that defines the fraction of associated and radial flow, β. Therefore,

$$a = a(\phi) \quad ; \quad b = b(\phi)$$

$$\tag{17.139}$$

$$\beta = \beta(\phi).$$

Through the loading cycle, the maximum volume fraction of solid material is written as,

$$\phi = \max_{t>0} \tilde{\phi}(t) \qquad (17.140)$$

where $\tilde{\phi}(t)$ is the current volume faction of solid material defined as

$$\tilde{\phi}(t) = \frac{\phi_0}{\exp\left(\varepsilon_v^{\mathrm{p}}\right)}, \qquad (17.141)$$

with ϕ_0 and $\varepsilon_v^{\mathrm{p}}$ being the initial solid volume fraction and plastic volumetric strain, respectively.

The foam damage model, as presented, provides a phenomenological model with enough flexibility to model the observed deformation and failure of rigid polyurethane foams.

- Consult the Sierra/SM 5.8 User's Guide Section 5.1.5 for more information on elastic constants input.

Output variables available for this model are listed in Table 17-13. For information about the foam damage model, consult [20].

Table 17-13. State Variables for FOAM DAMAGE Model

Name	Variable Description
ITER	number of sub-increments taken in subroutine
EPVOL	plastic volume strain
PHI	maximum volume fraction of solid material
EQPS	equivalent plastic strain
FA	shear strength - a
FB	hydrostatic strength - b
DAMAGE	damage
EMAX	maximum tensile strain
PWORK	plastic work rate

17.19. Thermo EP Power Model

Output variables available for this model are listed in Table 17-14.

Table 17-14. State Variables for THERMO EP POWER Model

Index	Name	Variable Description
1	EQPS	equivalent plastic strain
2	RADIUS	radius of yield surface
3	BACK_STRESS_XX	back stress - xx component
4	BACK_STRESS_YY	back stress - yy component
5	BACK_STRESS_ZZ	back stress - zz component
6	BACK_STRESS_XY	back stress - xy component
7	BACK_STRESS_YZ	back stress - yz component
8	BACK_STRESS_ZX	back stress - zx component

17.20.   Thermo EP Power Weld Model

Output variables available for this model are listed in Table 17-15.

Table 17-15. State Variables for THERMO EP POWER WELD Model

Index	Name	Variable Description
	EQPS	equivalent plastic strain
	RADIUS	radius of yield surface
	BACK_STRESS_XX	back stress - xx component
	BACK_STRESS_YY	back stress - yy component
	BACK_STRESS_ZZ	back stress - zz component
	BACK_STRESS_XY	back stress - xy component
	BACK_STRESS_YZ	back stress - yz component
	BACK_STRESS_ZX	back stress - zx component
	WELD_FLAG	

17.21.   Linear Thermoviscoelastic

```
BEGIN PARAMETERS FOR MODEL LINEAR_THERMOVISCOELASTIC
  #
  SHEAR MODULUS  = <real> μ^g
  BULK MODULUS   = <real> K^g
  #
  BULK RUBBERY 0  = <real> K^∞
  SHEAR RUBBERY 0 = <real> μ^∞
  ALPHA RUBBERY 0 = <real> α^∞
  #
  BULK GLASSY 0  = <real> K^∞
  SHEAR GLASSY 0 = <real> μ^∞
  ALPHA GLASSY 0 = <real> α^∞
  #
  #  – Optional temperature dependence of thermoelastic constants
  #
  BULK RUBBERY TEMPERATURE DEPENDENCE  = <string> f_∞^K
  SHEAR RUBBERY TEMPERATURE DEPENDENCE = <string> f_∞^μ
  ALPHA RUBBERY TEMPERATURE DEPENDENCE = <string> f_∞^α
  #
  BULK GLASSY TEMPERATURE DEPENDENCE  = <string> f_g^K
  SHEAR GLASSY TEMPERATURE DEPENDENCE = <string> f_g^μ
  ALPHA GLASSY TEMPERATURE DEPENDENCE = <string> f_g^α
  #
  #  – Reference temperature only needed
  #     if using temperature dependent parameters
  #
  T0 = <real> θ_0
  #
  SHIFT FACTOR MODEL = WLF | ARRHENIUS | USER_DEFINED | WLF_LAG
  #
  #  – IF SHIFT FACTOR MODEL = WLF | WLF_LAG
  #
  WLF C1 = <real> C_1
  WLF C2 = <real> C_2
  REFERENCE TEMPERATURE    = <real> θ_REF
  #
  #  – IF SHIFT FACTOR MODEL = ARRHENIUS
  #
  NORM ACTIVATION ENERGY = <real> E_a/R
  REFERENCE TEMPERATURE  = <real> θ_REF
  #
  #  – IF SHIFT FACTOR MODEL = USER_DEFINED
  #
  SHIFT FACTOR FUNCTION = <string> a^{ud}
  #
```

203

```
NUM BULK PRONY TERMS = <int> n_bulk
#
BULK RELAX TIME 1 = <real> τ₁ᵛ
BULK RELAX TIME 2 = <real> τ₂ᵛ
     ...
BULK RELAX TIME 30 = <real> τ₃₀ᵛ
#
f1 1  = <real> w₁ᵛ
f1 2  = <real> w₂ᵛ
     ...
f1 30 = <real> w₃₀ᵛ
#
NUM SHEAR PRONY TERMS = <int> n_shear
#
SHEAR RELAX TIME 1 = <real> τ₁ˢ
SHEAR RELAX TIME 2 = <real> τ₂ˢ
     ...
SHEAR RELAX TIME 30 = <real> τ₃₀ˢ
#
f2 1  = <real> w₁ˢ
f2 2  = <real> w₂ˢ
     ...
f2 30 = <real> w₃₀ˢ
#
END [PARAMETERS FOR MODEL LINEAR_THERMOVISCOELASTIC]
```

In the command blocks that define the LINEAR_THERMOELASTIC model:

- See the Sierra/SM 5.8 User's Guide Section 5.1.5 for more information on elastic constants input.

- The bulk and shear moduli of the equilibrium (rubbery) phase are defined by the BULK RUBBERY 0 and SHEAR RUBBERY 0 command lines, respectively.

- The bulk and shear moduli of the glassy phase are defined by the BULK GLASSY 0 and SHEAR GLASSY 0 command lines, respectively.

- The *linear* coefficients of thermal expansion of the rubbery and glassy phases are given by the ALPHA RUBBERY 0 and ALPHA GLASSY 0 command lines, respectively. Note, the UNIVERSAL POLYMER model uses volumetric – not linear – coefficients of thermal expansion.

- The shift factor model is specified via the SHIFT FACTOR MODEL command line.

- For the WLF or WLF_LAG shift factor models, the C_1 coefficient, C_2 coefficient, and reference temperature, θ_{ref}, are defined via the WLF C1, WLF C2, and REFERENCE TEMPERATURE command lines.

- For the Arrhenius shift factor model, the normalized activation energy, E_a/R, and reference temperature, θ_{ref}, are specified via the NORM ACTIVATION ENERGY and

`REFERENCE TEMPERATURE` command lines.

- For the user-defined shift factor model, the function name is specified via the `SHIFT FACTOR FUNCTION` command line. Note, any Sierra scope function may be defined and should be specified as a function of temperature. It is also emphasized the actual shift factor – not the logarithm of it – should be directly defined. Care should be taken to ensure the function is defined and admissible over the expected temperature domain.

- The number of terms in the Prony series of the bulk, n_{bulk}, and shear, n_{shear}, are given by the `NUM BULK PRONY TERMS` and `NUM SHEAR PRONY SERIES` command lines. These two numbers do not need to be the same and must both be defined. The maximum permissible number is 30 although any lower number is permissible.

- The bulk relaxation spectrum is characterized by a series of times, τ_k^v, and weights, w_k^v. The times are defined via `BULK RELAX TIME 1` through `BULK RELAX TIME` k where k is the number of terms in the spectrum. The same times do not need to be used for the bulk and shear terms. Weights are defined via the `f1 1` through `f1` k command lines. k should be equal to n_{bulk}.

- The shear relaxation spectrum is characterized by a series of times, τ_k^s, and weights, w_k^s. The times are defined via `SHEAR RELAX TIME 1` through `SHEAR RELAX TIME` k where k is the number of terms in the spectrum. The same times do not need to be used for the bulk and shear terms. Weights are defined via the `f2 1` through `f2` k command lines. k should be equal to n_{shear}.

Output variables available for this model are listed in Table 17-16. Note, for each of the various hereditary integral individual components may be output by specifying a hereditary integral "..._X" where X is an integer term in the spectrum. For instance, the 7 term of the Prony series of the volume strain integral (J_7^1) is output by specifying `J1_7`.

Table 17-16. State Variables for **LINEAR_THERMOVISCOELASTIC** Model (Section 17.21)

Name	Description
a	shift factor, a
J1	volume strain hereditary integral, J^1
J2_XX	XX component of the deviatoric hereditary integral, J_{11}^2
J2_YY	YY component of the deviatoric hereditary integral, J_{22}^2
J2_ZZ	ZZ component of the deviatoric hereditary integral, J_{33}^2
J2_XY	XY component of the deviatoric hereditary integral, J_{12}^2
J2_YZ	YZ component of the deviatoric hereditary integral, J_{23}^2
J2_ZX	ZX component of the deviatoric hereditary integral, J_{31}^2
J3	thermal strain hereditary integral, J^3

17.22.   NLVE 3D Orthotropic Model

```
BEGIN PARAMETERS FOR MODEL NLVE_3D_ORTHOTROPIC
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real> E
  POISSONS RATIO = <real> ν
  SHEAR MODULUS  = <real> G
  BULK MODULUS   = <real> K
  LAMBDA         = <real> λ
  TWO MU         = <real> 2μ
  #
  # Material coordinates system definition
  #
  COORDINATE SYSTEM               = <string> coordinate_system_name
  DIRECTION FOR ROTATION          = <real> 1|2|3
  ALPHA                           = <real> α₁ (degrees)
  SECOND DIRECTION FOR ROTATION   = <real> 1|2|3
  SECOND ALPHA                    = <real> α₂ (degrees)
  #
  #
  #
  FICTITIOUS LOGA FUNCTION = <string>fict_loga_function_name
  FICTITIOUS LOGA SCALE FACTOR = <real>fict_loga_scale_factor
  #
  # In each of the five "PRONY" command lines and in
  # the RELAX TIME command line, the value of i can be from
  # 1 through 30
  #
  1PSI PRONY <integer>i = <real>psi1_i
  2PSI PRONY <integer>i = <real>psi2_i
  3PSI PRONY <integer>i = <real>psi3_i
  4PSI PRONY <integer>i = <real>psi4_i
  5PSI PRONY <integer>i = <real>psi5_i
  RELAX TIME <integer>i = <real>tau_i
  REFERENCE TEMP = <real>tref
  REFERENCE DENSITY = <real>rhoref
  WLF C1 = <real>wlf_c1
  WLF C2 = <real>wlf_c2
  B SHIFT CONSTANT = <real>b_shift
  SHIFT REF VALUE = <real>shift_ref
  WWBETA 1PSI = <real>wwb_1psi
  WWTAU 1PSI = <real>wwt_1psi
  WWBETA 2PSI = <real>wwb_2psi
  WWTAU 2PSI = <real>wwt_2psi
  WWBETA 3PSI = <real>wwb_3psi
```

```
WWTAU 3PSI = <real>wwt_3psi
WWBETA 4PSI = <real>wwb_4psi
WWTAU 4PSI = <real>wwt_4psi
WWBETA 5PSI = <real>wwb_5psi
WWTAU 5PSI = <real>wwt_5psi
DOUBLE INTEG FACTOR = <real>dble_int_fac
REF RUBBERY HCAPACITY = <real>hcapr
REF GLASSY HCAPACITY = <real>hcapg
GLASS TRANSITION TEM = <real>tg
REF GLASSY C11 = <real>c11g
REF RUBBERY C11 = <real>c11r
REF GLASSY C22 = <real>c22g
REF RUBBERY C22 = <real>c22r
REF GLASSY C33 = <real>c33g
REF RUBBERY C33 = <real>c33r
REF GLASSY C12 = <real>c12g
REF RUBBERY C12 = <real>c12r
REF GLASSY C13 = <real>c13g
REF RUBBERY C13 = <real>c13r
REF GLASSY C23 = <real>c23g
REF RUBBERY C23 = <real>c23r
REF GLASSY C44 = <real>c44g
REF RUBBERY C44 = <real>c44r
REF GLASSY C55 = <real>c55g
REF RUBBERY C55 = <real>c55r
REF GLASSY C66 = <real>c66g
REF RUBBERY C66 = <real>c66r
REF GLASSY CTE1 = <real>cte1g
REF RUBBERY CTE1 = <real>cte1r
REF GLASSY CTE2 = <real>cte2g
REF RUBBERY CTE2 = <real>cte2r
REF GLASSY CTE3 = <real>cte3g
REF RUBBERY CTE3 = <real>cte3r
LINEAR VISCO TEST = <real>lvt
T DERIV GLASSY C11 = <real>dc11gdT
T DERIV RUBBERY C11 = <real>dc11rdT
T DERIV GLASSY C22 = <real>dc22gdT
T DERIV RUBBERY C22 = <real>dc22rdT
T DERIV GLASSY C33 = <real>dc33gdT
T DERIV RUBBERY C33 = <real>dc33rdT
T DERIV GLASSY C12 = <real>dc12gdT
T DERIV RUBBERY C12 = <real>dc12rdT
T DERIV GLASSY C13 = <real>dc13gdT
T DERIV RUBBERY C13 = <real>dc13rdT
T DERIV GLASSY C23 = <real>dc23gdT
T DERIV RUBBERY C23 = <real>dc23rdT
T DERIV GLASSY C44 = <real>dc44gdT
```

```
T DERIV RUBBERY C44 = <real>dc44rdT
T DERIV GLASSY C55 = <real>dc55gdT
T DERIV RUBBERY C55 = <real>dc55rdT
T DERIV GLASSY C66 = <real>dc66gdT
T DERIV RUBBERY C66 = <real>dc66rdT
T DERIV GLASSY CTE1 = <real>dcte1gdT
T DERIV RUBBERY CTE1 = <real>dcte1rdT
T DERIV GLASSY CTE2 = <real>dcte2gdT
T DERIV RUBBERY CTE2 = <real>dcte2rdT
T DERIV GLASSY CTE3 = <real>dcte3gdT
T DERIV RUBBERY CTE3 = <real>dcte3rdT
T DERIV GLASSY HCAPACITY = <real>dhcapgdT
T DERIV RUBBERY HCAPACITY = <real>dhcaprdT
REF PSIC = <real>psic_ref
T DERIV PSIC = <real>dpsicdT
T 2DERIV PSIC = <real>d2psicdT2
PSI EQ 2T = <real>psitt
PSI EQ 3T = <real>psittt
PSI EQ 4T = <real>psitttt
PSI EQ XX 11 = <real>psiXX11
PSI EQ XX 22 = <real>psiXX22
PSI EQ XX 33 = <real>psiXX33
PSI EQ XX 12 = <real>psiXX12
PSI EQ XX 13 = <real>psiXX13
PSI EQ XX 23 = <real>psiXX23
PSI EQ XX 44 = <real>psiXX44
PSI EQ XX 55 = <real>psiXX55
PSI EQ XX 66 = <real>psiXX66
PSI EQ XXT 11 = <real>psiXXT11
PSI EQ XXT 22 = <real>psiXXT22
PSI EQ XXT 33 = <real>psiXXT33
PSI EQ XXT 12 = <real>psiXXT12
PSI EQ XXT 13 = <real>psiXXT13
PSI EQ XXT 23 = <real>psiXXT23
PSI EQ XXT 44 = <real>psiXXT44
PSI EQ XXT 55 = <real>psiXXT55
PSI EQ XXT 66 = <real>psiXXT66
PSI EQ XT 1 = <real>psiXT1
PSI EQ XT 2 = <real>psiXT2
PSI EQ XT 3 = <real>psiXT3
PSI EQ XTT 1 = <real>psiXTT1
PSI EQ XTT 2 = <real>psiXTT2
PSI EQ XTT 3 = <real>psiXTT3
REF PSIA 11 = <real>psiA11
REF PSIA 22 = <real>psiA22
REF PSIA 33 = <real>psiA33
REF PSIA 12 = <real>psiA12
```

```
       REF PSIA 13 = <real>psiA13
       REF PSIA 23 = <real>psiA23
       REF PSIA 44 = <real>psiA44
       REF PSIA 55 = <real>psiA55
       REF PSIA 66 = <real>psiA66
       T DERIV PSIA 11 = <real>dpsiA11dT
       T DERIV PSIA 22 = <real>dpsiA22dT
       T DERIV PSIA 33 = <real>dpsiA33dT
       T DERIV PSIA 12 = <real>dpsiA12dT
       T DERIV PSIA 13 = <real>dpsiA13dT
       T DERIV PSIA 23 = <real>dpsiA23dT
       T DERIV PSIA 44 = <real>dpsiA44dT
       T DERIV PSIA 55 = <real>dpsiA55dT
       T DERIV PSIA 66 = <real>dpsiA66dT
       REF PSIB 1     = <real> psiB1
       REF PSIB 2     = <real> psiB2
       REF PSIB 3     = <real> psiB3
       T DERIV PSIB 1 = <real> dpsiB1dT
       T DERIV PSIB 2 = <real> dpsiB2dT
       T DERIV PSIB 3 = <real> dpsiB3dT
       PSI POT TT     = <real> psipotTT
       PSI POT TTT    = <real> psipotTTT
       PSI POT TTTT   = <real> psipotTTTT
       PSI POT XT 1   = <real> psipotXT1
       PSI POT XT 2   = <real> psipotXT2
       PSI POT XT 3   = <real> psipotXT3
       PSI POT XTT 1  = <real> psipotXTT1
       PSI POT XTT 2  = <real> psipotXTT2
       PSI POT XTT 3  = <real> psipotXTT3
       PSI POT XXT 11 = <real> psipotXXT11
       PSI POT XXT 22 = <real> psipotXXT22
       PSI POT XXT 33 = <real> psipotXXT33
       PSI POT XXT 12 = <real> psipotXXT12
       PSI POT XXT 13 = <real> psipotXXT13
       PSI POT XXT 23 = <real> psipotXXT23
       PSI POT XXT 44 = <real> psipotXXT44
       PSI POT XXT 55 = <real> psipotXXT55
       PSI POT XXT 66 = <real> psipotXXT66
    END [PARAMETERS FOR MODEL NLVE_3D_ORTHOTROPIC]
```

The NLVE three-dimensional orthotropic model is a nonlinear viscoelastic orthotropic continuum model that describes the behavior of fiber-reinforced polymer-matrix composites. In addition to being able to model the linear elastic and linear viscoelastic behaviors of such composites, it also can capture both "weak" and "strong" nonlinear viscoelastic effects such as stress dependence of the creep compliance and viscoelastic yielding. This model can be used in both Presto and Adagio.

Because the NLVE model is still under active development and also because it has an extensive

list of command lines, we have not followed the typical approach in documenting this model.

17.23. Other Undocumented Material Models

For a listing of other material models that exist in Sierra/SM See Table 17-17. Support for use of these models is limited.

Table 17-17. Other Material Models Available, but Undocumented

Material Name	Author
CDM_EP	Shawn English
UNIVERSAL_CURING	Kevin Long
JOHNSON COOK DAMAGE	Bill Scherzinger
FROST_ASHBY_CREEP	Bill Scherzinger
ELASTIC_PLASTIC_FAIL	Bill Scherzinger
ELASTIC_ORTHOTROPIC_FAIL	Shawn English
HAIL_ICE	Bill Scherzinger
ELASTO_VISCOPLASTIC	Arthur Brown
ELASTIC_UQ_SHELL	Mark Merewether
MLEP_WILKINS_FAIL	Mike Neilsen
UCP_FAIL	Mike Neilsen
SOLDER	
SOLDER_DAMAGE	
COULOMBMIXMODE	Shawn English
EVG	Jake Ostien
SPECTACULAR	Kevin Long
HILL_PLASTICITY_DAMAGE	Jake Ostien
CRYSTAL_PLASTIC	David Littlewood
CRYSTAL_PLASTICITY	
LOCAL_CRYSTAL_PLASTICITY	

References

[1] S.A. English. A 3D orthotropic strain-rate dependent elastic damage material model. Technical Report SAND2014-17469, Sandia National Laboratories, Albuquerque, NM, 2014. pdf.

[2] S.W. Attaway, R.V. Matallucci, S.W. Key, K.B. Morrill, L.J. Malvar, and J.E. Crawford. Enhancements to Pronto3D to predict structural response to blast. Technical Report SAND2000-1017, Sandia National Laboratories, Albuquerque, NM, 2000. pdf.

[3] ACI318-08: Building code requirements for structural concrete and commentary. Farmington Hills, MI. American Concrete Institute, 2008.

[4] R.M. Brannon, A.F. Fossum, and O.E. Strack. Kayenta: Theory and user's guide. Technical Report SAND2009-2282, Sandia National Laboratories, Albuquerque, NM, 2009.

[5] D. C. Lagoudas, editor. *Shape Memory Alloys: Modeling and Engineering Application*. Springer, New York, NY USA, 2008.

[6] K. Otsuka and X. Ren. Physical metallurgy of Ti-Ni-based shape memory alloys. *Progress in Materials Science*, 50:511–678, 2005.

[7] D. Lagoudas, D. Hartl, Y. Chemisky, L. Machado, and P. Popov. Constitutive model for the numerical analysis of phase transformation in polycrystalline shape memory alloys. *International Journal of Plasticity*, 32-33:155–183, 2012.

[8] D. J. Hartl, D. C. Lagoudas, F. T. Calkins, and J. H. Mabe. Use of a Ni60Ti shape memory alloy for active jet engine chevron application: I. thermomechanical characterization. *Smart Materials and Structures*, 19:015020, 2010.

[9] D. J. Hartl, J. T. Mooney, D. C. Lagoudas, F. T. Calkins, and J. H. Mabe. Use of a Ni60Ti shape memory alloy for active jet engine chevron application: II. experimentally validated numerical analysis. *Smart Materials and Structures*, 19:015021, 2010.

[10] G.R. Johnson and W.H. Cook. A constitutive model and data for metals subjected to large strains, high strain rates and high temperatures. In *Proc. 7th. Int. Symp. on Ballistics*, pages 541–547, The Hague, The Netherlands, 1983.

[11] G.R. Johnson and W.H. Cook. Fracture characteristics of three metals subjected to various strains, strain rates, temperatures and pressures. *Engineering Fracture Mechanics*, 21(1): 31–48, 1985.

[12] B.T. Lester and W.M. Scherzinger. Failure formulations in modular plasticity models. Memo to distribution SAND2019-9678R, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2019.

[13] B.T. Lester and W.M. Scherzinger. Modular growth and nucleation formulations in modular plasticity models. Memo to distribution SAND2020-7404R, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2020.

[14] B.T. Lester and W.M. Scherzinger. Adiabatic heating in modular plasticity models. Memo to distribution SAND2019-15003R, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2019.

[15] A. P. Karafillis and M. C. Boyce. A general anisotropic yield criterion using bounds and a transformation weighting tensor. *Journal of the Mechanics and Physics of Solids*, 41(12): 1859–1886, 1993.

[16] E. A. Olevsky. Theory of sintering: from discrete to continuum. *Materials Science and Engineering*, R23:41–100, 1998.

[17] J. G. Argüello, M. W. Reiterer, and K. G. Ewsuk. Verification, performance, validation, and modifications to the SOVS continuum constitutive model in a nonlinear large-deformation finite element code. *Journal of the American Ceramic Society*, 92(7):1442–1449, 2009.

[18] E. A. Olevsky, V. Tikare, T. J. Garino, and M. V. Braginsky. Simulation of sintering of layered structures. In *Proceedings of PM2000: World Congress on Powder Metallurgy*, Kyoto, Japan, 2002. Japan Society of Powder and Powder Metallurgy.

[19] M. K. Neilsen, W. Y. Lu, B. Olsson, and T. Hinnerichs. A viscoplastic constitutive model for polyurethane foams. In *Proceedings ASME 2006 International Mechanical Engineering Congress and Exposition*, Chicago, IL, 2006. ASME.

[20] M. K. Neilsen, W. Y. Lu, W. M. Scherzinger, T. D. Hinnerichs, and C. S. Lo. Unified creep plasticity damage (UCPD) model for rigid polyurethane foams. Technical Report SAND2015-4352, Sandia National Laboratories, Albuquerque, NM, 2015.

18. COHESIVE MATERIAL MODELS

This chapter describes the theory and usage of cohesive models in development. There are typically two different types of cohesive models – *intrinsic* and *extrinsic*. Intrinsic models are used for cohesive surfaces that are known a priori and are included in the model from the beginning. These models by definition produce zero traction for zero cohesive separation and have a loading region before failure. Extrinsic models are used when cohesive surfaces are dynamically inserted based on some material criteria. These models typically are initialized to produce an equilibrium traction at zero separation based on the cohesive zone insertion criteria. Section 18.1 describes the intrinsic cohesive zone models in development, whereas Section 18.2 describes the extrinsic models.

18.1. Intrinsic models

18.1.1. *Mixed-mode Dependent Toughness*

The MDGc CZM (Mixed-mode Dependent Toughness Cohesive Zone Mode) has two elements. Mode I energy dissipation is defined by a trapezoidal traction-separation relationship that depends only on normal separation. Mode II (III) dissipation is generated by shear yielding that depends only on the tangential separation components. A perfect plasticity-like formulation is used to define shear yielding by relating effective shear traction to effective slip rate. Shear yielding occurs within the region where Mode I separation (softening) occurs and can also occur ahead of that region. The MDGc CZM was developed to model crack propagation along an epoxy/solid interface when there is small-scale crack-tip yielding and when the epoxy and solid materials can be idealized as linear elastic. Nevertheless, this model might be applicable to other types of interfaces, but the user needs to use care in doing so. The MDGc CZM is described in detail in reference [1]. Note that the current implementation of the MDGc CZM differs slightly from that described in reference [1] in that shear unloading occurs after Mode I separation is complete (i.e. the normal traction has dropped to zero). In the initial implementation described in reference [1], shear unloading commenced as soon as Mode I softening initiated. A clear preference for either option is not obvious and the current choice generates a smoother solution.

```
BEGIN PARAMETERS FOR MODEL MDGc
  PEAK NORMAL TRACTION = <real>
  NORMAL LENGTH SCALE = <real>
  TANGENTIAL LENGTH SCALE = <real>
  LAMBDA_1 = <real>
  LAMBDA_2 = <real>
  PEAK SHEAR TRACTION = <real>
```

```
    LAMBDA_3 = <real>
    PENETRATION PENALTY = <real>
    UNLOAD TYPE = ELASTIC
 END [PARAMETERS FOR MODEL MDGc]
```

In the above command blocks:

- The maximum normal traction is specified by the PEAK NORMAL TRACTION command.
- The normal separation at which the normal traction falls to zero is prescribed by the NORMAL LENGTH SCALE command.
- The effective tangential separation over which plastic yield occurs before the interface fails in shear is prescribed by the TANGENTIAL LENGTH SCALE command. This should be large compared to NORMAL LENGTH SCALE. A recommended value is 100.0.
- LAMBDA_1 indicates the normalized separation at which the normal traction response flattens with an additional increase in normal separation. The initial Mode I loading slope K equals the PEAK NORMAL TRACTION/(LAMBDA_1×NORMAL LENGTH SCALE).
- LAMBDA_2 indicates the normalized separation at which the normal traction begins to decrease with additional increase in normal separation. Setting LAMBDA_1=LAMBDA_2 generates a triangular traction-separation relationship.
- The maximum shear traction is specified through the PEAK SHEAR TRACTION command.
- LAMBDA_3 controls the rapidity with which the shear is released. The shear unloading slope, K_u, equals the negative of the initial Mode I loading slope, K, times the ratio of LAMBDA_1/LAMBDA_3. One reasonable choice is LAMBDA_3=LAMBDA_1.
- The PENETRATION PENALTY parameter multiplies the Mode I loading slope, K, to provide an artificially increased penetration stiffness to help prevent interpenetration of cohesive surfaces when crack closure occurs. It is recommended that this parameter be set to zero (no penetration stiffness) and that Sierra/SM contact surfaces be used to prevent interpenetration.
- The only currently supported option for UNLOAD TYPE is ELASTIC.

The state variables for this model are listed in Table 18-1.

Table 18-1. State Variables for MDGc CZM (Section 18.1.1)

Name	Variable Description
LAMBDA_MAX	Maximum lambda the model has experienced (lambda equals the normal separation divided by the NORMAL LENGTH SCALE)
TRACTION AT LAMBDA MAX	Traction at LAMBDA_MAX

18.2. Extrinsic models

18.2.1. Tvergaard Hutchinson

This model is an extension of the trapezoidal traction-separation model proposed by Tvergaard and Hutchinson [2] generalized to multiple dimensions. The generalization is performed by appropriately scaling the normal and tangential components of the traction and separation into the 1D model depicted in Figure 18-1. In Figure 18-1, λ_c is the normalized final cohesive opening in the effective space, λ_1 is the length of the initial loading branch of the model, λ_2 is the separation length that begins the failure branch of the model, and $\hat{\sigma}$ is the maximum effective traction of the cohesive zone. These parameters have the following restrictions on their values:

$$0 \le \lambda_1 \le \lambda_2 \le \lambda_c = 1, \quad \hat{\sigma} > 0.$$

Finally, as shown in Figure 18-1, for $\lambda > \lambda_1$ unloading may be assumed towards the origin.

Assuming a loading condition ($\lambda > 0$, $\dot{\lambda} > 0$), the slope of the effective traction-separation model is evaluated as follows

$$\hat{t}' = \begin{cases} \hat{\sigma}/\lambda_1, & \lambda \in [0, \lambda_1) \\ \hat{\sigma}/\lambda, & \lambda \in [\lambda_1, \lambda_2) \\ \hat{\sigma}(1-\lambda)/(\lambda(1-\lambda_2)), & \lambda \in [\lambda_2, \lambda_c) \\ 0, & \lambda \ge \lambda_c \end{cases}$$

and the effective traction is computed as $\hat{t} = \hat{t}' \lambda$.

The effective traction-separation model is extended to 3D by defining the following additional values:

- The normal failure separation, δ_{cn}
- The tangential failure separation, δ_{ct}
- The ratio of failure separations, $r = \delta_{cn}/\delta_{ct}$
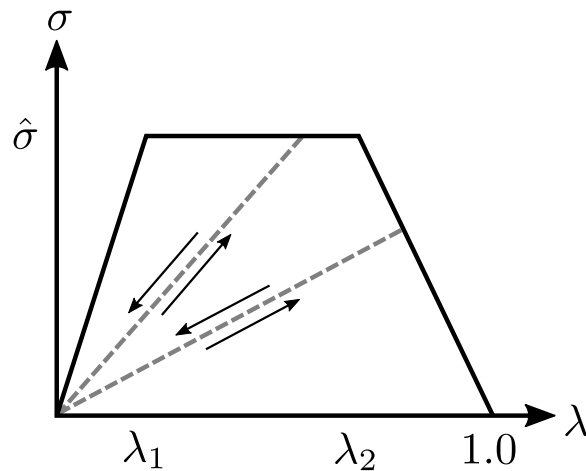
Figure 18-1. The effective traction-separation model following [2].

- The normalized normal separation, $\lambda_n = u_n / \delta_{cn}$
- The normalized tangential separations, $\lambda_t^i = u_t^i / \delta_{ct}$, $i = 1, 2$.
- The effective separation, $\lambda = \sqrt{\lambda_n^2 + (\lambda_t^1)^2 + (\lambda_t^2)^2}$

Then, the traction is computed as

$$
\begin{aligned}
t_t^1 &= \hat{t}' \lambda_t^1 r, \\
t_t^2 &= \hat{t}' \lambda_t^2 r, \\
t_n &= \hat{t}' \lambda_n.
\end{aligned}
$$

The model is extended to the extrinsic behavior by computing an effective opening $\tilde{\boldsymbol{\lambda}}$ that recovers the initialization traction. There are two modes of initialization: 1) where the initial effective traction is below the peak traction specified in the input file, and 2) where the initial effective traction exceeds the peak traction in the input file. In the first case, the components of the effective opening ($\tilde{\boldsymbol{\lambda}}$) are computed on the hardening branch of the cohesive model. In the second case, the peak traction is reset to the initial effective traction and the components of the initial effective opening are computed using the condition $|\tilde{\boldsymbol{\lambda}}| = \lambda_1$. Evaluation of the extrinsic effective opening is given by the following:

$$
\begin{aligned}
\tilde{\sigma} &= \sqrt{(t_n r)^2 + (t_t^1)^2 + (t_t^2)^2}, \\
\hat{\sigma} &= \max(\hat{\sigma}, \tilde{\sigma}), \\
\tilde{\lambda}_t^i &= \frac{t_t^i \lambda_1}{\hat{\sigma} r}, \quad i = 1, 2, \\
\tilde{\lambda}_n &= \frac{t_n \lambda_n}{\hat{\sigma}}.
\end{aligned}
$$

After initialization, the model is evaluated using

$$
\lambda = \sqrt{(\lambda_n + \tilde{\lambda}_n)^2 + (\lambda_t^1 + \tilde{\lambda}_t^1)^2 + (\lambda_t^2 + \tilde{\lambda}_t^2)^2}.
$$

The model is specified in adagio by the following command block:

```
BEGIN PARAMETERS FOR MODEL TVERGAARD_HUTCHINSON
  INIT TRACTION METHOD = IGNORE|ADD|EXTRINSIC (IGNORE)
  LAMBDA_1 = <real>
  LAMBDA_2 = <real>
  NORMAL LENGTH SCALE = <real>
  TANGENTIAL LENGTH SCALE = <real>
  PEAK TRACTION = <real>
  PENETRATION STIFFNESS MULTIPLIER = <real>
  USE ELASTIC UNLOADING = NO|YES (YES)
END [PARAMETERS FOR MODEL TVERGAARD_HUTCHINSON]
```

The `INIT TRACTION METHOD = EXTRINSIC|ADD` command line relates only to the dynamic insertion of cohesive zone elements through element death or XFEM.

18.2.2. *Thouless Parmigiani*

This model is an extension of the Tvergaard Hutchinson effective traction-separation model described in Section 18.2.1, but the normal and tangential traction components are treated independently. The model is specified in adagio by the following command block:

```
BEGIN PARAMETERS FOR MODEL THOULESS_PARMIGIANI
  INIT TRACTION METHOD = IGNORE|ADD|EXTRINSIC (IGNORE)
  LAMBDA_1_N = <real>
  LAMBDA_1_T = <real>
  LAMBDA_2_N = <real>
  LAMBDA_2_T = <real>
  NORMAL LENGTH SCALE = <real>
  PEAK NORMAL TRACTION = <real>
  TANGENTIAL LENGTH SCALE = <real>
  PEAK TANGENTIAL TRACTION = <real>
  PENETRATION STIFFNESS MULTIPLIER = <real>
  USE ELASTIC UNLOADING = NO|YES (YES)
END [PARAMETERS FOR MODEL THOULESS_PARMIGIANI]
```

References

[1] E. D. Reedy and J. M. Emery. A simple cohesive zone model that generates a mode-mixity dependent toughness. *International Journal of Solids and Structures*, 51:3727–3734, 2014.

[2] Viggo Tvergaard and John W. Hutchinson. On the toughness of ductile adhesive joints. *Journal of the Mechanics and Physics of Solids*, 44(5):789–800, 1996.

19.   MULTICRITERIA REBALANCE

This chapter describes how to use the multicriteria rebalance capability in Sierra/SM.

When running a typical Sierra/SM analysis, there are many capabilities that have a much higher computational cost than the other capabilities used in the simulation. An example of such capabilities is contact. Because contact typically does not occur on every element in the finite element model, the mesh can be decomposed (rebalanced) to split the contact work up across as many processors as possible to run most efficiently. This rebalance can occur at a user defined interval to account for contact patches coming in and out of contact with each other (consider a tire rolling on the ground), or it can be done automatically. Multicriteria rebalance also takes into account more expensive element formulations.

To activate multicriteria rebalance, use the following command:

```
BEGIN REBALANCE
     INITIAL REBALANCE = ON
     PERIODIC REBALANCE = AUTO
     LOAD RATIO THRESHOLD = 1.25
     REBALANCE STRATEGY = MULTICRITERIA
END
```

Warning: `REBALANCE STRATEGY = MULTICRITERIA` is still an experimental capability and should be used with caution.

Known Issue: `REBALANCE STRATEGY = MULTICRITERIA` does not currently work with restart.

20.   OTHER IN-DEVELOPMENT CAPABILITIES

This chapter describes other miscellaneous capabilities that are still in development or have limited testing.

20.1.   Element Birth (Element Activation)

```
BEGIN ELEMENT BIRTH <string>birth_name
  BLOCK = <string list>block_names
  BIRTH START TIME = <real>time
  CRITERION IS ELEMENT VALUE OF
    <string>var_name <|<=|=|>=|> <real>tolerance
    <integer>num_intg INTEGRATION POINTS REMAIN]
  CRITERION IS AVG|MAX|MIN NODAL VALUE OF
    <string>var_name <|<=|=|>=|> <real>tolerance
  CRITERION IS GLOBAL VALUE OF
    <string>var_name <|<=|=|>=|> <real>tolerance
END
```

A limited element birth/activation capability is provided for the target use cases of additive manufacturing and welds.

Elements are birthed upon an element variable, nodal variable, or global variable criterion. See Element Death section of the User's Manual for tips on properly setting up death/birth criteria based on a registered variable.

This capability is currently implemented for UG Hex8 elements with isotropic hypoelastic materials only.

Element birth works with thermal strains. Inactive elements do not accumulate thermal strains.

Element birth will error if a node is shared between an element birth block and a block involving: contact, force external boundary conditions, and kinetic boundary conditions.

20.2.   Initial Particle Conversion

```
BEGIN CONVERSION TO PARTICLES AT INITIALIZATION <string>name
  BLOCK   = <string list>block_names
  ASSEMBLY   = <string list>assembly_names
  SECTION = <string>section_name
END
```

The initial particle conversion capability is provided to facilitate the creation of particle meshes for particle based methods—such as smooth particle hydrodynamics (SPH) or reproducing kernel particle method (RKPM)—from an initial mesh of solid elements (e.g., hexes).

At the beginning of the analysis the solid element blocks listed in `block_names`, or assemblies of solid element blocks listed in `assembly_names` are converted to spherical particles of the type defined in the particle section `section_name`. It is important to note that the particle section will thus supersede any section specified in the original solid element block definition (consult [1] section on Element Block Parameters).

Note that elements may also be converted to particles via element death (consult [1] section on Element Death); however, conversion at initialization should offer more robust creation of particle meshes that are (a) compatible with the original mesh boundary conditions and (b) amenable to the chosen particle formulation methodology.

20.3.   Shell Contact Lofting Factor

⚠️   **Warning:** The shell contact lofting factor only works with Dash contact.

```
BEGIN SHELL SECTION <string>shell_section_name
  # ... see the Elements chapter of [1]
  CONTACT LOFTING FACTOR = <real>contact_lofting_factor
END [SHELL SECTION <string>shell_section_name]
```

The CONTACT LOFTING FACTOR line command is available in the SHELL SECTION command block to set a lofting factor specifically for use in contact. This contact lofting factor is used in place of the kinematic lofting factor for creation of the shell lofted geometry in contact. If no contact lofting factor is set, the kinematic lofting factor is used for contact.

The contact lofting factor has no effect on the shell element kinematics, and the LOFTING FACTOR and CONTACT LOFTING FACTOR line commands may be used in combination to independently set the kinematic and contact lofting factors, respectively.

20.4.   Reaction Diffusion Solver

An experimental diffusion capability eventually intended for use in phase field fracture. Currently in early development and not recommended for use.

```
BEGIN REACTION DIFFUSION rxndiffname
  BLOCK = <string list>block_names
  ASSEMBLY = <string list>assembly_names
  INCLUDE ALL BLOCKS
  INITIAL VALUE = <real>value(1.0)
  SOURCE COEFFICIENT = <real>source_coeff(1.0)
  DIFFUSION COEFFICIENT = <real>diff_coeff(0.0)
  REACTION COEFFICIENT = <real<rxn_coeff(1.0)
  SOLVE AT INITIALIZATION = OFF|ON(OFF)
  SOLVE INCREMENT = <integer>solv_incr(1)
  SOLVE EXPLICIT = OFF|ON(OFF)
  GRADIENT CONFIGURATION = MODEL|CURRENT(MODEL)
  BEGIN PRESCRIBED FLUX
    SURFACE = <string>surf_name
    NODE SET|NODESET = <string>node_set_name
    FUNCTION = <string>func_name
  END
  BEGIN PRESCRIBED FIELD
    SURFACE = <string>surf_name
    NODE SET|NODESET = <string>node_set_name
    FUNCTION = <string>func_name
  END
END

BEGIN GRADIENT DAMAGE gradDam
  BLOCK = <string>block_names
  ASSEMBLY = <string list>assembly_names
  INCLUDE ALL BLOCKS
  FORMULATION = LORENTZ|MIEHE|COSINE(LORENTZ)
  INITIAL VALUE = <real>value(1.0)
  SOURCE COEFFICIENT = <real>source_coeff(1.0)
  DIFFUSION COEFFICIENT = <real>diff_coeff(0.0)
  REACTION COEFFICIENT = <real<rxn_coeff(1.0)
  SOLVE AT INITIALIZATION = OFF|ON(OFF)
  SOLVE INCREMENT = <integer>solv_incr(1)
  SOLVE EXPLICIT = OFF|ON(OFF)
  SUBCYCLES = <int>num_sub(1)
  GRADIENT CONFIGURATION = MODEL|CURRENT(MODEL)
  BEGIN PRESCRIBED FIELD
    SURFACE = <string>surf_name
    NODE SET|NODESET = <string>node_set_name
    FUNCTION = <string>func_name
```

```
    END
END
```

In both reaction diffusion and gradient damage, assemblies may contain blocks, or assemblies of these.

20.5.   Phase Field Fracture Material

An experimental fracture capability. Currently in early development and not recommended for use.

```
BEGIN MATERIAL <name>
  BEGIN PARAMETERS FOR MODEL PHASE_FIELD_LINEAR_ELASTIC
  END
END

BEGIN PFFRAC phasefieldfracname
  FRACTURE LENGTH SCALE = <real>reaction_value
  CONDITIONING COEFFICIENT = <real>diffusion_value
  FRACTURE ENERGY = <real>forcing_value
END
```

20.6. Discrete Element Method (DEM)

The discrete element method is a particle based element formulation. This method is in early development, experimental, and currently not recommended for use.

```
BEGIN DEM OPTIONS
 ....
END
BEGIN DEM SECTION
 ....
END
```

20.7.   Q1P0 Element

A selectively integrated formulation is specified with the command `FORMULATION = Q1P0`.
This is only available for 8-node hexahedral element blocks.

```
BEGIN SOLID SECTION <string>solid_section_name
  ...
  FORMULATION = Q1P0
  ...
  Q1P0 STABILIZATION THRESHOLD = <real>threshold(0.0)
  Q1P0 TIMESTEP SCALE FACTOR = <real>scale_factor(0.95)
  Q1P0 TIMESTEP WAVE SPEED = <string>VOLUMETRIC|SHEAR|
    AUTOMATIC(AUTOMATIC)
  Q1P0 TIMESTEP LENGTH SCALE = <string>DEFORMED_NODAL_DISTANCE|
    MINIMUM_MAPPING_STRETCH|INSCRIBED_SPHERE_DIAMETER
    (MINIMUM_MAPPING_STRETCH
END [SOLID SECTION <string>solid_section_name]
```

In the Q1P0 element formulation, the internal forces arising from material stress are selectively
integrated. Forces arising from the pressure component of the stress are integrated using a single
integration point while forces arising from the deviatoric stress are integrated using a $2 \times 2 \times 2$
Gauss rule.

The only `STRAIN INCREMENTATION` option available for this element is
`STRONGLY_OBJECTIVE`.

When post-processing information such as the plastic strain with this element, information at the
first integration point should typically be used as it is more accurate than at any other point. The
first integration point corresponds to the location at the center of the element where the pressure
response is evaluated.

Warning: Material model evaluations at the q1p0 element's deviatoric integra-
tion points can result in spurious high and low locked pressures for incompressible
(or nearly incompressible) material models. The q1p0 element avoids the pressure
locking when calculating the internal forces (for the balance of linear momentum)
by discarding the pressures calculated at the deviatoric integration points and re-
placing them with the pressure from the central integration point. Note that the
locked pressures are replaced during element calculations, not inside the constitu-
tive model. This means that material models and element death criteria that fail
or accumulate damage based on pressure may be adversely affected by this devi-
atoric pressure locking. For this reason, the selective-deviatoric (SD) element is
generally preferred for material failure analyses. The SD element calculates a sin-
gle average element volumetric strain and passes that average volumetric strain to
all material integration points. The volume averaging of strain in the SD element
prevents pressure locking in the material constitutive equations and in the overall
element response.

Stress-based values such as `stress` and values derived from it such as `von_mises` are evaluated using a stress tensor taken from a volumetric average of the 8 deviatoric Gauss points for the deviatoric response combined with a pressure response at the central integration point.

The `Q1P0 STABILIZATION THRESHOLD` command modifies the formulation to provide additional stabilization as elements become distorted at the cost of accuracy. If a simulation produces inverted elements, these may be able to be mitigated by providing a value of 0.25. One may look at the `stabilization_factor` element variable to determine if this option is being activated in the analysis. A value of 0 in this variable corresponds to a fully q1p0 formulation while a value of 1 corresponds to a fully integrated formulation. Keep in mind that if this is changed from the default value of 0, the formulation is no longer truly q1p0.

Three additional parameters are available to select how the critical time step is evaluated: `Q1P0 TIMESTEP SCALE FACTOR`, `Q1P0 TIMESTEP WAVE SPEED`, and `Q1P0 TIMESTEP LENGTH SCALE`. The critical time step is evaluated using the following formula:

$$\text{timestep} = \text{scale factor} \times \frac{\text{length scale}}{\text{wave speed}}$$

The `Q1P0 TIMESTEP SCALE FACTOR = scale_factor` command scales the calculated time step for elements with this section. The default of 0.95 should be sufficient for almost all analyses. Lowering this slightly may provide better results in certain circumstances. If another time step scale factor is specified within the `PARAMETERS FOR PRESTO REGION` block, they are effectively multiplied together for elements using this section.

The `Q1P0 TIMESTEP WAVE SPEED` command chooses the wave speed used by the time step calculation. The default, `AUTOMATIC`, should be sufficient for all analyses. The `VOLUMETRIC` option calculates wave speed using the bulk modulus while the `SHEAR` option uses the shear modulus. The `AUTOMATIC` option uses the maximum of the other two options.

The `Q1P0 TIMESTEP LENGTH SCALE` selects the method used to calculate the length scale of the element. The default `MINIMUM_MAPPING_STRETCH` option calculates this as the minimum stretch from the mapping between a unit cube and the current configuration of the element. While this option is relatively slow, it is robust. The `DEFORMED_NODAL_DISTANCE` option calculates this as the minimum non-zero node to node distance within the element. This is the fastest option and a potential increase in speed is achieved by selecting it at the cost of robustness. The `INSCRIBED_SPHERE_DIAMETER` option calculates this as the diameter of the largest sphere which can fit inside the element.

References

[1] Sierra/SolidMechanics Team. Sierra/SolidMechanics VOTD User's Guide. Technical Report In draft, Sandia National Laboratories, Albuquerque, NM, 2019.

INDEX

This page intentionally left blank.

DISTRIBUTION

Email—Internal ██████████

Name	Org.	Sandia Email Address
Technical Library	1911	sanddocs@sandia.gov