SANDIA REPORT
SAND2025-11084O
Printed October 14, 2025

Sandia National Laboratories

Sierra/SolidMechanics 5.26 Capabilities in Development

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

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from

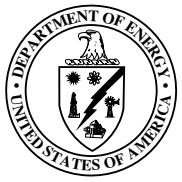
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods



Abstract

This manual 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 User Manual. 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.

Acknowledgements

This document is the result of the collective effort of many individuals. The current core development team responsible for the Sierra/SM codes includes Frank N. Beckwith, Michael R. Buche, Gabriel J. de Frias, Scott O. Gampert, Kevin L. Manktelow, Mark T. Merewether, Scott T. Miller, Krishen J. Parmar, Matt G. Rand, Timothy R. Shelton, Jesse D. Thomas, Jeremy Trageser, Benjamin C. Treweek, Michael G. Veilleux, and Ellen B. Wagman. This document is written and maintained by this team.

Many others have contributed to this document, either directly or indirectly. These include, but are not limited to Kenneth (Noel) Belcourt, Guy L. Bergel, Manoj K. Bhardwaj, Nicole L. Breivik, Arthur Brown, James V. Cox, Nathan K. Crane, David M. Day, J. Franklin Dempsey, James W. Foulk III, Brian D. Giffin, Steven P. Gomez, Jeffrey D. Gruda, Arne S. Gullerud, Jason D. Hales, Daniel C. Hammerand, Martin W. Heinstein, David M. Hensinger, Michael C. Hillman, Jason T. Ivey, Jacob Koester, Timothy D. Kostka, J. Richard Koteras, San Le, Brian T. Lester, Alex J. Lindblad, David J. Littlewood, Chi S. (David) Lo, Kevin N. Long, Edward Love, Matt D. Mosby, Kyran D. (Kim) Mish, Jakob T. Ostien, Kendall H. Pierson, Julia A. Plews, Vicki L. Porter, Ramon Reyes, Rhonda K. Reinert, Nathaniel S. Roehrig, William M. Scherzinger, Gregory D. Sjaardema, Benjamin W. Spencer, Brian L. Stevens, Michael R. Tupek, Julia R. Walker, Gerald W. Wellman, Patrick G. Xavier, and Edouard Yreux.

Front Matter

List of Tables

List of Figures

1 Introduction

This document is a manual 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.

This page left blank

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 microstructures 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:

- Internal force algorithm is called in the reference region to compute rate of deformation.
- Each RVE gets the rate of deformation from its integration point on its parent element in the reference region.

- The rate of deformation is applied to each RVE as a periodic boundary condition using prescribed velocity.
- The RVE region is solved to obtain the stress in each element of each RVE.
- The stresses in the elements of an RVE are volume-averaged over the RVE.
- Each RVE passes its homogenized or volume-averaged stress tensor back to its integration point of its parent element in the reference mesh.
- 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. Fig. 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.

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.

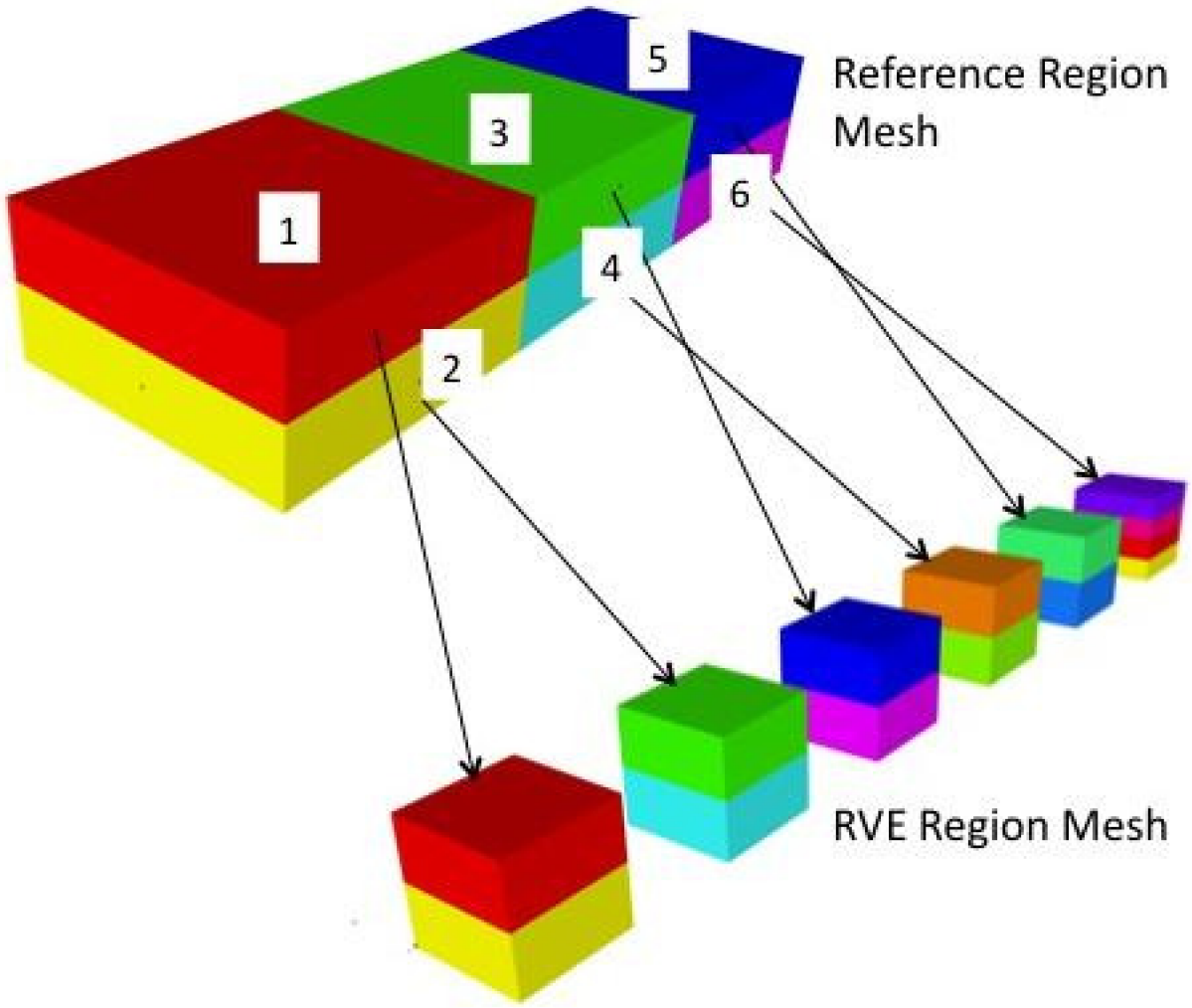


Fig. 2.1 Example of meshes for RVE analysis.

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 User Manual 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 User Manual. 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.

```
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
```

(continues on next page)

(continued from previous page)

```
   #
   # 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

```
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 Fig. 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 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 User Manual.

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
  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 User Manual 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/SM will 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.

- Define the trial coarse region based off of everything not in the fine region.
- Compute the critical time step of the coarse region as the smallest element time step in that region.

- Compute the maximum time step each node can be integrated at as the smallest time step of any element near the node.
- 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.
- 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.
- 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/SM can be found in [7]. The portions of this paper specifically used in Sierra/SM are: 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 User Manual for more information on mesh rebalancing.

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

This page left blank

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 the Sierra/SM User Manual) so that - 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. 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.

4.1.1 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 4.1.2.

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.

4.1.2 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
```

(continues on next page)

(continued from previous page)

```
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.

> **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 Sierra/SM development.

```
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.

This page left blank

6 Solvers and Solver Options

This chapter lists some solvers and solver options that are under development.

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 Fig. 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.

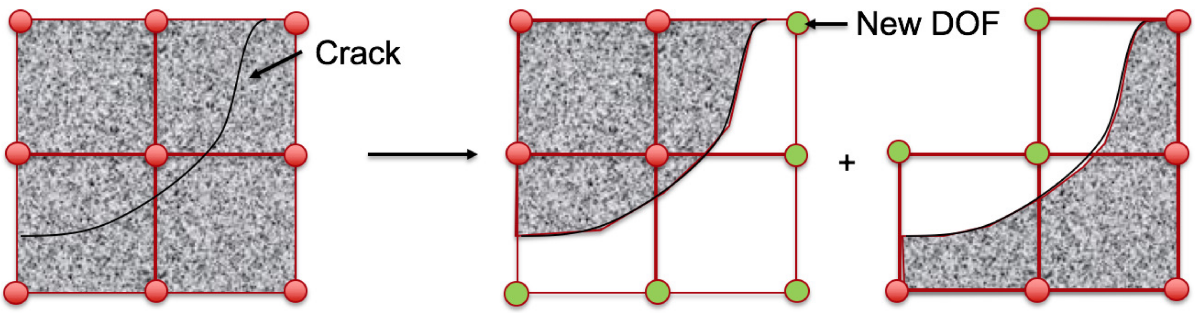
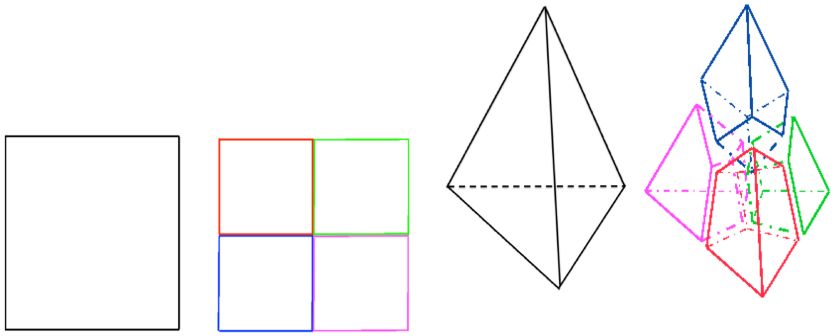
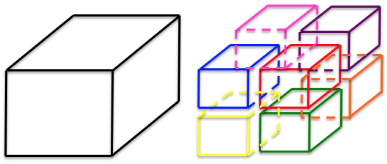


Fig. 7.1 Example of XFEM element cutting and duplication.

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 Fig. 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.

Fig. 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)
```

(continues on next page)

(continued from previous page)

```
  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, sand 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.

7.2.2.1 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 User Manual 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.
- `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 emph{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 emph{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.”

7.2.2.2 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.

7.2.2.3 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 Fig. 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.

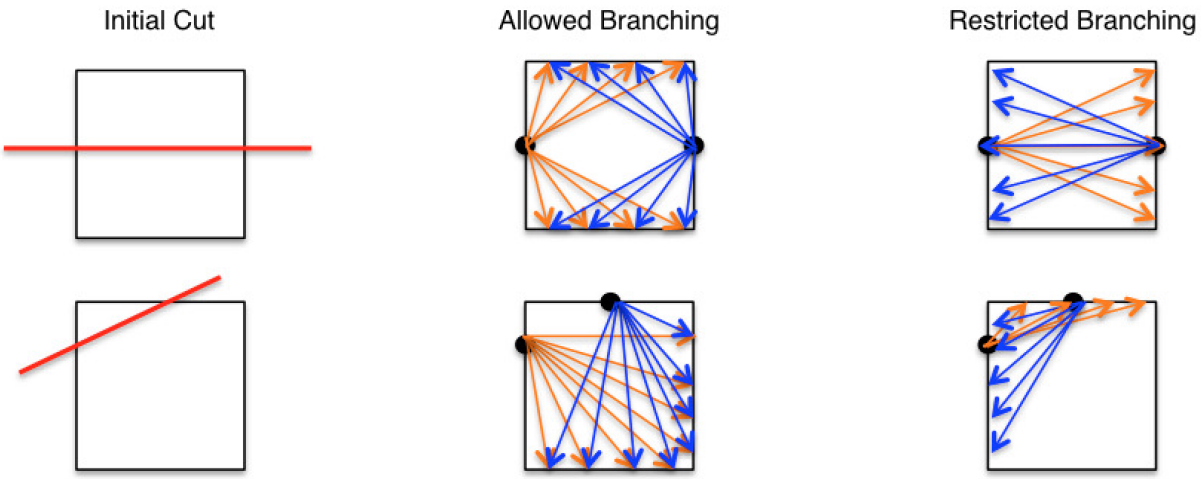


Fig. 7.3 Example of allowed and restricted branching.

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.

7.2.4.1 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.

7.2.4.2 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.

7.2.4.3 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 [24], 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 the Sierra/SM User Manual. 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 User Manual.

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

8 External Loadstep Predictor

> **Warning:** The external loadstep predictor is for implicit analyses only.

Production-ready loadstep predictor types are available in the Sierra/SM User Guide. 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.

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.
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.

This page left blank

9 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 (9.1). 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.

$$\left(\frac{u_n}{u_{n_{crit}}}\right)^p + \left(\frac{u_t}{u_{t_{crit}}}\right)^p < 1.0. \tag{9.1}$$

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 9.1 describes the output variables available on the bolt elements.

Table 9.1 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.

10 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 TheusersguideSection~ref{user:ele:beamsec} 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 [18].

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.

11 Contact

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

11.1 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 the Sierra/SM User Manual.

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.

11.2 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 patch for each face in `Side A` that is in contact. For quad faces, this patch is a Gregory patch (following the method of [19]]), while for tri3 faces, it is a Nagata patch (following the method of [[17]). This is done to predict 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 and tri3 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.

11.2.1 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 [21].

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.

11.2.2 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.

11.3 Auto Angle Contact Subsets

```
BEGIN CONTACT SURFACE <string>name
  INCLUDE ALL BLOCKS
  BLOCK = <string list>block_names
  ELEMENT = <int list>elem_numbers
  SURFACE = <string list>surface_names
  ASSEMBLY = <string list>assembly_names
  REMOVE BLOCK = <string list>block_names
```

(continues on next page)

(continued from previous page)

```
  REMOVE SURFACE = <string list>surface_names
  CONTACT TYPE = STANDARD | SMOOTH (STANDARD)
  AUTO SUBSET <string>subname [NUMBER OF SUBSETS <integer>] [ANGLE
↪<real>(60)]
END [CONTACT SURFACE <string>name]
```

A contact surface can optionally be subdivided into multiple sub-surfaces based on the angle between surface normals. Contact surfaces that are connected and whose normals are less than the angle are grouped together into sub-surfaces. The angle can be specified in degrees with the optional *ANGLE* argument. It will create the number of subsets specified by the optional *NUMBER OF SUBSETS* argument. The angle is set to 60 degrees by default. By default, the number of subsets created will be 360.0 divided by the angle defined or set by default rounded down to the nearest integer. Thus, if both defaults are used, it will create 6 subsets. Only one auto subset command line can be specified per contact surface. Shared nodes between created subsets will go into the lowest numbered sub-surface. The ordering of auto subsets is based on the lowest face ID on the surface. It is not based on the normal direction or any other identifying features of the mesh. If the IDs on the mesh are changed, the surface indices will change.

For example, the following command:

```
BEGIN CONTACT SURFACE surf1
  BLOCK = block_1
  AUTO SUBSET sub1 NUMBER OF SUBSETS 6 ANGLE 45.0
END
```

on a unit hex will create six sub-surfaces, `surf1_sub1_auto_1`, `surf1_sub1_auto_2`, … `surf1_sub1_auto_6`, since *NUMBER OF SUBSETS* is set to 6. Since the angle between each set of surface normals exceeds the specified 45 degrees, each sub-surface will contain 1 contact surface.

Warning: If more subsets are found than the number specified for *NUMBER OF SUBSETS* or the default (6), the excess contact surfaces will be put into the default contact surface and may give unexpected results.

Warning: Specifying an excessive number of subsets for *NUMBER OF SUBSETS* will create empty subsets that will significantly slow down the simulation.

Warning: Within a contact surface, only one subset type can be specified. Thus, subset with normal and auto subset cannot be specified for a given contact surface. For example, the

following is not permitted:

```
BEGIN CONTACT SURFACE surf1
  BLOCK = block_1
  SUBSET sub1 WITH NORMAL  1  0  0
  AUTO SUBSET sub2 NUMBER OF SUBSETS 6 ANGLE 45.0
END
```

This page left blank

12 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.

12.1 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.

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

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.

12.2 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. Fig. 12.1 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.

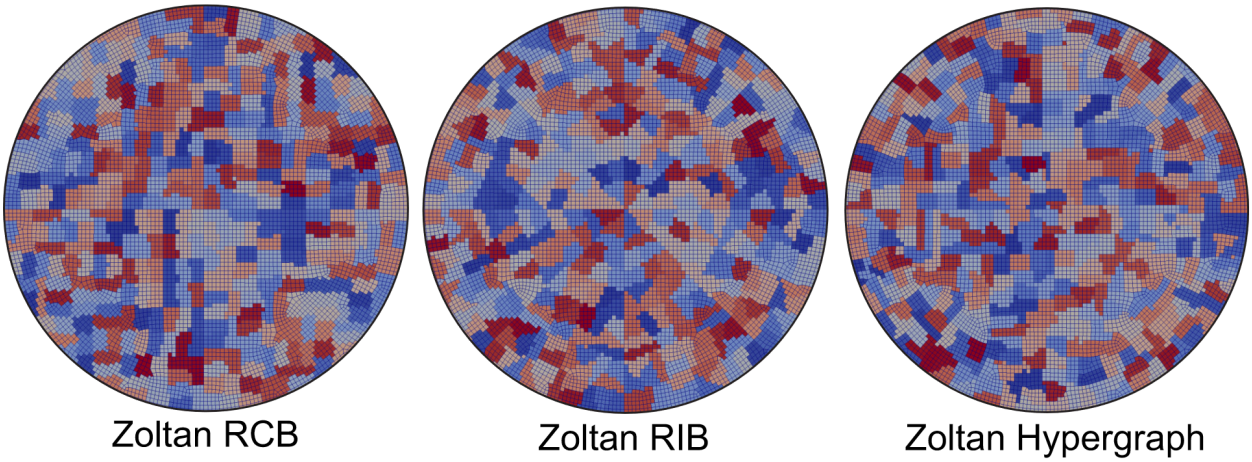


Fig. 12.1 Illustration of 400 nonlocal partitions at a sharp crack tip using Zoltan Recursive Coordinate Bisection (RCB), Zoltan Recursive Inertial Bisection (RIB), and Zoltan Hypergraph partitioning methodologies. Note that Zoltan Hypergraph can generate non-contiguous domains. The default partitioning methodology in Sierra is Zoltan RCB.

Initial findings employing geometric partitioning illustrated a sensitivity to domain shape. A re-examination of Fig. 12.1 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 [15]], we gravitated towards clustering algorithms and the resulting isotropy [[8]. Fig. 12.2 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 Fig. 12.2. 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.

12.3 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 $V = \text{length}^3$. 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 User Manual. 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

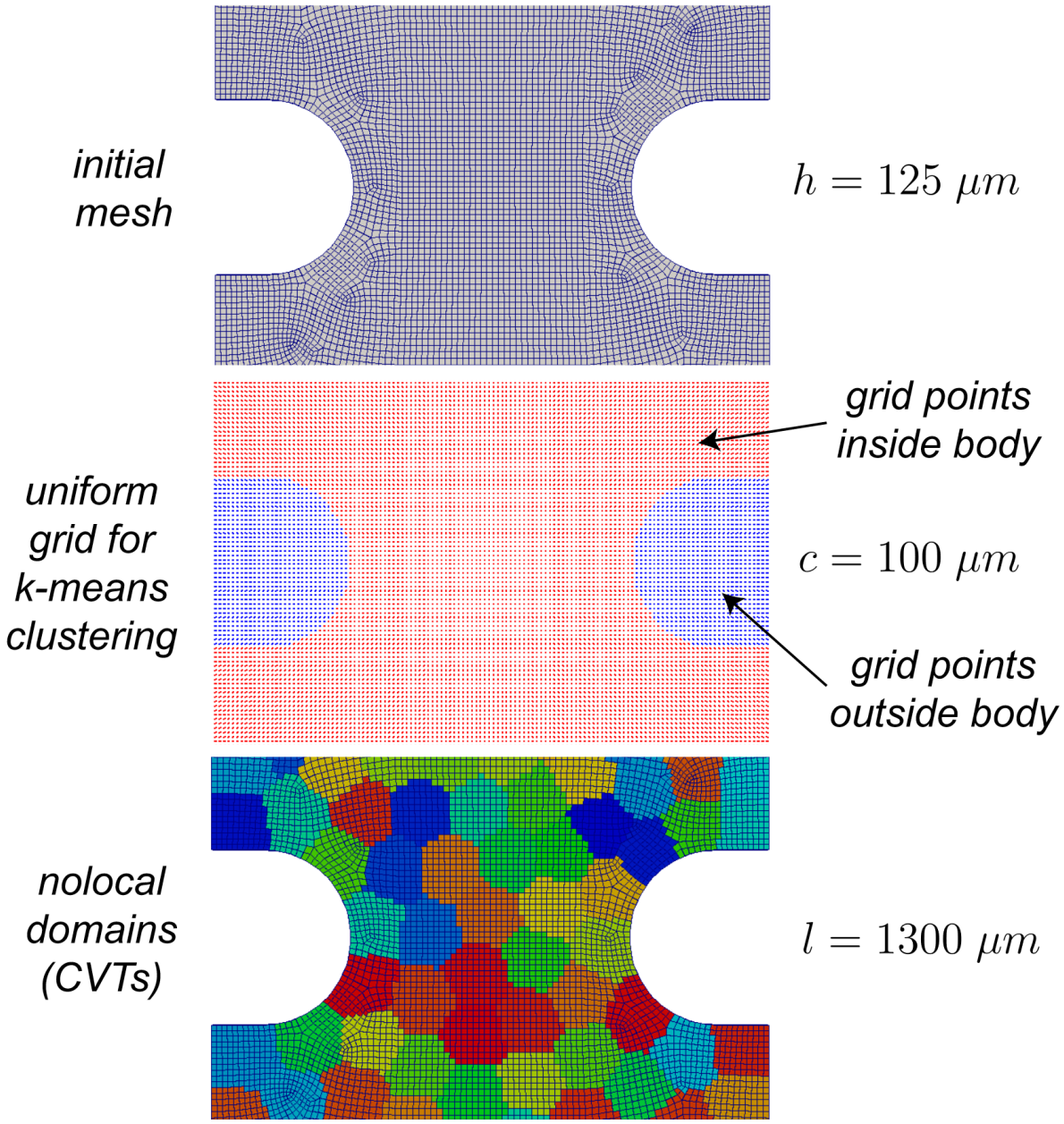


Fig. 12.2 Nonlocal domains derive from a Centroidal Voronoi Tessellation (CVT). A partitioned mesh for parallel processing with element size h determines the boundaries of a uniform grid with cell size c. K-means clustering evolves a set of N centroids into a CVT.

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 Sierra/SM 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".

12.4 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

- 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.
- 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.
- 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.
- 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 $\sim$ 1000 and $\sim$ 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.
- 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.

- 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.
- 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.
- 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.
- 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.

13 POD

Proper Orthogonal Decomposition (POD) and Explicit Control Modes (ECM) 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.

Explicit only

POD is an experimental analysis technique.

13.1 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 [10].

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, 1e-4 seconds here, the POD modes are updated using the new information.

14 Conforming Reproducing Kernel Method

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

14.1 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 [14], [13]]. 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, etc.). CRK uses robust integration techniques for nearly incompressible materials [[16]] and is being used to explore bond-based material failure methodologies [[13].

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 [9]]. 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 [[16]. 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 meshing. Many features are tailored to tetrahedral inputs.

14.2 Usage Guidelines

The following block command defines a `CRK SECTION`.

```
# Within Sierra scope...
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 stress-strain conjugate pairs to use
  # ADAGIO uses a Cauchy-stress/rate-of-deformation conjugate
  # LOGARITHMIC uses Hencky strain/"Hencky" stress
  STRAIN MEASURE = ADAGIO|LOGARITHMIC (ADAGIO)

  # Controls when post processing occurs for performance
  POST PROCESS EVERY STEP = FALSE|TRUE (TRUE)

  # Creates face entities to apply contact to.  Required when
  # using Dash contact with CRK.
  CREATE CONTACT FACES = ON|OFF (OFF)

  # Commands controlling bond-based failure modeling in CRK
  FAILURE METHOD = BOND|NONE (NONE)
  FAILURE INDEX = <integer>failure_index
  FAILURE CRITERION = <real>bond_failure_criterion
  BOND OPENING DISPLACEMENT = <real>bond_opening_displacement
  BOND SECONDARY CRITERION = <real>bond_secondary_criterion
  SHOW BONDS = ON|OFF (OFF)

  # Controls which nodes are DOFs in the model
  # (for nearly-incompressible media)
  THEX ACTIVE NODES = NODES|EDGES|FACES|CENTROIDS (NODES)

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.

Warning: The option SHAPE FUNCTION TYPE = TWO_STAR is only compatible with FORMULATION = ELEMENT described below. Additionally, the input mesh must be a non thex, tetrahedral topology.

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.

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, referred to as an integration "cell." This results in the SCNI strategy [9]. 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 as described in Section 14.2.2.

If NODAL_FROM_THEX or TET_FROM_THEX are specified above, the nodes in the CRK domain will be divided into an "active" group and an "inactive" group. The "active" group will be the DOF-carrying nodes and usually represents the original tet nodes before the thex command is issued in Cubit. The "inactive" group will contain all other nodes which are effectively evaluation points used to setup the nodal integration domains used in CRK. These complementary sets may

be adjusted using the `THEX ACTIVE NODES` command as described in Section 14.2.2.

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`.

The command `STRAIN MEASURE = ADAGIO|LOGARITHMIC` controls which strain to use when evaluating the constitutive law. The default is `ADAGIO` which uses the rate of deformation (the default in Thecode). The option `LOGARITHMIC` uses the logarithmic strain, or Hencky strain, and its corresponding conjugate stress instead.

14.2.1 Contact

CRK currently supports Dash contact but the setup will be slightly different depending on whether the command `FORMULATION` is set to `ELEMENT` versus `NODAL_FROM_THEX` or `TET_FROM_THEX`. For both cases, the shape function type must be either `SPHERICAL_SURFACE_KRONECKER` or `LINEAR_TET`.

In the case that `FORMULATION = ELEMENT`, every node in the mesh will be labeled as an "active" node and Dash contact may be setup as if the simulation was using normal elements without any additional modification as given in the Sierra/SM User Manual.

In the case of `FORMULATION = NODAL_FROM_THEX` or `TET_FROM_THEX`, the CRK surface must be manually created using the command `CREATE CONTACT FACES = ON`. This command will skin the CRK block(s) and create a new block with the name `<block_name>_contact_surface` where `<block_name>` is the original block name. This new surface is necessary to setup CRK shape function weights and area densities that Dash requires. These weights spread the contact forces from inactive nodes to active ones associated with the problem unknowns. Once this surface is created, it may be referenced within the contact definition as, for example

```
# Within Sierra scope...
BEGIN CRK SECTION crk_section
  SHAPE FUNCTION TYPE = SPHERICAL_SURFACE_KRONECKER
  FORMULATION = NODAL_FROM_THEX
  CREATE CONTACT FACES = ON
END CRK SECTION crk_section
...
# Within Region scope...
BEGIN CONTACT DEFINITION collide
```

(continues on next page)

(continued from previous page)

```
  SEARCH = Dash
  CONTACT SURFACE crk_surface CONTAINS block_1_contact_surface
  CONTACT SURFACE block_2 CONTAINS block_2

  BEGIN INTERACTION DEFAULTS
    GENERAL CONTACT = ON
    FRICTION MODEL = FRICTIONLESS
    INTERACTION BEHAVIOR = SLIDING
  END INTERACTION DEFAULTS
END CONTACT DEFINITION collide
```

where in the above `block_1` is the CRK block and `block_2` is an FEM block. Dash will then take the information present on the CRK surface including shape function values and weights in order to create the contact representation.

14.2.2 Advanced integration methods

By default, the $\overline{\mathbf{F}}$ integration technique [16] is activated if the command `FORMULATION = NODAL_FROM_THEX` or `TET_FROM_THEX` and if the `NUMBER OF INTEGRATION SUBCELLS` is greater than one. In this case, the sub-tet hex elements will be used for sub-cells in the $\overline{\mathbf{F}}$ method for improving performance with nearly incompressible materials. In this context a "sub-cell" is a collection of sub-tet hex elements which share an original tet node and represents a deviatoric domain of the integration cell. 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 and defaults to `NODES`. 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. This option is useful for specifying additional "bubble" DOFs when using advanced integration methods such as $\overline{\mathbf{F}}$ [16].

14.2.3 Bond-based fracture

> **Warning:** The bond-based fracture method is only compatible with the command `FORMULATION = NODAL_FROM_THEX`.

CRK currently supports an experimental fracture formulation known as bond-based fracture. This formulation creates a "bond" between each node-node pair and maintains a damage variable associated with the bond which degrades the incremental deformation gradient and resulting internal force [13]. The failure criterion is specified using the `FAILURE INDEX` command to specify the emph{0-based} state variable index of the material model in use along with `FAILURE CRITERION` command which specifies the threshold the state variable must exceed before the node is considered "damaged." When both nodes of a bond have reached the criterion, the bond is marked as damaged and begins to degrade linearly over a distance given by the `BOND OPENING DISPLACEMENT` command. This bond opening displacement is given by the relative displacement between the two nodes (in an abuse of terminology, the "stretch" of the bond).

The criterion above requires both nodes of a bond to be flagged as "damaged" before the bond can begin degrading. An alternative criterion may be specified via the command `BOND SECONDARY CRITERION = <real>`u_{second}. For any bond that fails due to the first criterion, this option creates a failure plane passing through that bond's midpoint normal to the bond. Any bond which shares a node with the first failed bond and intersects the failure plane may also fail due to a second criterion given by

$$u_{ab} - \bar{u}_{ab} \geq u_{\text{second}},$$

where $u_{ab} = \|\mathbf{u}_b - \mathbf{u}_a\|$ is the relative displacement between nodes a and b, and $\bar{u}_{ab}$ is the relative displacement at the time one of the bond's nodes becomes damaged due to the criterion given by `FAILURE CRITERION` and `FAILURE INDEX` described above. The bond will then begin to degrade over the `BOND OPENING DISPLACEMENT` as before.

> **Warning:** The bonds may be viewed using the `SHOW BONDS = ON` command. However, this currently only works on a single processor.

14.2.4 Output

Note, CRK shape functions in general do not possess the Kronecker delta property, i.e.

$$N_I(x_J) \neq \delta_{IJ}.$$

As such, nodal variables such as `displacement` and `velocity` **are not true displacements or velocities** but rather the emph{generalized coefficients} of those fields. In addition, other

quantities of interest will also be generalized (e.g. mass, force, etc.). These generalized values must be converted to their physical counterparts by interpolating these coefficients with their respective shape functions. These quantities are stored in the variables `physical_displacement` and `physical_velocity`. As a workaround, one can provide an alternative name in the results output to switch the meaning of `displacement` and `physical_displacement` as follows:

```
# Within Region scope...
BEGIN RESULTS OUTPUT my_output
  ...
  NODAL VARIABLES = physical_displacement as displacement
  NODAL VARIABLES = displacement as gen_displacement
  ...
END RESULTS OUTPUT my_output
```

which will output the displacement in a form amenable to Paraview or Ensight.

Warning: If an analysis contains both FEM and CRK formulations, then the `displacement` and `velocity` fields will be generalized for the CRK blocks and physical for the FEM blocks. Since `physical_displacement` does not exist in the FEM portions, the workaround mentioned above will output zero physical displacements for those blocks.

The output variables available for CRK are given below in Table 14.1 and Table 14.2 below. Bond-based fracture specific variables are given in Table 14.3.

Table 14.1 Nodal Variables for the CRK Formulation

Name	Type	Comments
physical_displacement	Vector_3D	The true displacement predicted by CRK
physical_velocity	Vector_3D	The true velocity predicted by CRK
displacement	Vector_3D	The generalized displacement predicted by CRK, i.e. the displacement coefficients
velocity	Vector_3D	The generalized velocity predicted by CRK, i.e. the velocity coefficients
nodal_status	Integer	Indicates whether a node is active (carrying a DOF) or inactive
max_edge_length	Real	Maximum edge length connected to the node of the original, non thex mesh
local_support_size	Real	Local support size radius when SHAPE FUNCTION TYPE is one of SPHERICAL or SPHERICAL_SURFACE_KRONECKER
num_neighbors	Integer	Number of active neighbors which have shape function value at the node
crk_contact_mass	Real	Generalized mass scattered to inactive nodes for Dash purposes

Table 14.2 Element Variables for the CRK Formulation

Name	Type	Comments
cell_id	Integer	The ID of the CRK strain smoothing cell
sub_cell_id	Integer	The ID of the CRK strain smoothing sub-cell
cell_volume	Real	The volume of the smoothing cell
sub_cell_volume	Real	The volume of the smoothing sub-cell

Table 14.3 Output Variables for Bond-Based FeFpmodelnameracture

Name	Type	Comments	
node_damage	Nodal	Integer	Flag indicating if the failure criterion is met at a given node's material point
damage_fraction	Element	Real	Fraction of damaged bonds over total bonds associated with an integration cell
bond_degradation	Element	Real	Scalar between 0 and 1 to indicate level of damage in the bond where 1 indicates no damage and 0 indicates fully damaged (only available when SHOW BONDS = ON)

15 Material Models

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

15.1 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

- two of the five general elastic material constants,
- directional properties, and
- the coordinate system.

The following is a brief description of each input.

- See the Sierra/SM User Manual 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 User Manual. The material orientation may then be specified using **one** of the following approaches:
  - 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. The Sierra/SM User Manual 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.
  - 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 User Manual. 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`.
  - 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 User Manual 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 15.1.

Table 15.1 State Variables for ELASTIC_ORTHOTROPIC Model

Index	Name	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

15.2 Elastic Orthotropic Damage Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_DAMAGE
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
  BULK MODULUS   = <real>
  LAMBDA         = <real>
  #
  # Required parameters
  #
  E11    = <real>
  E22    = <real>
  E33    = <real>
  NU12   = <real>
  NU13   = <real>
  NU23   = <real>
  G12    = <real>
  G13    = <real>
  G23    = <real>
  ALPHAD = <real>
  BETAD  = <real>
  GAMMA0 = <real>
  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
```

(continues on next page)

(continued from previous page)

```
  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:

  -[textbullet] Young's modulus is defined with the `YOUNGS MODULUS` command line.
  -[textbullet] Poisson's ratio is defined with the `POISSONS RATIO` command line.
  -[textbullet] The bulk modulus is defined with the `BULK MODULUS` command line.
  -[textbullet] The shear modulus is defined with the `SHEAR MODULUS` command line.
  -[textbullet] 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 User Manual.

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

15.3 Elastic Orthotropic Fail Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_FAIL
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
  BULK MODULUS   = <real>
  LAMBDA         = <real>
  TWO MU         = <real>
  #
  # Required parameters
  #
  E11 = <real>e11
  E22 = <real>e22
  E33 = <real>e33
```

(continues on next page)

(continued from previous page)

```
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
```

(continues on next page)

(continued from previous page)

```
#
# 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
```

(continues on next page)

(continued from previous page)

```
  #
  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 [11] 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 15.1, 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
```

(continues on next page)

(continued from previous page)

```
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
```

(continues on next page)

(continued from previous page)

```
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 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 Table 15.1 and Table 15.2.

Table 15.2 Additional State Variables for ELASTIC ORTHOTROPIC FAIL Model.

Index	Name	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

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.

15.4 Elastic Orthotropic Shell Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_ORTHOTROPIC_SHELL
  #
  # required parameters
  #
  YOUNGS MODULUS RR = <real>
  YOUNGS MODULUS SS = <real>
  YOUNGS MODULUS TT = <real>
  POISSONS RATIO RS = <real>
  POISSONS RATIO ST = <real>
  POISSONS RATIO TR = <real>
  SHEAR MODULUS RS  = <real>
  SHEAR MODULUS RT  = <real>
  SHEAR MODULUS ST  = <real>
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 User Manual 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

$$\varepsilon_{TT} = -\frac{1}{1 - \nu_{RS}\nu_{SR}} \left[\left(\nu_{RT} + \nu_{RS}\nu_{ST} \right) \varepsilon_{RR} + \left(\nu_{ST} + \nu_{SR}\nu_{RT} \right) \varepsilon_{SS} \right]$$

and results in the following stiffness

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

where

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

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`

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

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

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

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

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

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

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

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 emph{only}, the original input syntax remains valid. However, the new behavior of the model is to ignore any input value of ν_{SR} and compute it automatically as $\nu_{SR} = \nu_{RS}(E_{SS}/E_{RR})$. 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 ν_{ST} or ν_{TR}, it will default to zero. For best results emph{all required values should be input} in future usages of this model.

15.5 BCJ Model

```
BEGIN PARAMETERS FOR MODEL BCJ
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
  BULK MODULUS   = <real>
  LAMBDA         = <real>
  TWO MU         = <real>
  #
  #
  #
  C1 = <real>c1
  C2 = <real>c2
  C3 = <real>c3
  C4 = <real>c4
  C5 = <real>c5
  C6 = <real>c6
  C7 = <real>c7
  C8 = <real>c8
  C9 = <real>c9
  C10 = <real>c10
  C11 = <real>c11
  C12 = <real>c12
  C13 = <real>c13
  C14 = <real>c14
  C15 = <real>c15
  C16 = <real>c16
  C17 = <real>c17
  C18 = <real>c18
  C19 = <real>c19
  C20 = <real>c20
  DAMAGE EXPONENT = <real>damage_exponent
```

(continues on next page)

(continued from previous page)

```
  INITIAL ALPHA_XX = <real>alpha_xx
  INITIAL ALPHA_YY = <real>alpha_yy
  INITIAL ALPHA_ZZ = <real>alpha_zz
  INITIAL ALPHA_XY = <real>alpha_xy
  INITIAL ALPHA_YZ = <real>alpha_yz
  INITIAL ALPHA_XZ = <real>alpha_xz
  INITIAL KAPPA = <real>initial_kappa
  INITIAL DAMAGE = <real>initial_damage
  YOUNGS MODULUS FUNCTION = <string>ym_function_name
  POISSONS RATIO FUNCTION = <string>pr_function_name
  SPECIFIC HEAT = <real>specific_heat
  THETA OPT = <integer>theta_opt
  FACTOR = <real>factor
  RHO    = <real>rho
  TEMP0  = <real>temp0
END [PARAMETERS FOR MODEL BCJ]
```

The BCJ plasticity model is a state-variable model for describing the finite deformation behavior of metals. It uses a multiplicative decomposition of the deformation gradient into elastic, volumetric plastic, and deviatoric parts. The model considers the natural configuration defined by this decomposition and its associated thermodynamics. The model incorporates strain rate and temperature sensitivity, and damage, through a yield-surface approach in which state variables follow a hardening-minus-recovery format.

Because the BCJ model has such an extensive list of parameters, we will not present the usual synopsis of parameter names with command lines. As with most other material models, the `thermal strain option` is used to define thermal strains. See the Sierra/SM User Manual for further information on defining and activating thermal strains. In addition, only two of the five elastic constants are required. The user should consult [5], [4], [3] for a description of the various parameters. The parameters for the `SPECIFIC HEAT`, `THETA OPT`, `FACTOR`, `RHO`, and `TEMP0` command lines are used to accommodate changes to the model for heat generation due to plastic dissipation. For coupled solid/thermal calculations, the plastic dissipation rate is stored as a state variable and passed to a thermal code as a heat source term. For uncoupled calculations, temperature is stored as a state variable, and temperature increases due to plastic dissipation are calculated within the material model.

If temperature is provided from an external source, `theta_opt` is set to 1. If the temperature is calculated by the BCJ model, `theta_opt` is set to 1.

Output variables available for this model are listed in Table 15.3.

Table 15.3 State Variables for BCJ Model

Name	Description
BACK_STRESS_XX	back stress tensor - xx component
BACK_STRESS_YY	back stress tensor - yy component
BACK_STRESS_ZZ	back stress tensor - zz component
BACK_STRESS_XY	back stress tensor - xy component
BACK_STRESS_YZ	back stress tensor - yz component
BACK_STRESS_ZX	back stress tensor - zx component
KAPPA	hardening scalar
DAMAGE	damage term
DAMAGE_RATE	rate of change of damage term
EQPS	equivalent plastic strain
THETA	temperature for adiabatic heating
HEAT	rate of heating due to plastic dissipation
YM	Young's modulus
PR	Poisson's ratio

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.

15.6 Karagozian and Case Concrete Model

```
BEGIN PARAMETERS FOR MODEL KC_CONCRETE
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
  BULK MODULUS   = <real>
  LAMBDA         = <real>
  TWO MU         = <real>
  #
  #
  #
  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
```

(continues on next page)

(continued from previous page)

```
  TENSILE STRENGTH = <real>tensile_strength
  UNLOAD BULK MODULUS FUNCTION = <string>bulk_function_name
END PARAMETERS FOR MODEL KC_CONCRETE
```

The Karagozian and 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 User Manual 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 ([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 $\lambda_z > \lambda_m$ ([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 ([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 [1] 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` ([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 [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
```

(continues on next page)

(continued from previous page)

```
  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
```

(continues on next page)

(continued from previous page)

```
   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
```

15.7 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
```

(continues on next page)

(continued from previous page)

```
  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 [6] 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 strengt 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.

15.8 Shape Memory Alloy

Please consult the LAMÉ manual.

15.9 Linear Elastic

```
BEGIN PARAMETERS FOR MODEL LINEAR_ELASTIC
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
  BULK MODULUS   = <real>
  LAMBDA         = <real>
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.

15.10 Elastic Three-Dimensional Anisotropic Model

```
BEGIN PARAMETERS FOR MODEL ELASTIC_3D_ANISOTROPIC
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
```

(continues on next page)

(continued from previous page)

```
BULK MODULUS    = <real>
LAMBDA          = <real>
TWO MU          = <real>
#
# Material coordinates system definition
#
COORDINATE SYSTEM               = <string>  coordinate_system_name
DIRECTION FOR ROTATION          = <real>    1|2|3
ALPHA                           = <real>    (degrees)
SECOND DIRECTION FOR ROTATION   = <real>    1|2|3
SECOND ALPHA                    = <real>    (degrees)
#
# Required parameters
#
STIFFNESS MATRIX 11 = <real>
STIFFNESS MATRIX 22 = <real>
STIFFNESS MATRIX 33 = <real>
STIFFNESS MATRIX 12 = <real>
STIFFNESS MATRIX 13 = <real>
STIFFNESS MATRIX 23 = <real>
STIFFNESS MATRIX 44 = <real>
STIFFNESS MATRIX 55 = <real>
STIFFNESS MATRIX 66 = <real>
STIFFNESS MATRIX 45 = <real>
STIFFNESS MATRIX 46 = <real>
STIFFNESS MATRIX 56 = <real>
STIFFNESS MATRIX 14 = <real>
STIFFNESS MATRIX 15 = <real>
STIFFNESS MATRIX 16 = <real>
STIFFNESS MATRIX 24 = <real>
STIFFNESS MATRIX 25 = <real>
STIFFNESS MATRIX 26 = <real>
STIFFNESS MATRIX 34 = <real>
STIFFNESS MATRIX 35 = <real>
STIFFNESS MATRIX 36 = <real>
#
# Thermal strain functions
#
THERMAL STRAIN 11 FUNCTION = <real>
THERMAL STRAIN 22 FUNCTION = <real>
THERMAL STRAIN 33 FUNCTION = <real>
THERMAL STRAIN 12 FUNCTION = <real>
THERMAL STRAIN 23 FUNCTION = <real>
THERMAL STRAIN 13 FUNCTION = <real>
#
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

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

15.11 Karafillis Boyce Plasticity Model

```
BEGIN PARAMETERS FOR MODEL KARAFILLIS_BOYCE_PLASTICITY
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
  BULK MODULUS   = <real>
  LAMBDA         = <real>
  TWO MU         = <real>
```

(continues on next page)

(continued from previous page)

```
  #
  # Yield surface parameters
  #
  YIELD STRESS = <real>
  A            = <real> (4.0)
  C            = <real> (0.0)
  COEFF        = <real> (2.0/3.0)
  ALPHA 1      = <real> (1.0)
  ALPHA 2      = <real> (1.0)
  GAMMA 1      = <real> (1.5)
  GAMMA 2      = <real> (1.5)
  GAMMA 3      = <real> (1.5)
  #
  # Hardening model
  #
  HARDENING MODEL = LINEAR | POWER_LAW | USER_DEFINED |
    CUBIC_HERMITE_SPLINE
  #
  # Linear hardening
  #
  HARDENING MODULUS = <real>
  #
  # Power law hardening
  #
  HARDENING CONSTANT = <real>
  HARDENING EXPONENT = <real> (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> (degrees)
  SECOND DIRECTION FOR ROTATION = <real> 1|2|3
  SECOND ALPHA                  = <real> (degrees)
END [PARAMETERS FOR MODEL KARAFILLIS_BOYCE_PLASTICITY]
```

The Karafillis and Boyce model [12] 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 User Manual 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 15.4.

Table 15.4 State Variables for KARAFILLIS_BOYCE_PLASTICITY Model

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

15.12 Cazacu Plasticity Model

Please consult the LAMÉ manual.

15.13 Cazacu Orthotropic Plasticity Model

Please consult the LAMÉ manual.

15.14 Skorohod-Olevsky Viscous Sintering (SOVS)

Please consult the LAMÉ manual.

15.15 Hydra Plasticity

Please consult the LAMÉ manual.

15.16 Honeycomb Model

```
BEGIN PARAMETERS FOR MODEL HONEYCOMB
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
  BULK MODULUS   = <real>
  LAMBDA         = <real>
  TWO MU         = <real>
  #
  # Orthotropic response
  #
  MODULUS_TTTT = <real>
  MODULUS_LLLL = <real>
  MODULUS_WWWW = <real>
  MODULUS_TTLL = <real>
  MODULUS_TTWW = <real>
  MODULUS_LLWW = <real>
  MODULUS_TLTL = <real>
  MODULUS_LWLW = <real>
  MODULUS_WTWT = <real>
  #
  # Material orientation
  #
  TX = <real>
  TY = <real>
  TZ = <real>
  LX = <real>
  LY = <real>
  LZ = <real>
  #
  # Yield behavior
  #
  YIELD_STRESS = <real>
  A1           = <real>
  B1           = <real>
  C1           = <real>
```

(continues on next page)

(continued from previous page)

```
  A2            = <real>
  B2            = <real>
  C2            = <real>
  A3            = <real>
  B3            = <real>
  C3            = <real>

  TS  = <real>
  LS  = <real>
  WS  = <real>
  TLS = <real>
  LWS = <real>
  WTS = <real>

  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.

$$\begin{Bmatrix} \dot{\sigma}_{TT} \\ \dot{\sigma}_{LL} \\ \dot{\sigma}_{WW} \\ \dot{\sigma}_{TL} \\ \dot{\sigma}_{LW} \\ \dot{\sigma}_{WT} \end{Bmatrix} = \begin{bmatrix} E_{TTTT} & E_{TTLL} & E_{TTWW} & 0 & 0 & 0 \\ E_{TTLL} & E_{LLLL} & E_{LLWW} & 0 & 0 & 0 \\ E_{TTWW} & E_{LLWW} & E_{WWWW} & 0 & 0 & 0 \\ 0 & 0 & 0 & E_{TLTL} & 0 & 0 \\ 0 & 0 & 0 & 0 & E_{LWLW} & 0 \\ 0 & 0 & 0 & 0 & 0 & E_{WTWT} \end{bmatrix} \begin{Bmatrix} \dot{d}_{TT} \\ \dot{d}_{LL} \\ \dot{d}_{WW} \\ \dot{d}_{TL} \\ \dot{d}_{LW} \\ \dot{d}_{WT} \end{Bmatrix}$$

Output variables available for this model are listed in Table 15.5.

Table 15.5 State Variables for HONEYCOMB Model

Index	Name	Description
1	CRUSH	minimum volume ratio
2	EQDOT	effective strain rate
3	RMULT	rate multiplier
5	ITER	iterations
6	EVOL	volumetric strain

15.17 Viscoplastic Foam

Please consult the LAMÉ manual.

15.18 Thermo EP Power Model

Output variables available for this model are listed in Table 15.6.

Table 15.6 State Variables for THERMO EP POWER Model

Index	Name	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

15.19 Thermo EP Power Weld Model

Output variables available for this model are listed in Table 15.7.

Table 15.7 State Variables for THERMO EP POWER WELD Model

Index	Name	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
9	WELD_FLAG	

15.20 NLVE 3D Orthotropic Model

```
BEGIN PARAMETERS FOR MODEL NLVE_3D_ORTHOTROPIC
  #
  # Elastic constants
  #
  YOUNGS MODULUS = <real>
  POISSONS RATIO = <real>
  SHEAR MODULUS  = <real>
  BULK MODULUS   = <real>
  LAMBDA         = <real>
  TWO MU         = <real>
  #
  # Material coordinates system definition
  #
  COORDINATE SYSTEM             = <string> coordinate_system_name
  DIRECTION FOR ROTATION        = <real> 1|2|3
  ALPHA                         = <real> (degrees)
  SECOND DIRECTION FOR ROTATION = <real> 1|2|3
  SECOND ALPHA                  = <real> (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
```

(continues on next page)

(continued from previous page)

```
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
```

(continues on next page)

(continued from previous page)

```
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
```

(continues on next page)

(continued from previous page)

```
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
```

(continues on next page)

(continued from previous page)

```
  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.

15.21 Other Undocumented Material Models

For a listing of other material models that exist in Sierra/SM see Table 15.8. Support for use of these models is limited.

Table 15.8 Other Material Models Available (Undocumented)

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	

16 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 16.1 describes the intrinsic cohesive zone models in development, whereas Section 16.2 describes the extrinsic models.

16.1 Intrinsic Models

16.1.1 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 [20]]. Note that the current implementation of the MDGc CZM differs slightly from that described in reference [[20]] 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 [[20], 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 16.1.

Table 16.1 State Variables for MDGc CZM (Section 16.1.1)

Name	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

16.2 Extrinsic Models

16.2.1 Tvergaard Hutchinson

This model is an extension of the trapezoidal traction-separation model proposed by Tvergaard and Hutchinson [22] 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 Fig. 16.1. In Fig. 16.1, 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:

$$0 \leq \lambda_1 \leq \lambda_2 \leq \lambda_c = 1, \quad \hat{\sigma} > 0.$$

Finally, as shown in Fig. 16.1, for $\lambda > \lambda_1$ unloading may be assumed towards the origin.

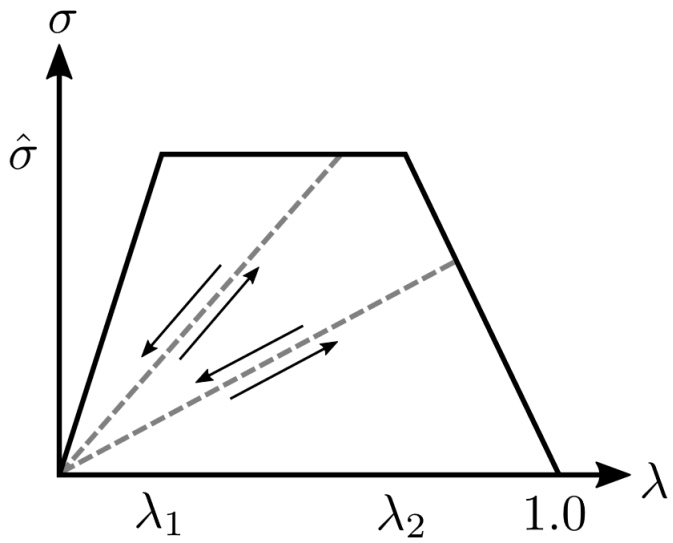


Fig. 16.1 The effective traction-separation model following Tvergaard and Hutchinson.

Assuming a loading condition ($\lambda > 0,\ \dot{\lambda} > 0$), the slope of the effective traction-separation model is evaluated as follows

$$\hat{t}' = \begin{cases} \hat{\sigma}/\lambda_1, & \lambda \in [0,\ \lambda_1) \\ \hat{\sigma}/\lambda, & \lambda \in [\lambda_1, \lambda_2) \\ \hat{\sigma}(1-\lambda)/(\lambda(1-\lambda_2)), & \lambda \in [\lambda_2, \lambda_c) \\ 0, & \lambda \geq \lambda_c \end{cases}$$

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, $r = \delta_{cn}/\delta_{ct}$
- 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:

$$\begin{aligned} \tilde{\sigma} &= \sqrt{(t_n r)^2 + (t_t^1)^2 + (t_t^2)^2}, \\ \hat{\sigma} &= \max(\hat{\sigma}, \tilde{\sigma}), \\ \tilde{\lambda}_t^i &= \frac{t_t^i \lambda_1}{\hat{\sigma} r}, \ i = 1, 2, \\ \tilde{\lambda}_n &= \frac{t_n \lambda_n}{\hat{\sigma}}. \end{aligned}$$

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>
```

(continues on next page)

(continued from previous page)

```
  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.

16.2.2 Thouless Parmigiani

This model is an extension of the Tvergaard Hutchinson effective traction-separation model described in Section 16.2.1, 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]
```

This page left blank

17 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.

This page left blank

18 Phase Field Fracture

A phase-field fracture model has been implemented as a means of providing a mesh-convergent failure modeling capability. Usage of the phase-field fracture capability involves two primary components: (1) using a specific phase-field material model (Phase Field F^eF^p) and (2) establishing a solver to determine the phase field update. The phase-field material model is implemented in the LAMÉ material library and includes all model form choices and model input parameters; this is documented in the LAMÉ manual. The phase-field solver is implemented in Sierra/SM (reaction-diffusion solver) and is documented in this chapter. This includes documentation of the reaction-diffusion solver itself, as well as the control reaction diffusion block, which is used in implicit analyses. Additional usage and debugging guidelines are provided.

18.1 Reaction Diffusion Solver

The phase-field model solves the phase-field evolution using a reaction-diffusion solver implemented in Sierra/SM. The reaction-diffusion command block is as follows:

```
BEGIN REACTION DIFFUSION rxndiffname
  BLOCK = <string list>block_names
  REMOVE BLOCK = <string list>remove_block_names
  INCLUDE ALL BLOCKS

  ASSEMBLY = <string list>assembly_names
  REMOVE ASSEMBLY = <string list>remove_assembly_names

  ACTIVE PERIODS = <string list> active_period_names
  INACTIVE PERIODS = <string list> inactive_period_names

  PHASE FIELD BOUND CONSTRAINTS

  BEGIN PRESCRIBED FIELD
    SURFACE = <string>surf_name
    NODE SET|NODESET = <string>node_set_name
    FUNCTION = <string>func_name
  END
  BEGIN PRESCRIBED FLUX
    SURFACE = <string>surf_name
    NODE SET|NODESET = <string>node_set_name
    FUNCTION = <string>func_name
  END

  GRADIENT CONFIGURATION = MODEL|CURRENT(MODEL)
  USE FINITE ELEMENT MODEL = <string>model_name
```

(continues on next page)

(continued from previous page)

```
      [Model Coordinates Are <string> nodal_variable_name]

  SOLVE AT INITIALIZATION = OFF|ON(OFF)
  SOLVE STEP INCREMENT = <integer>solv_step_incr(1)
  SOLVE TIME INCREMENT = <real>solv_time_incr

  EQUATION SYSTEM = LINEAR|NONLINEAR(LINEAR)
  USE LINEAR SOLVER = FALSE|TRUE(TRUE)

  BEGIN SOLVER OPTIONS
    ACCEPTABLE RESIDUAL = <real>accept_resid(1.0e-14)
    ACCEPTABLE RELATIVE RESIDUAL = <real>accept_rel_resid(1.0e-12)
    TARGET RESIDUAL = <real>target_resid(1.0e-15)
    TARGET RELATIVE RESIDUAL = <real>target_rel_resid(1.0e-13)
    MAXIMUM ITERATIONS = <integer>max_iter(1000)
    LINE SEARCH BACKTRACK [<real>min_step(1.0e-8) <real>reduction(0.1)
        <real>init_step(1.0)
        BISECTION|CUTBACK|SECANT|TRUST_REGION|CRITICAL_POINT(BISECTION)
        <real>dispinc(0.5) <real>rotinc(0.0) ]
    LINEAR SOLVER = <string>linear_solver_name(DEFAULT_FETI_SOLVER)
    PRECONDITIONER = PROBE|IDENTITY(PROBE)
  END

END
```

The commands regarding `BLOCK`, `ASSEMBLY`, and `ACTIVE PERIODS` are common to Sierra/SM, and allow the user to specify reaction-diffusion solution only on the blocks or time periods of interest.

The `SOLVE AT INITIALIZATION` command provides a way to solve the phase field before any mechanical loading is applied. This is particularly useful when applying a phase boundary condition (e.g. prescribed field, prescribed flux), as the phase field will not be in equilibrium prior to the mechanics solve in the first time-step.

Explicit only

The commands `SOLVE STEP INCREMENT` and `SOLVE TIME INCREMENT` offer a way of decreasing the frequency of reaction diffusion solves, particularly for the use-case involving implicit reaction diffusion solves with explicit time incrementation of the solid mechanics solution. Individually, the two commands define the period (in steps, or in time) between successive reaction-diffusion solves. It is an error to specify both commands simultaneously.

The command `PHASE FIELD BOUND CONSTRAINTS` offers a way of ensuring satisfaction of the phase field bound constraints. Specifically, this command restricts the reaction-diffusion solution to the interval $\phi \in [0, 1]$ and also ensures that the field evolves monotonically $\Delta\phi \leq 0$.

Implementation is done using a bound-constrained conjugate gradient method [23] with modifications around the active set selection.

The commands `BEGIN PRESCRIBED FIELD` and `BEGIN PRESCRIBED FLUX` offer a means to specify Dirichlet or Neumann boundary conditions on the phase field. The prescribed field is commonly used to instantiate the model with an initial damage field, such as a phase field pre-crack. Often this provides for better numerical performance than a meshed sharp crack with no initial phase field. The prescribed flux has not been tested, as it lacks a clear physical motivation. It is recommended to use the `SOLVE AT INITIALIZATION` command with these options to ensure that the phase field is in equilibrium before performing any mechanical solution.

The command `GRADIENT CONFIGURATION` provides the option to compute the phase-field gradient in model or current coordinates. Model coordinates are recommended.

The command `USE FINITE ELEMENT MODEL` provides the user the option to specify the finite element model for the phase field. By default, the same finite element model as the Adagio region is used. Using a different finite element mesh for the reaction-diffusion solve has not been tested and is not recommended.

The commands `EQUATION SYSTEM` and `USE LINEAR SYSTEM` are used to determine whether to solve the reaction-diffusion equation as a linear or nonlinear partial differential equation. For the current Phase Field F^eF^p implementation, the phase-field evolution equation is a linear partial differential equation when the degradation function parameter $\gamma = 0$ (corresponding to $\psi_c = \frac{3G_c}{16\ell}$), so the linear reaction-diffusion solve may be used: `LINEAR`, `TRUE`; this uses the default FETI solver to solve the linear system. For other cases ($\gamma \neq 0$), it is the phase-field evolution equation is a nonlinear partial differential equation, so it is recommended to use the nonlinear reaction-diffusion solver: `NONLINEAR`, `FALSE`; this uses a preconditioned conjugate gradient method to iteratively solve the system.

The `BEGIN SOLVER OPTIONS` block offers options to customize the preconditioned conjugate gradient solve (`EQUATION SYSTEM = NONLINEAR`, `USE LINEAR SYSTEM = FALSE`), with commands that reflect a limited set of the commands in the Sierra/SM `BEGIN CG` block. Two nodal preconditioners are available: identity and probe. The probe preconditioner defines its entries as the quotient of the phase system residual divided by the nodal phase stiffness; the phase stiffness is computed as the derivative of the phase force-internal at that node with respect to phase, estimated using a forward finite difference ($\Delta\phi = -1.0 \cdot 10^{-10}$).

18.2 Control Reaction Diffusion

Implicit only

For implicit analyses, it is recommended to use the alternating minimization strategy to iterate between the phase-field update and displacement solution to ensure that an equilibrium state is reached. This is crucial for temporal convergence for implicit analyses with large time-steps.

Explicit only

For explicit analyses, it is assumed that the time-step is sufficiently small that the phase field is never far from equilibrium, so iteration between the two problems is unnecessary. In fact, given the relative cost of the phase-field solve compared to the explicit integration of the displacement system, alternating minimization would be prohibitively expensive.

The CONTROL REACTION DIFFUSION block within the implicit solver block provides the means to perform this iteration. Using this capability, the solver iterates between the displacement solve and the phase field solve until either convergence is realized or the maximum number of iterations is reached. Convergence is assessed using the displacement system residual immediately after the phase field solve; if the phase field update does not throw the displacement system out of equilibrium, it is said to have converged.

Note: The phase field residual is not explicitly assessed. Implementation of a coupled energy-residual norm is under consideration.

Warning: Due to the additional level of iteration, it is not recommended to use CONTROL REACTION DIFFUSION together with other control iterations such as CONTROL CONTACT, CONTROL STIFFNESS, CONTROL FAILURE, or CONTROL DAMPED SOLVE.

```
BEGIN CONTROL REACTION DIFFUSION
  ACCEPTABLE RELATIVE RESIDUAL = <real>accept_rel_resid
  ACCEPTABLE RESIDUAL = <real>accept_resid
  TARGET RELATIVE RESIDUAL = <real>target_rel_resid
  TARGET RESIDUAL = <real>target_resid

  ITERATION PLOT = <integer>iplot
  ITERATION PLOT OUTPUT BLOCKS = <string_list>plot_blocks

  LEVEL = <integer>control_rxndiff_level(1)

  MAXIMUM ITERATIONS = <integer>max_iter(100)
  MINIMUM ITERATIONS = <integer>min_iter(0)

  REFERENCE = EXTERNAL|INTERNAL|RESIDUAL|BELYTSCHKO|ENERGY(EXTERNAL)
  RESIDUAL NORM TYPE = ALL|ALLTRANSLATION|SCALE_RB_ROTATIONS(ALL)
  RESIDUAL ROUNDOFF TOLERANCE = <real>resid_roundoff_tol
END
```

The commands in the CONTROL REACTION DIFFUSION mirror those found in other

Sierra/SM control blocks.

18.3 Output Fields

This section lists output variables for the phase-field fracture capability.

- Table 18.1 Nodal Variables for Phase-Field Fracture
- Table 18.2 Element Variables for Phase-Field Fracture

Table 18.1 Nodal Variables for Phase-Field Fracture

Name	Type	Comments
`rxndiff_bound_constraint`	`Real`	bound constraint indicator (active if non-zero)
`rxndiff_bound_constraint_prev`	`Real`	previous iteration bound constraint indicator
`rxndiff_gradient_direction`	`Real`	PCG gradient direction
`rxndiff_mass`	`Real`	nodal phase field mass
`rxndiff_rhs`	`Real`	phase field evolution equation residual
`rxndiff_scratch_sol`	`Real`	PCG scratch solution
`rxndiff_search_dir`	`Real`	PCG search direction
`rxndiff_sol`	`Real`	phase field ϕ

Table 18.2 Element Variables for Phase-Field Fracture

Name	Type	Comments
`rxndiff_sol_grad`	`Vector_3D`	phase field gradient $\nabla\phi$
`rxndiff_sol_grad_ip`	`Real`	phase field gradient inner product $\nabla\phi \cdot \nabla\phi$
`rxndiff_sol_intpts`	`Real`	phase field at integration points

18.4 Usage Guidelines

Some usage guidelines for the phase-field fracture capability are provided below:

- It is often convenient to rename the phase-field solution in the results output, e.g. `NODAL RXNDIFF_SOL AS PHASE`.
- When building or debugging simulations, it may be useful to remove the *phase-field* (fracture) aspect from the problem and verify the constitutive response. The F^eF^p (development) and J2 Plasticity (production) models are the nearest approximations to the base constitutive model of Phase Field F^eF^p. The `REACTION DIFFUSION` and `CONTROL REACTION DIFFUSION` blocks should be removed accordingly.
- It is strongly recommended to have the element size be smaller than the phase field length scale ℓ, e.g. half or smaller. The half-thickness of the phase field approximation of a fully developed crack, i.e. from $\phi = 0$ to $\phi = 1$, is approximately 2ℓ. Sufficient resolution of this phase gradient is needed to achieve a converged solution; insufficient resolution, especially having the element size be greater than ℓ, is not faithful to the gradient-regularization approach.
- While the Lorentz degradation function relieves the length scale ℓ of direct meaning as a physical property, restoring it to be a numerical parameter that approximates Griffith

fracture as $\ell \to 0$, there are practical recommendations for setting ℓ: - Due to the mesh resolution requirements, an analyst would prefer to set ℓ as large as possible to enable using a coarse mesh. - The degradation function convexity limit ($\gamma \geq -1/3$) establishes a bound that $\ell \leq \frac{9}{32}\frac{G_c}{\psi_c}$. - The length scale should be small compared to features of the structure and smaller than the plastic zone size $r_p \approx \frac{1}{3\pi}\frac{EG_c}{(1-\nu^2)\sigma_y^2}$. - If possible, setting $\ell = \frac{3}{16}\frac{G_c}{\psi_c}$ sets the degradation function parameter $\gamma = 0$, enabling the use of the linear phase field solver, which is often faster than the iterative nonlinear preconditioned conjugate gradient solver used for the nonlinear system.

- The phase field solve adds expense to the simulation, especially when iterated over (control reaction diffusion for implicit integration) or solved frequently (explicit integration). In an effort to minimize this cost, it is recommended to use the Phase Field F^eF^p model and apply the `REACTION DIFFUSION` block judiciously to specific regions where failure is expected. The F^eF^p or J2 Plasticity models are recommended substitutes for consitutive response in regions without damage.
- Be sure to select the appropriate phase-field solver (`EQUATION SYSTEM` and `USE LINEAR SOLVER` pair) that reflects the linearity/non-linearity of the Phase Field F^eF^p material model. If $\gamma = 0$, $\psi_c = \frac{3G_c}{16\ell}$ (default), then the linear solver can be used. Otherwise, the nonlinear preconditioned conjugate gradient solver must be used.
- When using phase field boundary conditions, such as pre-cracks, be sure to activate `SOLVE AT INITIALIZATION`.
- When using explicit time integration: - Consider using `SOLVE STEP INCREMENT` or `SOLVE TIME INCREMENT` to balance accuracy and computational expense.
- When using implicit time integration: - It is recommended to use the `CONTROL REACTION DIFFUSION` block to ensure that the phase field and displacement solutions are in equilibrium. Note that this adds a level of iteration, causing a non-trivial computation time increase. Adaptive time-stepping is also useful with control reaction diffusion, as it allows the solver to take smaller steps if converged equilibrium is not reached. - It is not recommended to use the `CONTROL REACTION DIFFUSION` block together with other control blocks, such as `CONTROL CONTACT`. The computational expense of two levels of iteration is considerable, even on simple problems. If implicit contact must be used, the contact iterations alone may be sufficient to achieve equilibrium with the phase field problem, but this has not been investigated and is not guaranteed. - Convergence of the alternating minimization approach (`CONTROL REACTION DIFFUSION`) becomes more difficult as the increment of phase field grows. Quick, unstable crack propagation would be such a case. Accordingly, the use of adaptive timestepping or implicit dynamics may aid convergence. Similarly, for notched or pre-cracked geometries, adding an initial phase boundary condition may aid solver performance.
- In some problems, removal of highly-damaged elements (i.e. via element death) may be useful to prevent element inversion and maintain explicit time-step size.
- The usage of `PHASE FIELD BOUND CONSTRAINTS` has been shown to improve phase field solutions, especially when fields with values outside the natural bounds ($[0, 1]$) or

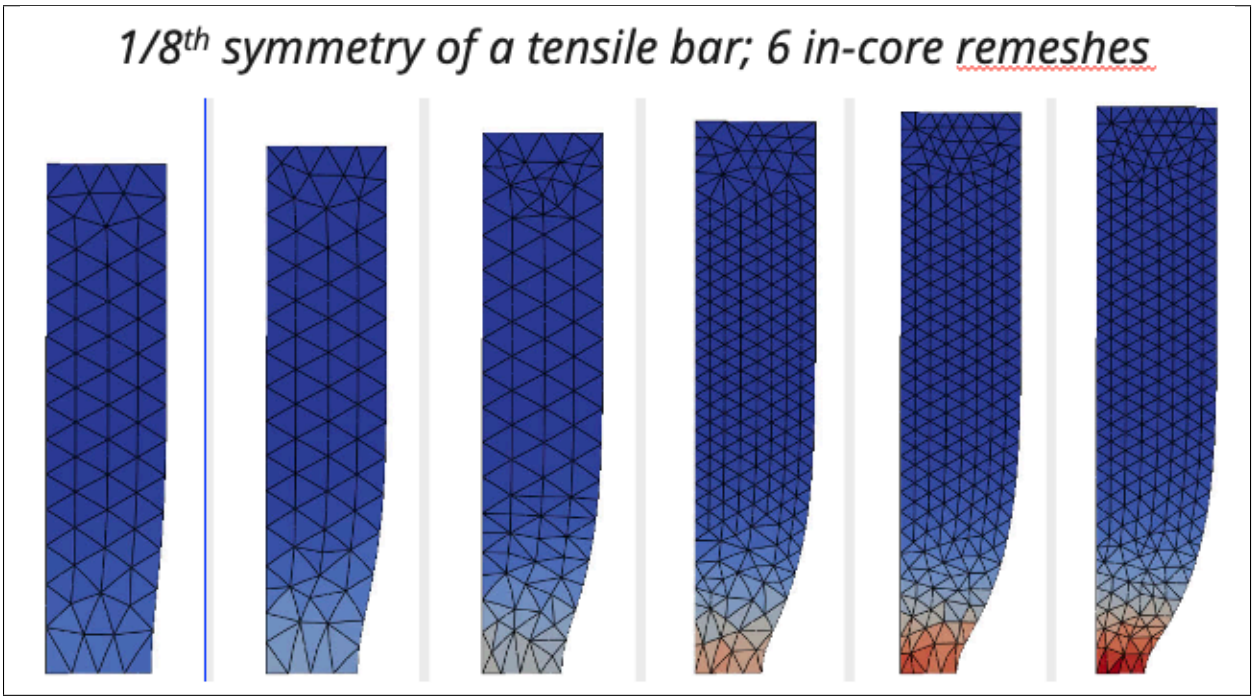
damage reversion have been observed. The bound constraint enforcement has been observed to impact the convergence of the preconditioned conjugate gradient solver, however.

This page left blank

19 Remeshing

The in-core remeshing capability within Sierra/SM allows for dynamic mesh adaption, refinement, and coarsening during simulations. This enables users to enhance mesh quality in regions of interest, represent complex geometries and large deformations, improve convergence rates, and reduce computational costs.

This section describes the setup and execution of in-core remeshing, including current limitations and best practices for selecting remeshing criteria for common use cases.



19.1 Usage

In order to enable remeshing in your simulation, a "BEGIN REMESHING" command block must be invoked. This command block is placed within the Region scope of the Sierra/SM input file. The available commands within the remeshing command block are displayed below:

```
BEGIN REMESHING <string>remesh_name
  #
  # block set commands
  BLOCK = <string list>block_names
  ASSEMBLY = <string list>assembly_names
  INCLUDE ALL BLOCKS
  REMOVE BLOCK = <string list>block_names
  CRITERION IS ELEMENT VALUE OF <string>var_name <|<=|=|>=|>␣
 ↪<real>tolerance
```

```
END REMESHING
```

As an example, consider the below remeshing syntax:

```
BEGIN REMESHING remesh1
  BLOCK = BLOCK_1
  CRITERION IS ELEMENT VALUE OF NORMALIZED_INRADIUS < 1.0e-1
END REMESHING
```

If this syntax is provided, the code will examine all elements within BLOCK_1 to check if any elements have a value of NORMALIZED_INRADIUS < 1.0E-1. If an element is found to meet that criterion, remeshing is performed.

Remeshing subsets of the mesh is not currently supported.

Although the element criteria is only evaluated over block_1, a new mesh can not currently be created for just block_1. Therefore, the entire mesh will be examined and potentially modified if an element meets the remeshing criteria. This capability is currently being developed.

In-Core Remeshing is currently only supported for Composite Tet10 Elements

“**********

---

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 Sierra/SM User 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 Sierra/SM User Manual section on Element Block Parameters).

Note that elements may also be converted to particles via element death (consult Sierra/SM User Manual 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 Sierra/SM User Manual
  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 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
```

20.5 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 integration 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 replacing them with the pressure from the central integration point. Note that the locked pressures are replaced during element calculations, not inside the constitutive model. This means that material models and element death criteria that fail or accumulate damage based on pressure may be adversely affected by this deviatoric pressure locking. For this reason, the selective-deviatoric (SD) element is generally preferred for material failure analyses. The SD element calculates a single 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.

Explicit only

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] ACI318-08: building code requirements for structural concrete and commentary. Farmington Hills, MI. American Concrete Institute. 2008.

[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. URL: http://infoserve.sandia.gov/sand_doc/1972/720883.pdf.

[3] D.J. Bammann. Modeling temperature and strain dependent large deformations in metals. *Applied Mechanics Reviews*, 43(5):312–319, 1990. doi:10.1115/1.3120834.

[4] D.J. Bammann, M.L. Chiesa, M.F. Horstemeyer, and L.E. Weingarten. Failure in ductile materials using finite element methods. In N. Jones and T. Wierzbicki, editors, *Structural Crashworthiness and Failure*, 1–53. London, 1993. Elsevier Applied Science.

[5] D.J. Bammann, M.L. Chiesa, and G.C. Johnson. Modelling large deformation and failure in manufacturing processes. In T. Tatsumi, E. Watanabe, and T. Kambe, editors, *Proceedings of the 19th International Congress of Theoretical and Applied Mechanics*, 359–376. Amsterdam, 1997. Elsevier Science Publishers.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] S.A. English. A 3D orthotropic strain-rate dependent elastic damage material model. Technical Report SAND2014-17469, Sandia National Laboratories, Albuquerque, NM, 2014. URL: http://infoserve.sandia.gov/sand_doc/2014/17469.pdf.

[12] 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.

[13] 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.

[14] Jacob J. Koester and Jiun-Shyan Chen. Conforming window functions for meshfree methods. *Computer Methods in Applied Mechanics and Engineering*, 347:588 – 621, 2019. doi:https://doi.org/10.1016/j.cma.2018.12.042.

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

[16] 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.

[17] D. M. Neto, M. C. Oliveira, L. F. Menezes, and J. L. Alves. Improving nagata patch interpolation applied for tool surface description in sheet metal forming simulation. *Computer-Aided Design*, 45(3):639–656, 2013.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] Edwin A.H. Vollebregt. The bound-constrained conjugate gradient method for non-negative matrices. *Journal of Optimization Theory and Applications*, 162(3):931–953, 2014.

[24] 3D Systems, Inc. SLC File Specification. 1994.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.