Tahoe
User Guide

Input Version 3.4.1

last updated August 11, 2011

Abstract

Tahoe is a general purpose simulation code incorporating finite element, meshfree methods, and atomistic methods. The code is capable of execution on parallel platforms and includes interfaces to third-party libraries that can solve linear systems of equations in parallel. This document is intended as a guide for users. It does not contain the detailed descriptions of the code required by developers. This guide is a work in progress. Please send questions and bug reports to `tahoe-help@sandia.gov`.

Contents

List of Figures

List of Tables

1 Introduction

Tahoe is a general purpose simulation code. It originally included only finite element methods, but has been extended to include meshfree methods, atomistics, and cohesive surface approaches to modeling fracture. Aside from interfaces for adding element formulations and constitutive models, Tahoe also supports specialized boundary conditions such as K-fields and rigid boundaries of various geometries. The code has also been extended parallel execution.

The purpose of this guide is to explain how to run Tahoe in its various modes. Tahoe execution is driven by a parameters file. The layout of this file is described in Section 4. The parameters in the file are scanned sequentially. The interceding chapters provide some additional information about the various formulations available for execution. This guide is not intended as a guide of developers. Although some references to specific classes do appear throughout this guide, the structure of the code itself is not explained in any detail. Additional information about the code itself can be obtained from the Tahoe Development Server at

<`http://tahoe.ca.sandia.gov`>

If you are interested in getting access to the code repository for Tahoe, please contact

`tahoe-help@sandia.gov`

This guide contains hypertext links. These can be used to navigate the guide if your document viewer support links.

1.1 Conventions

[To do: describe conventions used for this guide.]

1.2 Overview of Tahoe components

Although this guide will not discuss the implementation of Tahoe in any detail, some discussion of its internal organization is included to provide a rationale for the organization of input parameters. Internally, Tahoe is organized into the components shown in Figure 1.1. Data and parameters are "owned" by specific components of the code. A brief description of the components follows. For further information on a given capability, identify the component whose scope covers the capability and seek more information in the later sections of this guide.

Tahoe is a hierarchical collection of components, or managers. Each manager is responsible for specific data and functionality. The configuration of each manager is set in the parameters input file. A brief description of each follows.

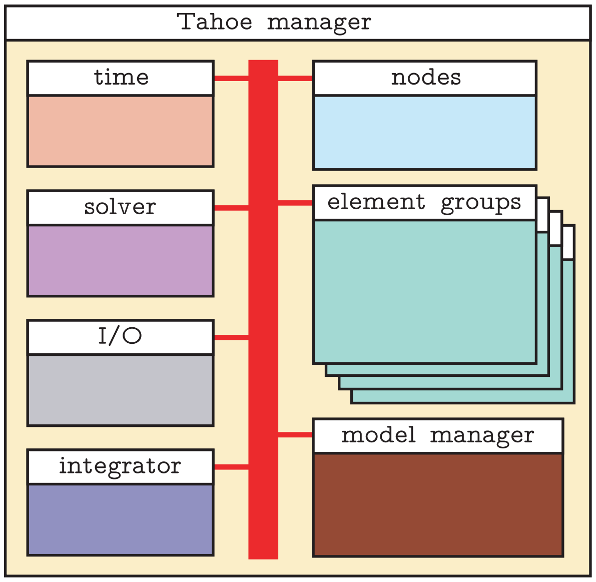


Figure 1.1: Division of functionality and data within Tahoe.

1.2.1 Tahoe manager

The `Tahoe manager` sits at the top of the manager hierarchy. It is responsible of for first configuring other managers during the initialization phase of a simulation, checking for consistency and compatibility of input parameters, and second serving as the back bone for communication between the managers during the solution phase. Additionally, the `Tahoe manager` is the "last hope" for trapping and correcting exceptional conditions that occur during the solution phase.

1.2.2 Time manager

The `time manager` is responsible for all parameters related to the solution progress variable, or time. This includes information about the initial time, final time, and time step. Boundary conditions may be specified as functions of time. These schedules are specified to the `time manager`. With solution procedures for which it is supported, the `time manager` handles automatic time step control.

1.2.3 Controller

Tahoe supports a number of time integration schemes that are handled by the `controller`. There are no input parameters for directly affecting the configuration of the `controller`. The time integration scheme is determined by parameters specified to the `Tahoe manager` and the `time manager`.

1.2.4 Node manager

The `node manager` is responsible for handling the fields of unknowns, i.e. $\mathbf{u}(\mathbf{X}, t)$. Depending on the type of analysis, this data includes the field variables and their derivatives at the current time step t_n, the last time t_{n-1}, and at other times t_{n-N} for recovery from exceptional conditions that occur during the solution phase. The `node manager` handles essential boundary conditions and specification of the externally applied conjugate forces.

1.2.5 Element groups

An `element group` in `Tahoe` generally refers to the implementation of any formulation that translates nodal field variables to their conjugate forces. For most simulations, a majority of the parameters provided to `Tahoe` are concerned with the `element groups`. For finite element types, these parameters would include specifications of the element geometry and integration scheme for the element-level variational equations. The specification of constitutive parameters is embedded in the element data although the types of constitutive models will depend on the particular element. The elements supported by `Tahoe` are listed in Table 7.1.

1.2.6 I/O manager

The `I/O manager` is responsible for supporting the various input formats for geometry information and the various output formats for results data. There are no input parameters for directly affecting the `I/O manager`. I/O file formats are specified in the parameters for the `Tahoe manager`. A description of supported file formats is given in Section 10.

1.2.7 Solver

The `solver` is responsible bringing the system to equilibrium

$$\mathbf{F}^{\mathrm{int}}(\mathbf{u}) = \mathbf{F}^{\mathrm{ext}}(t)\,.$$

Depending on the type of analysis, the `solver` may require specifications of the nonlinear solution algorithm and convergence tolerances. Parameters for solvers for linear system of equations is embedded in the `solver` data, including installed third-party libraries.

1.3 Supported platforms

[To do: provide table of supported architectures, compilers, and extensions, i.e., MPI.]

2 Execution procedures

Tahoe has be ported to a number of platforms. The code can be executed in parallel as well as in serial. Furthermore, Tahoe runs in several modes that have been added to support execution in parallel. These modes are listed below. Only the Parallel Analysis mode actually runs in parallel, the other modes run in serial.

(2.1) Serial analysis

(2.2) Parallel analysis

(2.3) Domain decomposition

(2.4) Joining results data

The procedure for running in serial and parallel has been made as similar as possible. The only differences arise from machine-dependent limitations. The parameters file and geometry file that are needed to define an analysis are identical for both modes of analysis. Some additional files are needed to run parallel analysis, but Tahoe generates these for the user. The user must only be aware of which files are needed on the machine for each mode of operation at the time of execution. The following sections explain how to execute Tahoe with the following command line options:

```
tahoe
        -f [job name].in
        -decomp
        -join
        -split_io
```

2.1 Serial analysis

For running serially, the user must have:

[job name]`.in`	parameters file
[geometry name]`.exo`	geometry file (ExodusII [33])

The file extensions are not interpreted by Tahoe, so these may be selected arbitrarily. The resulting output files will be:

[job name]`.out`	"echo" file of parameters file
[job name]`.io`*[j]*`.exo`	results data (ExodusII [33])

The results data have labels j, where $j = 0, \ldots, (\textit{number of output groups} - 1)$. The number of output groups is determined by the type of analysis. For the simplest case, each element group will produce one output group, and the sequence of output groups will be the same as the sequence the element groups are declared in the parameters file. Each results file has only one element block corresponding to the connectivities for the element group. If multiple element blocks are used to define the connectivities for a given element group, these will be combined into a single element block for output. Serial analysis can also be done by running a parallel executable with the number of processors specified as one.

2.2 Parallel analysis

The procedure for launching parallel executables varies depending on the user environment. In the examples that follow, we assume parallel executables are launched using the `mpirun` command:

`% mpirun -np` [*number of processors*] `tahoe`

Users should substitute the appropriate command, such as `dmpirun` or `yod`, depending on the environment.

The procedure for running Tahoe in parallel varies depending on whether the machine has "limited" memory with respect to the problem size. A machine is considered to have "limited" memory with respect to a given problem if the entire model from the geometry definition cannot be stored in memory at the same time. On machines that are not memory limited, execution of Tahoe is *identical* whether running on serial or in parallel, aside from any special procedure needed to launch a parallel job. For this case, no additional steps are required for the user in order to pre- and post-process the model definition. With limited memory machines, a non-limited machine is needed in order to decompose the model geometry and join the results data after execution. Domain decomposition and result file joining is discussed in the next two sections.

The identical input files (parameters and geometry) are needed to run in parallel as are needed to run in serial.

[*job name*]`.in`	parameters file
[*geometry name*]`.exo`	geometry file (ExodusII [33])

The first thing that the program does is look to see if the required decomposition files already exist. Tahoe creates a domain decomposition based on the connectivities seen at run time. If the geometry file contains blocks of elements that are not referenced in the parameters file, these blocks are not considered during decomposition. Additionally, meshfree methods generate connectivities at run time the depend on parameters defining the shape functions. Tahoe does not keep track of which values in the parameters file affect the domain decomposition. If any of these parameters are changed, the user must delete any of the corresponding decomposition files. The decomposition files are:

[*geometry name*]`.io.exo`	global geometry file
[*geometry name*]`.n`[*M*]`.io.map`	output map
[*geometry name*]`.io.ID`	block ID's per output group
[*geometry name*]`.n`[*M*]`.part`[*i*]	decomposition data file
[*geometry name*]`.n`[*M*]`.p`[*i*]`.exo`	partial geometry file

In the file names above, M is the total number of partitions, while $i = 0, \ldots, (M-1)$. If any decomposition files are not found, Tahoe will attempt to create all of them. If Tahoe is launched on a non-limited memory machine, mesh decomposition is done serially on the processor with rank 0 before starting the analysis phase of execution in parallel. If the decomposition files already exist, the decomposition phase is not repeated. Tahoe produces the following files during execution:

[*job name*]`.p`[*i*]`.out`	"echo" file of parameters file
[*job name*]`.p`[*i*]`.log`	message passing log file
`console`[*i*]	redirected console output
[*job name*]`.io`[*j*]`.exo`	results data (ExodusII [33])

where $j = 0, \ldots, (\textit{number of output groups} - 1)$. The output from the standard output streams is not redirected to a file for the processor with rank 0, that is, no `console0` file is created. Any prompts for use input still appear on the screen. When running a job in the background the user may opt to redirect the screen output to a file, called `console0` for consistency. The following input makes use of the `-f` command line option to specify the name of the parameters file:

```
% mpirun -np [number of processors] tahoe -f [job name].in > console0 &
```

On non-limited memory machines, the output from the distributed output groups is collected on processors as designated in the output map file. Since the output is "joined" at run time, no post-processing of the results data is required. Running Tahoe in this mode does require more memory. Tahoe will leave the output in its decomposed form if launched with the `-split_io` command line option:

```
% mpirun -np [number of processors] tahoe -split_io
```

When running on a limited memory machine, or when specifying the `-split_io` option, Tahoe will generate distributed results files:

[*job name*]`.p`[*i*]`.io`[*j*]`.exo`	distributed results file

where $i = 0, \ldots, (\textit{number of partitions} - 1)$ and $j = 0, \ldots, (\textit{number of output groups} - 1)$. The distributed results files must then be joined in a separate post-processing step to produce the results files that would have been written by a serial execution of the same calculation.

2.3 Domain decomposition

Serial mesh decomposition of a large mesh often requires more memory than some architectures have for each individual compute node. In these cases the user must perform the decomposition and results file joining on a pre/post processor machine that is not memory limited. If Tahoe is launched with the -decomp command line option, it will generated the decomposition files without subsequently performing any analysis:

```
% tahoe -decomp
```

As mentioned about, Tahoe creates a domain decomposition based on the connectivities seen *at run time*. This is a combination of information in the parameters and geometry files. If the geometry file contains blocks of elements that are not referenced in the parameters file, these blocks are not considered during decomposition. Also, Tahoe does not record the values in the parameters files used to generate a given decomposition. If the parameters file changes in any way that would affect the set of “active” elements, or the connectivities active at run time, the user must remove any existing decomposition files. Otherwise, Tahoe will use any existing decomposition files.

The decomposition process is not parallelized. In order to avoid allocating processors that will remain idle, Tahoe will not perform decomposition on a parallel platform if more than one processors is allocated for execution. Therefore, decomposition must be initiated with:

```
% mpirun -np 1 tahoe -decomp
```

After prompting the user for the name of the job file, Tahoe will prompt the user for the number of partitions to create.

2.4 Joining results files

To join distributed output files, the parameters, geometry, decomposition files, and distributed results files are needed. As with the decomposition process, the joining process has not been parallelized. On a multi-processor platform, it must be launched with a single process:

```
% mpirun -np 1 tahoe -join
```

After prompting the user for the name of the job file, Tahoe will prompt the user for the number of partitions in the distributed results data. The number given to Tahoe must be the *total* number of partitions generated from the original model geometry. Under certain circumstances, not all partitions will contribute distributed output to all output groups.

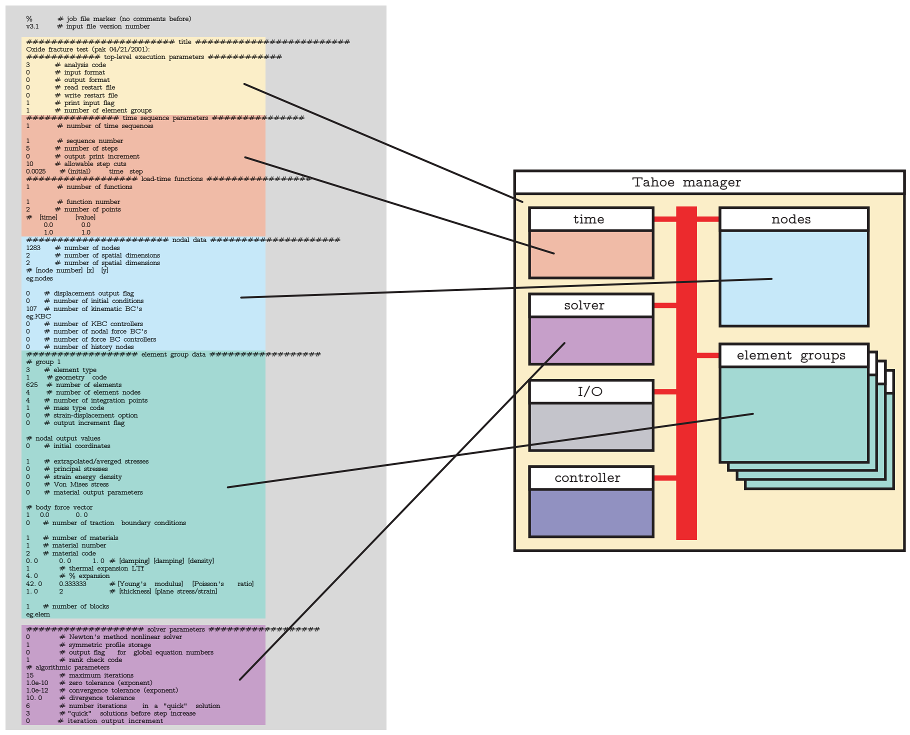
3 Regression tests and benchmarks

[To do: A system of regression tests and benchmarks is in development. The purpose of the system is to allow users to exercise various parts of the Tahoe *to ensure they are functioning properly. Additionally, the tests will provide examples of input parameters for various types of simulations.]*

3.1 Repeatability of tests across platforms

3.2 Repeatability of tests during parallel execution

4 Input file version 3.4.1



```
%       # job file marker (no comments before)
v3.1    # input file version number

######################### title ###########################
Oxide fracture test (pak 04/21/2001):
############ top-level execution parameters ############
3       # analysis code
0       # input format
0       # output format
0       # read restart file
0       # write restart file
1       # print input flag
1       # number of element groups
############### time sequence parameters ################
1        # number of time sequences

1        # sequence number
5       # number of steps
0       # output print increment
10      # allowable step cuts
0.0025    # (initial)      time  step
################### load-time functions ##################
1        # number of functions

1        # function number
2       # number of points
#  [time]      [value]
     0.0         0.0
     1.0         1.0
######################## nodal data ######################
1283    # number of nodes
2       # number of spatial dimensions
2       # number of spatial dimensions
# [node number] [x]  [y]
eg.nodes

0    # displacement output flag
0    # number of initial conditions
107  # number of kinematic BC's
eg.KBC
0    # number of KBC controllers
0    # number of nodal force BC's
0    # number of force BC controllers
0    # number of history nodes
################### element group data ###################
# group 1
3    # element type
1     # geometry  code
625   # number of elements
4     # number of element nodes
4    # number of integration points
1    # mass type code
0    # strain-displacement option
0    # output increment flag

# nodal output values
0    # initial coordinates

1    # extrapolated/averged stresses
0    # principal stresses
0    # strain energy density
0    # Von Mises stress
0    # material output parameters

# body force vector
1   0.0        0. 0
0   # number of traction  boundary conditions

1   # number of materials
1   # material number
2   # material code
0. 0      0. 0      1. 0  # [damping] [damping] [density]
1         # thermal expansion LTf
4. 0      # % expansion
42. 0     0.333333      # [Young's  modulus]  [Poisson's    ratio]
1. 0      2             # [thickness] [plane stress/strain]

1   # number of blocks
eg.elem

#################### solver parameters ###################
0         # Newton's method nonlinear solver
1         # symmetric profile storage
0         # output flag   for  global equation numbers
1         # rank check code
# algorithmic parameters
15        # maximum iterations
1.0e-10   # zero tolerance (exponent)
1.0e-12   # convergence tolerance (exponent)
10. 0     # divergence tolerance
6         # number iterations    in  a  "quick"  solution
3         # "quick"  solutions before step increase
0        # iteration output increment
```

Figure 4.1: Layout of a Tahoe parameters file.

Simulation parameters for Tahoe are specified in white space delineated, plain text files. Input values are separated by white space, but the arrangement of white space is ignored. White space includes blank spaces, tabs, new line characters, and carriage returns. Tahoe uses calls to the platform-specific, standard C/C++ libraries to handle input from text files. Plain text files contain platform-specific line ending characters that may cause compatibility problems when files are transferred between platforms unless some file translation is applied. The # symbol denotes that the remainder of a given line of text should be ignored, or treated as a comment. Values are read from the parameters file sequentially. Therefore, all parameters must be specified, and there are no default values. The only exceptions to these conventions may come in specifications to the third-party libraries used by Tahoe for certain types of analysis.

As discussed in Section 1.2, Tahoe is internally divided into components. Data and parameters are owned by each component. Figure 4.1 shows how the parameters file may be divided into sections of parameters for the different components.

(1) "Global" analysis parameters

(2) Time parameters

(3) Nodal parameters

(4) Element group(s) parameters

(5) Solver parameters

Details of the parameters in each block follow below.

4.1 Global analysis parameters

The format of the global analysis parameters is shown below:

```
%     # job file marker
3.4.1  # input file version marker
# title
[one line of at most 254 characters]
#### analysis parameters
[analysis code]
[input format]
[output format]
[read restart flag]
[write restart increment]
[echo input flag]
```

The first character in a parameters file must be %. The input file version is used by Tahoe to determine the compatibility of the input file with the version of the executable. Tahoe issues a warning if the input file version is older than the current. Developers are encouraged to make changes to the input file format backward compatible; however, backward compatibility cannot be maintained in all cases. Changes to the input file format since version 3.1 are logged in Section 11.2. Values of the *analysis code* are listed in Table 5.1. Table 10.1 lists the values for the *input format*. For input file formats using external geometry files, the code is followed by the relative path to the geometry database, as in:

```
# input format
5    # ExodusII format
[path to geometry database]
```

Table 10.2 lists the values for the *output format*. The *read restart flag* is boolean of either `0` or `1`. A `1` indicates that Tahoe should initialize the system using information in a restart file. In this case, the flag is followed by the relative path to the restart file, as in:

```
# read restart flag
1    [path to restart file]
```

A *write restart increment* of $N > 0$ indicates that Tahoe should write a restart file every N

steps. The *echo input flag* is a boolean flag of either `0` or `1`. A value of `1` causes verbose echoing of input parameters and geometry information.

4.2 Time parameters

The time parameters are grouped into three sections:

`#### time parameters`

`# list of time sequences`

[n_{seq} = *number of time sequences*]

`1` [*parameters for sequence* 1]

⋮

[n_{seq}] [*parameters for sequence* n_{seq}]

`# list of schedule functions`

[n_{scf} = *number of schedule functions*]

`1` [*parameters for schedule function* 1]

⋮

[n_{scf}] [*parameters for schedule function* n_{scf}]

`# time integration parameters`

[*parameter list*]

The numbered sets of time sequence parameters and numbered schedule functions may appear in any order in their respective lists. The format of time sequence parameters is described in Section 4.2.1, while the format of schedule functions in described in Section 4.2.2. Most of the time integration schemes do not have any input parameters. Only the time integration scheme used for implicit elastodynamics (see Table 5.1) requires additional parameters. The HHT-α [17] integrator takes the single parameter $0 \leq \alpha \leq 1$, where $\alpha > 0$ introduces numerical dissipation. Dissipation is not supported for nonlinear analysis.

4.2.1 Time sequence parameters

A time sequence is specified by the following list of parameters:

`# time sequence parameters`

[*number of steps*]

[*output print increment*]

[*maximum number of step cuts*]

[*initial step size*]

If the *number of steps* is specified as zero, Tahoe will only complete the initialization phase of a simulation, reading all values from the parameters file. An *output print increment* specified as $N > 0$ causes results data to be written every N steps of the initial step size. An *output print increment* specified as $N < 0$ produces output every N steps of the current step size. Some analysis times support automatic time step control. The *maximum number of step cuts* controls how many times the initial time step can be reduced by a factor of two before the simulation is aborted.

4.2.2 Schedule functions

Schedule functions are used to specify the time-varying behavior of boundary conditions or other analysis-specific parameters such as the thermal expansion in elastostatic and elastodynamic analysis. A function $f(t)$ is specified with a list of ordered pairs $\{t_i, f(t_i)\}, i = 1, \ldots, n_{pts}$. The function is linearly interpolated between the data provided. The function is extended at zero slope beyond the range of time provided, that is

$$f(t) = \begin{cases} f(t_1) & \text{if } t < t_1, \\ f(t_i) + (t - t_i)\,\frac{f(t_{i+1}) - f(t_i)}{t_{i+1} - t_i} & \text{if } t_i < t < t_{i+1} \text{ for } i = 1, \ldots, n_{pts} - 1, \\ f\left(t_{n_{pts}}\right) & \text{if } t > t_{n_{pts}}. \end{cases}$$

[To do: add figure] An example of the input format for schedule function is shown below:

$$\begin{array}{ll} \multicolumn{2}{l}{[n_{pts} = \textit{number of points on the schedule}]} \\ [t_1] & [f(t_1)] \\ \vdots & \\ \left[t_{n_{pts}}\right] & \left[f\left(t_{n_{pts}}\right)\right] \end{array}$$

The ordered pairs on the schedule function must be specified with time in sequentially increasing order.

4.3 Nodal parameters

The nodal parameters of the input file contains the following sections:

```
#### nodal data
```
[*coordinate data*]
[*list of initial conditions*]
[*list of kinematic boundary conditions*]
[*list of kinematic boundary condition controllers*]
[*list of force boundary conditions*]
[*list of force boundary condition controllers*]
[*list of history node specifications*]

The *coordinate data* only appears in the nodal section if geometry format TahoeI is specified in the global parameters section, described in Section 4.1. For all other geometry file formats listed in Table 10.1, the nodal coordinate data resides in an external file, and this section of the nodal parameters is omitted. The format for all of the lists specified in this section is similar. The list is initiated with its length and followed by the entries in the list if the length is nonzero. The *list of initial conditions* is either empty

```
0 # initial conditions
```

or contains a list specified as

`# initial conditions`
[n_{ic} = *number of initial conditions*]
[*parameters for initial condition* 1]
⋮
[*parameters for initial condition* n_{ic}]

The format of initial condition specifications is described in Section 6.1. The *list of kinematic boundary conditions*, or essential boundary conditions, is either empty

```
0 # kinematic boundary conditions
```

or contains a list specified as

`# kinematic boundary conditions`
[n_{kbc} = *number of kinematic boundary conditions*]
[*parameters for kinematic boundary condition* 1]
⋮
[*parameters for kinematic boundary condition* n_{kbc}]

The format of kinematic boundary condition specifications is described in Section 6.2. Kinematic boundary conditions allow specification of the primal variables along the cartesian axes as a function of time. Kinematic boundary condition controllers allow specification the field at groups of nodes to follow specific functional forms, such as the displacements surrounding a crack tip based on the asymptotic K-field solution from isotropic linear elasticity. The *list of kinematic boundary condition controllers* is either empty

```
0 # kinematic boundary condition controllers
```

or contains a list specified as

```
# kinematic boundary condition controllers
```
[n_{kctl} = *number of kinematic boundary conditions*]
[*parameters for kinematic boundary condition controller* 1]
$\vdots$
[*parameters for kinematic boundary condition controller* n_{kctl}]

The format of kinematic boundary condition controllers specifications is described in Section 6.3. The *list of force boundary conditions* is either empty

```
0 # nodal force boundary conditions
```

or contains a list specified as

```
# nodal force boundary conditions
```
[n_{fbc} = *number of nodal force boundary conditions*]
[*parameters for force boundary condition* 1]
$\vdots$
[*parameters for force boundary condition* n_{fbc}]

The format of nodal force boundary condition specifications is described in Section 6.4. The *list of force boundary condition controllers* is either empty

```
0 # force boundary condition controllers
```

or contains a list specified as

```
# force boundary condition controllers
```
[n_{fctl} = *number of force boundary condition controllers*]
[*parameters for force controller* 1]
$\vdots$
[*parameters for force controller* n_{fctl}]

The format of nodal force boundary condition controllers specifications is described in Section 6.5. Force boundary condition controllers implement a number of "rigid" objects that generate forces on nodes through contact. For example, the rigid sphere controllers can be used to represent an indenter with a motion specified as a function of time.

Nodes, or sets of nodes, can be specified to produce more detailed output. The format of history node output is described in Section 10.2.7. The *list of history node specifications* is either empty

```
0 # number of history nodes
```

or contains a list specified as

```
# history nodes
```
[n_{hst} = *number of history node ID's*]
[*node* ID_1] ... [*node* $ID_{n_{hst}}$]

The *node ID* differs depending on geometry file format (see Table 10.1). For TahoeI format, the *node ID* is simply the node number. Some of the geometry database formats supported by Tahoe allow nodes to be grouped into sets identified by an *ID*. With these database formats, the *node ID* corresponds to the node set *ID*.

4.4 Element parameters

4.5 Solver parameters

4.6 Batch file format

5 Analysis types

Table 5.1 shows a list of analysis available in Tahoe.

Table 5.1: This is the list of types.

analysis type	**linear**	**nonlinear**
elastostatic	1	3
implicit elastodynamic	2	4
explicit elastodynamic	6	7
steady-state diffusion	19	N/A
transient diffusion	20	N/A

5.1 Analysis types

Linear analysis implies that the governing equations are linear in the unknown variables over a given time increment. Nonlinear analysis implies that some iterative procedure is required to determine the solution in the unknown variables. Most cases require the nonlinear solution procedures. Specifying a nonlinear analysis type for a linear problem will produce a solution after a single iteration of the solution procedure.

Whether linear or nonlinear, the solution procedure ultimately results in a linear system of equation. A number of different storage schemes are available for the global matrix associated with this linear system. The different types of matrix storage are listed in Table 5.2. The choice of matrix depends on the analysis type as is outlined below. Moreover, all matrix types may be used for serial execution, while parallel execution must use either the diagonal, Aztec [18], or SPOOLES [4] matrices.

Table 5.2: Matrix types for the global system of equations.

matrix type	**code**
diagonal	0
profile	1
full	2
Aztec [18]	3
SPOOLES [4]	5

5.1.1 Elastostatic analysis

This analysis can be used with profile, full, Aztec, or SPOOLES global matrix types.

5.1.2 Implicit elastodynamic analysis

This analysis uses HHT-α time integration developed by Hilber *et al.* [17] and can be used with profile, full, Aztec, or SPOOLES global matrix types.

5.1.3 Explicit elastodynamic analysis

This analysis uses the explicit, central-difference time integration scheme from the classic Newmark family of methods. In order to make this method "matrix-free", the diagonal matrix storage should be used.

5.1.4 Steady-state and transient diffusion

Transient head conduction uses a trapezoidal time integration scheme, and can be used with profile, full, Aztec, or SPOOLES global matrix types.

5.2 Solution techniques

5.2.1 Linear

The linear solution technique is used with all of the linear analyses and with explicit dynamic analysis. Explicit dynamic analysis may involve large, nonlinear kinematics of deformation, nonlinear constitutive response, and contact; however, as a result of the time discretization, the incremental accelerations are linearly related to the driving forces over a single time increment.

5.2.2 Nonlinear

Nonlinear solution techniques are used with all nonlinear analyses except with nonlinear explicit dynamics analysis. Table 5.3 shows a list of nonlinear solution techniques supported. As mentioned above, the nonlinear solution procedures iteratively search for the solution until some convergence tolerance is satisfied. In general, a solution procedure involves determining two aspects of the search for the solution. The procedure must determine the "length" and "direction" of the update to the current best guess of the solution. The methods listed in Table 5.3 differ in how they determine these two parameters. The various nonlinear solution methods use two criteria for assessing convergence based on the norm of the residual forces. The first is an "absolute" tolerance while the second is a "relative" tolerance. The "absolute" tolerance is an indication of "small" in terms of the numerical precision displayed for a given calculation. Depending on the platform, the various formulations used throughout a calculation, and details of the implementation, the level of numerical "noise" in a calculation may limit how close to absolute zero the solution procedure is able to drive the norm of the residual. The second tolerance examines the "relative" error. This compares the norm of the residual at the end of an iteration with the norm of the residual at the first iteration. If one examines the console output during a simulation, the absolute error is written to the output after incremental boundary conditions are applied followed by the relative error after each iteration. The solution procedure stops if either condition is satisfied.

Table 5.3: Nonlinear solution techniques.

solution method	**code**
standard Newton	0
initial tangent	1
modified Newton	2
dynamic relaxation [46]	3
Newton with line search	4
preconditioned nonlinear conjugate gradient	5
interactive Newton	6

All of the nonlinear solution techniques automatically adjust the time, or load, increment depending on the behavior of the iterative solution procedure. The solvers have user-adjustable parameters for determining whether the increment should increase or decrease. A cut in the increment is triggered if the number of iterations exceeds to the user-defined value or if the "relative" error exceeds a user-defined value. Additionally, the increment is decreased if "overload" conditions are detected in calculations anywhere in the code. Once the increment has been cut, Tahoe begins to monitor two parameters in combination to determine if the increment should increased. The user can prescribe the number of iterations in a "quick" solution and the number of sequential "quick" solutions that must pass before the increment is increased. An increment adjustment changes the step size by a factor of two.

5.2.2.1 Standard Newton Newton's method is a second order optimization technique that makes use of a tangent matrix to determine the update vector. This solver reforms a new tangent matrix at every iteration, and does not rescale the update vector. The convergence behavior of Newton methods is known to be poor "far" from the solution, but the method does display locally quadratic convergence "near" the solution.

5.2.2.2 Initial tangent This solver differs from the standard Newton in that a new tangent matrix is calculated only once per time, or load, increment. The tangent matrix is formed for the initial iteration and is subsequently reused. This approach results in slower rates of convergence, but may result in faster solution times since the number of times the tangent matrix is formed and factorized is reduced. This method assumes that the matrix type selected allows the factorized matrix to be reused at. This method may also provide better performance in cases in which the tangent matrix becomes rank deficient.

5.2.2.3 Modified Newton This solver attempts to determine conditions under which to reform or reuse the tangent matrix based on a series of rules. This solver may be considered experimental.

5.2.2.4 Dynamic relaxation Dynamic relaxation [46] is a first order optimization technique that wraps the problem of finding the solution with a set of damped, second order differential equations. The solution is determined by dissipating the transient behavior resulting from residual forces.

5.2.2.5 Newton with line search This solver augments the standard Newton solver with a line search to determine the length of the update vector. User-specified parameters control the effort expended for the line search. The goal of the line search is to determine the optimal distance along the update direction. User-defined parameters control the maximum number of iterations expended to determine this optimal distance. The optimal distance is defined by the point along the update vector at which the residual vector is orthogonal to the update vector. An *exact* line search finds this point exactly. A second input parameter provides an approximation to the exact search.

5.2.2.6 Preconditioned nonlinear conjugate gradient This solver incorporates the preconditioned, conjugate gradient first order optimization method. User-specified parameters control the number of conjugate gradient iterations between "restarts", as well as specifications for the line search.

5.2.2.7 Interactive Newton This solver allows the user to interactively control the solution process. The underlying solver is a Newton solver with line search. The user can step forward or backward in time, modify solver parameters, and write output data. Additionally, the value of other parameters in the simulation may be viewed and changed by addressing specific *scopes* in Tahoe. The console commands also change with scope.

6 Initial conditions and boundary conditions

Initial conditions and boundary conditions may be specified on selected nodes. Additionally, the variational form of governing equations leads to natural boundary conditions that may be specified on selected parts of the domain boundary. Tahoe supports a number of geometry database types. These are listed in Table 10.1. The database format determines how nodes or sets of nodes are identified and how parts of the domain boundary are identified. Table 6.1

Table 6.1: Identification of nodes and boundaries.

database	**node ID**	**side ID**
TahoeI	node	element,facet
TahoeII	node set ID	side set ID
ExodusII [33]	node set ID	side set ID

lists how nodes and element sides are identified for the different database types. Their use will be explained with examples below. For the TahoeI format, boundary segments are explicitly specified with an element and facet number. Canonical facet numbering for the different integration domain geometries supported by Tahoe are listed in Section 7.1. For TahoeII and ExodusII databases, groups of nodes may be collected into node sets and groups of element facets are collected into side sets. Both types of sets are identified with ID numbers.

Many of the boundary conditions may be specified as functions of time. Their values have the form

$$g(t) = g\, f^{(i)}(t)\,, \tag{6.1}$$

where g is a scalar coefficient and $f^{(i)}(t)$ is a schedule function. Schedule functions are explained in Section 4.2.2. Time varying functions in the input file are specified by providing g and i, as defined in (6.1).

6.1 Nodal initial conditions

Each initial condition is specified as

[*node ID*] [*degree of freedom*] [*order*] [*value*]

The definition of the *node ID* depends on the geometry database type as listed in Table 6.1. The *order* refers to the time derivative of the field variable to which the initial conditions applies. A *node ID* of `-1` applies the boundary condition to *all* nodes.

6.2 Nodal kinematic boundary conditions

Each nodal kinematic boundary condition is specified as

[*node ID*] [*degree of freedom*] [*code*] [*schedule*] [*value*]

The definition of the *node ID* depends on the geometry database type as listed in Table 6.1. The *code* specifies if the condition is applied to the nodal unknown u, or one of its time derivatives $\dot{u}$, $\ddot{u}$, or higher. The codes are listed in Table 6.2. The *value* and *schedule* in the

Table 6.2: Codes for nodal kinematic boundary conditions.

code	**type**
0	*fixed*
1	u
2	$\dot{u}$
3	$\ddot{u}$

nodal kinematic boundary condition specification refer to g and i in (6.1), respectively, for the variation of the boundary condition in time.

6.3 Kinematic boundary condition controllers

Table 6.3: Kinematic boundary condition controllers.

code	**description**
0	K-field
1	bimaterial K-field
2	Mapped displacements
3	Tied displacements
4	Symmetry line
5	Periodic displacements

6.3.1 K-field

6.3.2 Interface K-field

6.3.3 Mapped displacements

6.3.4 Tied displacements

6.3.5 Symmetry line

6.3.6 Periodic displacements

6.4 Applied nodal forces

Each nodal force boundary condition is specified as

[*node ID*] [*degree of freedom*] [*schedule*] [*value*]

The definition of the *node ID* depends on the geometry database type as listed in Table 6.1. The *value* and *schedule* in the nodal kinematic boundary condition specification refer to g and i in (6.1), respectively, for the variation of the boundary condition in time.

6.5 Force controllers

Table 6.4: Force boundary condition controllers.

code	**description**
0	Rigid barrier (penalty)
1	Rigid sphere (penalty)
2	Rigid sphere (augmented Lagrangian)
3	Rigid sphere for meshfree domains (penalty)
4	Rigid barrier (augmented Lagrangian)

[To do: This boundary condition type actually behaves more like "elements" because they do produce a contribution to the stiffness matrix, i.e., they act as "follower forces" and are not simply defined as functions of time.]

6.5.1 Rigid barrier (penalty method)

6.5.2 Rigid sphere (penalty method)

6.5.3 Rigid sphere (augmented Lagrangian method)

6.5.4 Rigid sphere for meshfree domains (penalty method)

6.5.5 Rigid barrier (augmented Lagrangian method)

6.6 Body forces

Certain element formulations include body forces. Applied body forces are specified with the vector analog of (6.1). Their values have the form

$$\mathbf{b}(t) = \mathbf{b}\, f^{(i)}(t)\,, \tag{6.2}$$

where $\mathbf{b}$ is a vector coefficient of length n_{dof}, the number of degrees of freedom per node, and $f^{(i)}(t)$ is a scalar schedule function. Schedule functions are explained in Section 4.2.2. Body forces are specified as

$$[\mathit{schedule}] \quad [b_1] \quad \ldots \quad [b_{n_{dof}}]$$

6.7 Natural boundary conditions

Recasting balance laws in weak form leads to natural boundary conditions involving flux terms across domain surfaces. These fluxes are specified as part of the input of element parameters. Fluxes are associated with the faces of elements. As shown in Table 6.1, the description of element faces differs depending on the geometry database format. Each natural boundary condition requires specification of the flux vector at every node on the selected element face, or set of faces. Canonical number of element faces and node numbering over each face are described in Section 7.1. Natural boundary conditions are specified as

$$\begin{array}{lll} [\mathit{side\ ID}] & [\mathit{schedule}] & [\mathit{coordinate\ system}] \\ [t_{11}] \quad \ldots \quad [t_{1n_{dof}}] & & \\ \vdots & & \\ [t_{n_{fn}1}] \quad \ldots \quad [t_{n_{fn}n_{dof}}] & & \end{array}$$

Each flux vector is length n_{dof}, and a vector must be specified for all n_{fn} nodes on the element face. The *coordinate system* is identified by the values listed in Table 6.5. The facet local coordinate frame is constructed such that the last component of the flux vector $t_{n_{dof}}$ corresponds to flux in the outward normal direction. The tangential components of the flux vector a constructed from the mapping of the face to its corresponding parent integration domain.

An example specification of two-dimensional, traction boundary conditions appears below:

Table 6.5: Coordinate systems for natural boundary conditions.

code	description
0	Cartesian
1	facet local

```
1 # number of traction boundary conditions
####
1  # side ID
1  # schedule number
0  # coordinate system
# traction vectors
0.0  1.0
0.0  1.0
```

The example shows a single specification. As shown in Table 6.1, the format of the *side ID* specification depends on the geometry database format. In the example, a single integer is used to define the element edges where the boundary conditions are applied. This format would be appropriate for geometry database types that support definition of side sets. Two traction vectors, each with two components, are given indicating that the element edges specified by the *side ID* have two nodes per edge. The traction vectors are specified with respect to the global Cartesian coordinate frame, meaning the values for each vector correspond to $\{t_x, t_y\}$. A traction vector must be provided for each node of the element face. The number of nodes on each face for a given element geometry and the canonical order of the nodes on each face is described in Section 7.1. The traction is interpolated over the face from the nodal values using the standard finite element shape functions.

7 Element types

Within the structure of Tahoe, any formulation that relates nodal unknowns to the corresponding conjugate forces falls into the category of *elements*. A list of the element codes are shown in Table 7.1. The category of continuum elements is broken into three parts: Solid, Diffusion, and Mesh Free. Although mesh free methods do not use elements, they do need interpolation schemes, these schemes are discussed in this chapter.

Table 7.1: Element types.

type	**sub-type**	**code**	**sub-code**
finite element continuum	small deformation	2	N/A
	updated Lagrangian	3	
	total Lagrangian	17	
	updated Lagrangian AES [37]	29	
	updated Lagrangian Q1P0 [40]	36	
	linear diffusion	21	
meshfree continuum	small deformation	18	N/A
	total Lagrangian	19	
	strain gradient total Lagrangian	20	
finite element cohesive surface	isotropic	11	0
	anisotropic with rotating frame		1
	anisotropic with fixed frame		2
meshfree cohesive surface	anisotropic with rotating frame	22	N/A
thermal contact conducting surface	rotating frame	12	N/A
contact	penalty	14	N/A
	augmented Lagrangian (2D)	16	
	ACME [?] library	23	
pair-potential	predefined	1	N/A
	self-connecting	8	
	8 with periodic BC's	9	
two- and three-body potential	Stillinger-Weber [42]	6	N/A
	mixed Stillinger-Weber	7	
	6 with periodic BC's	10	
surface-surface interaction	two-point potential	37	N/A

7.1 Integration domains

Tahoe supports a number of Gaussian integration rules for the finite element types listed in Table 7.1. These rules are defined for the geometries listed in Table 7.2. Additionally, the meshfree formulations make use of a background grids composed of "integration cells" with the same geometries and integration rules. Each of these geometries has a corresponding

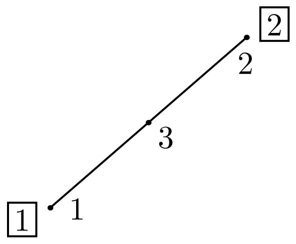
Table 7.2: Integration domain geometries and Gauss rules.

geometry	**nodes**	**Gauss rule**	**code**
point	1	1	-1
line	2, 3	1, 2, 3, 4	0
quadrilateral	4–8	1, 4, 9, 16	1
triangle	3, 6	1, 4, 6	2
hexahedron	8, 20	1, 8, 27, 64	3
tetrahedron	4, 10	1, 4	4
pentahedron	6, 15	N/A	5

local node numbering. The local node numbering, as well as the local ordering of the facets is shown in Figures 7.1 through 7.6. Although the pattern of local node numbering is not derived from a set of rules, it does display several common characteristics for all domain geometries. The vertex nodes always have lower local numbers that the mid-side nodes. For the three-dimensional domains, the local numbering of the facet nodes is always defined in a sense, or line direction, consistent with the outward normal to the face. Finally, the local node numbering for the follows the convention used by ABAQUS [16] and EnSight [10], which is not the same as the convention used by the ExodusII [33] database format for some domain geometries. However, the numbering of nodes on facets differs from the convention used by ABAQUS in that element normals point outward. When reading from or writing to an ExodusII database, Tahoe adjusts for any differences in the numbering convention.

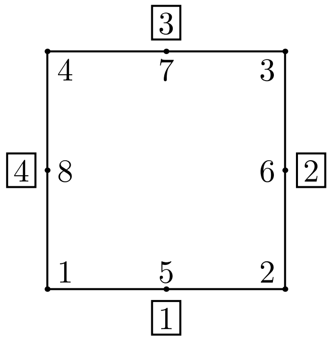
7.2 Meshfree methods

Recently, there has been intense interest in the development of meshfree numerical simulation methods. The methods used in Tahoe have their origins in Smoothed Particle Hydrodynamics (SPH) proposed by Gringold and Monaghan[14]. A number of methods have developed that employ variants of the Moving Least Squares fitting procedure of Lancaster and Salkauskas[22]. These include, but are not limited to, the Diffuse Element Method of Nayroles *et al.* [29], the Element Free Galerkin method of Belytschko *et al.* [8], the Reproducing Kernel Particle Method of Liu and Chen[23], and the Meshless Local Petrov-Galerkin method of Atluri and Zhu[5]. A number of overviews and reviews have appeared regarding these methods[7, 25, 24, 26, 27], though the interested reader will find no shortage of new developments.



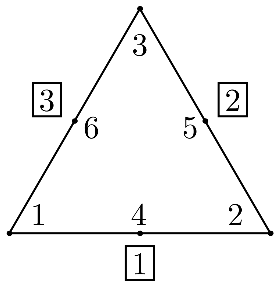
face	**node**
1	1
2	2

Figure 7.1: Node, face, and node-on-face numbering for line integration domains.



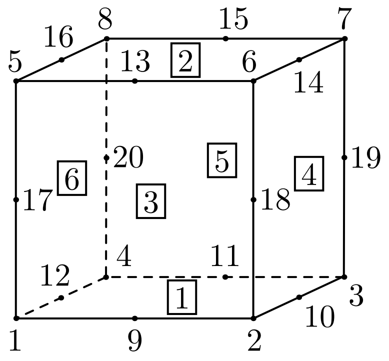
face	**nodes**
1	1, 2, 5
2	2, 3, 6
3	3, 4, 7
4	4, 1, 8

Figure 7.2: Node, face, and node-on-face numbering for quadrilateral integration domains.



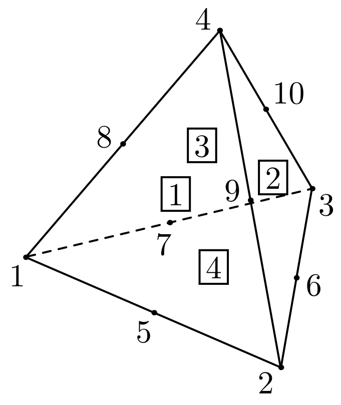
face	**nodes**
1	1, 2, 4
2	2, 3, 5
3	3, 4, 6

Figure 7.3: Node, face, and node-on-face numbering for triangular integration domains.



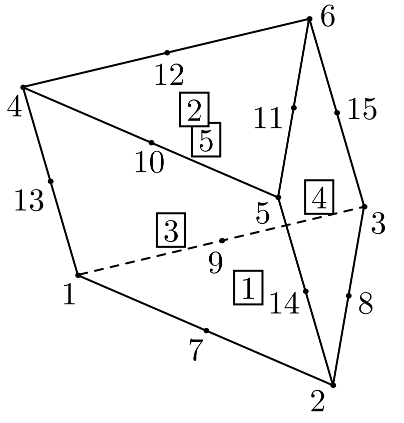
face	nodes
1	1, 4, 3, 2, 12, 11, 10, 9
2	5, 6, 7, 8, 13, 14, 15, 16
3	1, 2, 6, 5, 9, 18, 13, 17
4	2, 3, 7, 6, 10, 19, 14, 18
5	3, 4, 8, 7, 11, 20, 15, 19
6	4, 1, 5, 8, 12, 17, 16, 20

Figure 7.4: Node, face, and node-on-face numbering for hexahedral integration domains.



face	nodes
1	1, 2, 4, 5, 9, 8
2	2, 3, 4, 6, 10, 9
3	3, 1, 4, 7, 8, 10
4	1, 3, 2, 7, 6, 5

Figure 7.5: Node, face, and node-on-face numbering for tetrahedral integration domains.



face	nodes
1	1, 3, 2, 9, 8, 7
2	4, 5, 6, 10, 11, 12
3	1, 2, 5, 4, 7, 14, 10, 13
4	2, 3, 6, 5, 8, 15, 11, 14
5	3, 1, 4, 6, 9, 13, 12, 15

Figure 7.6: Node, face, and node-on-face numbering for pentahedral integration domains.

7.2.1 Reproducing Kernel Particle Method (RKPM)

RKPM belongs to class of methods for which the approximation, or "image", of a signal is given by a kernel expression. Without loss of generality, we can consider the expression for the approximation in one dimension

$$u^{R_a}(x) = \int_{-\infty}^{+\infty} \phi_a(x-y)\ u(y)\ dy, \tag{7.1}$$

where ϕ_a is alternately called a weight, kernel, or smoothing function. From the analogy to signal processing, ϕ_a may be viewed as a customizable low pass filter between the original signal, or data, $u(y)$ and its reproduced image. This function is positive, even, and has compact support characterized by the dilation parameter a. Liu and Chen[23] improved the accuracy of the method by modifying the window function with a correction to yield a reproducing condition as

$$u^{R_a}(x) = \int_{-\infty}^{+\infty} \overline{\phi}_a(x-y)\ u(y)\ dy, \tag{7.2}$$

where the modified window function

$$\overline{\phi}_a(x-y) = C(x; x-y)\ \phi_a(x-y) \tag{7.3}$$

incorporates the polynomial

$$C(x; x-y) = b_0(x) + b_1(x)\ (x-y) + b_2(x)\ (x-y)^2 + \ldots + b_m(x)\ (x-y)^m \tag{7.4}$$

which ensures that the approximation can exactly represent polynomials of order m. In general, we may express the correction function as

$$C(\mathbf{x};\mathbf{x}-\mathbf{y}) = \mathbf{b}(\mathbf{x}) \cdot \mathbf{P}(\mathbf{x}-\mathbf{y}) \tag{7.5}$$

where $\mathbf{P}$ is a basis of polynomials that possess the desired degree of completeness and $\mathbf{b}(\mathbf{x})$ is an vector of unknown coefficients determined from the reproducing condition (7.2). The requirement that each member of the basis be reproduced follows from (7.2) as

$$\mathbf{P}(\mathbf{0}) = \int_{-\infty}^{+\infty} \mathbf{b}(\mathbf{x}) \cdot \mathbf{P}(\mathbf{x}-\mathbf{y})\ \phi_a(\mathbf{x}-\mathbf{y})\ \mathbf{P}(\mathbf{x}-\mathbf{y})\ d\mathbf{y}. \tag{7.6}$$

The vector of coefficients follows as

$$\mathbf{b}(\mathbf{x}) = \mathbf{M}_a^{-1}(\mathbf{x})\,\mathbf{P}(\mathbf{0})\,, \tag{7.7}$$

where

$$\mathbf{M}_a(\mathbf{x}) = \int_{-\infty}^{+\infty} \mathbf{P}(\mathbf{x}-\mathbf{y}) \otimes \mathbf{P}(\mathbf{x}-\mathbf{y})\ \phi_a(\mathbf{x}-\mathbf{y})\,d\mathbf{y} \tag{7.8}$$

is known as the moment matrix. Using the result from (7.7), the reproducing condition (7.2) can be written as

$$\mathbf{u}^{R_a}(\mathbf{x}) = \int_{-\infty}^{+\infty} \mathbf{b}(\mathbf{x}) \cdot \mathbf{P}(\mathbf{x}-\mathbf{y})\,\phi_a(\mathbf{x}-\mathbf{y})\ \mathbf{u}(\mathbf{y})\ d\mathbf{y}. \tag{7.9}$$

First and higher order gradients of the field representation follow from (7.7)–(7.9) through simple differentiation. For clarity, we present the expressions for derivatives. We note that these derivatives may be evaluated exactly though some early work in the meshfree field advocated approximate expressions for these derivatives[29]. The gradient of the field may be expressed as

$$\begin{aligned} \frac{\partial \mathbf{u}^{R_a}(\mathbf{x})}{\partial \mathbf{x}} = \int_{-\infty}^{+\infty} \mathbf{u}(\mathbf{y}) \otimes \Bigg[& \frac{\partial \mathbf{b}(\mathbf{x})}{\partial \mathbf{x}}\ \mathbf{P}(\mathbf{x}-\mathbf{y})\ \phi_a(\mathbf{x}-\mathbf{y}) + \\ & \mathbf{b}(\mathbf{x})\ \frac{\partial \mathbf{P}(\mathbf{x}-\mathbf{y})}{\partial \mathbf{x}}\,\phi_a(\mathbf{x}-\mathbf{y}) + \\ & \mathbf{b}(\mathbf{x}) \cdot \mathbf{P}(\mathbf{x}-\mathbf{y})\ \frac{\partial \phi_a(\mathbf{x}-\mathbf{y})}{\partial \mathbf{x}} \Bigg]\,d\mathbf{y}, \end{aligned} \tag{7.10}$$

where the gradient of the coefficient vector

$$\frac{\partial \mathbf{b}(\mathbf{x})}{\partial \mathbf{x}} = -\mathbf{M}_a^{-1}(\mathbf{x})\,\frac{\partial \mathbf{M}_a(\mathbf{x})}{\partial \mathbf{x}}\,\mathbf{b}(\mathbf{x}) \tag{7.11}$$

follows from (7.7) making use of the relation

$$\frac{\partial\left[\mathbf{M}_a^{-1}\right]_{ij}}{\partial x_k} = -\left[\mathbf{M}_a^{-1}\right]_{ir}\frac{\partial\left[\mathbf{M}_a\right]_{rs}}{\partial x_k}\left[\mathbf{M}_a^{-1}\right]_{sj}. \tag{7.12}$$

Higher-order gradients may be calculated by further differentiation of (7.10)–(7.12). Notably, the expressions for the representation of the unknown field variables are general for an arbitrary number of field and spatial dimensions, including the expressions for the first and higher order gradients of the field.

In evaluating the representations for the field numerically, we discretize the integrals in the previous expressions. The discrete reproducing condition follows from (7.9) as

$$\mathbf{u}^{R_a^h}(\mathbf{x}) = \sum_{I=1}^{N_p} \mathbf{b}(\mathbf{x}) \cdot \mathbf{P}(\mathbf{x}-\mathbf{x}_I)\,\phi_a(\mathbf{x}-\mathbf{x}_I)\;\mathbf{u}_I\,\Delta V_I, \tag{7.13}$$

where N_p is the number of sampling points, or particles, under consideration, and $\mathbf{x}_I$, $\mathbf{u}_I = \mathbf{u}(\mathbf{x}_I)$, and ΔV_I are the coordinates, field value, and integration weight (volume) associated with particle I, respectively. From (7.13), we can identify the RKPM nodal shape functions as

$$\mathbf{u}^{R_a^h}(\mathbf{x}) = \sum_{I=1}^{N_p} \Phi_{aI}(\mathbf{x})\;\mathbf{u}_I, \tag{7.14}$$

where

$$\Phi_{aI}(\mathbf{x}) = \mathbf{b}(\mathbf{x}) \cdot \mathbf{P}(\mathbf{x}-\mathbf{x}_I)\,\phi_a(\mathbf{x}-\mathbf{x}_I)\;\Delta V_I. \tag{7.15}$$

Unlike the shape functions derived for standard finite element methods, the RKPM shape functions (7.15) do not in general possess the so-called Kronecker delta property, that is

$$\Phi_{aI}(\mathbf{x}_J) \neq \delta_{IJ}. \tag{7.16}$$

The lack of this property complicates the enforcement of essential boundary conditions. Furthermore, this characteristic affects the enforcement of contact constraint and the imposition of cohesive tractions.

[To do: User must specify
(1) the window function
(2) basis functions
(3) order of completeness for polynomial basis functions]

7.3 Solid deformation

This section describes the elements available for performing analysis of stress and deformation in solids. The material models available for these elements are described in Section 8.

7.3.1 Element 29: finite strain element with enhanced modes

This element formulation is due to Simo, Armero, and Taylor [37]. The element incorporates internal element modes of deformation to improve performance under bending and under nearly incompressible states of deformation.

7.3.2 Element 36: finite strain mixed element

The formulation is due to Simo, Taylor, and Pister [40]. The basic idea behind the formulation is to represent the pressure and dilatation Θ as separate fields from the displacement. For the continuous case, the determinant of the deformation gradient

$$J = \det \mathbf{F} = \mathbf{1} + \frac{\partial \mathbf{u}}{\partial \mathbf{X}} \tag{7.17}$$

is equal to the dilatation. However, when the displacement field $\mathbf{u}$ is restricted to a finite dimensional representation, it may not contain enough degrees of freedom to represent nearly incompressible deformations without making the response overly stiff. Therefore, the dilatation and pressure are represented as separate fields. The modified deformation gradient is given by

$$\bar{\mathbf{F}} = \left(\frac{\Theta}{J} \right)^{1/3} \mathbf{F}. \tag{7.18}$$

The remainder of the formulation results as a consequence. For Q1P0, the (2D) bi- or (3D) trilinear displacement field (Q1) is combined with piecewise constant (P0) pressure and dilatation fields. Since the space for these fields is restricted to within element domains, these degrees of freedom can be determined analytically at the element level and substituted into the remaining element equations to results in a purely displacement-based formulation.

Note that several errors appear in the derivation of the consistent tangent in the CMAME paper. Therefore, the implementation of the tangent here does not match the published formulation.

7.3.3 Output variables

The values that can be written to output as nodal or element quantities are listed in Tables 7.3 and 7.4, respectively. The element output of integration point values have the labels

$$\texttt{ip}N\texttt{.}\,[\textit{label}] \qquad N = 1, \ldots, n_{ip},$$

where n_{ip} is the number of element integration points.

7.3.4 Constitutive models

The materials available for analysis of solid deformation are listed in Table 8.1.

Table 7.3: Nodal output values for solid deformation.

#	var.	**description**	**labels**
1	$\mathbf{X}$	reference coordinates	`x` $[i] \quad i = 0, \ldots, n_{sd}$
2	$\mathbf{d}$	displacements	`D_` $[i] \quad i =$ `X`, `Y`, $\ldots, n_{dof}$
3	$\boldsymbol{\sigma}$	extrapolated stresses	2D: `s11`, `s22`, `s12` 3D: `s11`, `s22`, `s33`, `s23`, `s13`, `s12`
4	$\hat{\boldsymbol{\sigma}}$	extrapolated principal stresses	`s` $[i] \quad i = 0, \ldots, n_{sd}$
5	ϕ	extrapolated strain energy density	`phi`
6	c_d, c_s	extrapolated acoustical wave speeds	2D: `cd`, `cs` 3D: `cd`, `cs_min`, `cs_max`
7		extrapolated material model output	determined by model

Table 7.4: Element output values for solid deformation.

#	var.	**description**	**labels**
1	$\bar{\mathbf{X}}$	reference centroid coordinates	`xc_` $[i] \quad i = 0, \ldots, n_{sd}$
2	m	integrated mass	`mass`
3	ϕ	integrated strain energy	`U`
4	T	integrated kinetic energy	`T`
5	$\mathbf{L}$	integrated linear momentum	`L_` $[i] \quad i =$ `X`, `Y`, $\ldots, n_{dof}$
6	$\boldsymbol{\sigma}$	integration point stresses	2D: `s11`, `s22`, `s12` 3D: `s11`, `s22`, `s33`, `s23`, `s13`, `s12`
7		integration point material output	determined by model

7.4 Diffusion

7.4.1 Output variables

7.4.2 Constitutive models

7.5 Cohesive surface elements

At every point along a cohesive surface, we define a local coordinate frame which resolves the opening displacements across the surface into normal and shear components. As shown in Figure 7.7, this coordinate frame is not uniquely defined for finite deformations. We choose to define this coordinate system with respect to the mid-plane between the displaced surfaces. Alternately, this frame may be fixed to the upper or lower facet. The discussion that follows is independent of the particular choice of local coordinate system though we present specific expressions for the mid-plane. Let the transformation of the gap vector from

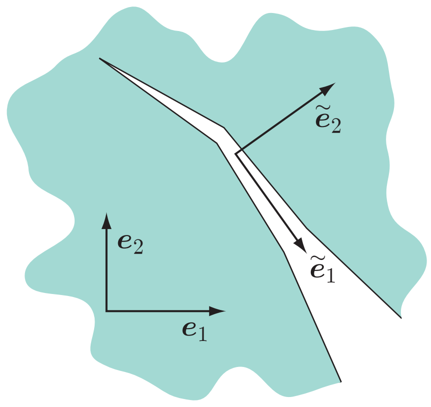


Figure 7.7: Local coordinate frame $\tilde{\mathbf{e}}_i$ constructed along the surface where $\mathbf{e}_i$ is the global cartesian frame.

its representation in the global coordinate frame $\mathbf{\Delta}$ to its local representation $\tilde{\mathbf{\Delta}}$ in a frame defined with respect to the cohesive surface be given by

$$\tilde{\mathbf{\Delta}} = \mathbf{Q}^{\mathrm{T}}\,\mathbf{\Delta}. \tag{7.19}$$

The transformation resolves the opening displacement into normal and shear components relative to the current orientation of the surface. This distinction allows for the definition of mixed-mode, surface constitutive relations that show a dependence on the character of the opening displacement. In a variational setting, the contribution to the virtual work from cohesive surface elements is

$$\delta w^{\mathrm{CSE}} = \int_{\Gamma_0} \mathbf{T} \cdot \delta\mathbf{\Delta}\, d\Gamma, \tag{7.20}$$

where the integration domain Γ_0 is defined in the undeformed configuration and $\delta\mathbf{\Delta}$ is the variation in the surface opening displacement. We review the formulation of several surface constitutive models in Section 7.5.2. The discussion that follows applies to all displacement-driven cohesive relations $\tilde{\mathbf{T}}(\tilde{\mathbf{\Delta}})$ defined in the local surface frame. From (7.19), the traction in the global coordinate frame is given by the transformation

$$\mathbf{T} = \mathbf{Q}\,\tilde{\mathbf{T}}. \tag{7.21}$$

7.5.1 Output variables

Nodal and element output variables are prescribed in the input file by specifying a `1` for variables to be written and a `0` for variables not to be written. The sequence and definition of the nodal and element output variables are described in the next sections.

7.5.1.1 Nodal output variables

The output variables written by cohesive surface elements are listed in Tables 7.5 and Table 7.6. Output variables defined at the integration points are projected to the nodes using the algorithm proposed by Zinkiewickz *et al.* [**?**].

The isotropic and anisotropic element write the same variables; however, some of the variables returned as scalars for the isotropic formulation are returned as vectors for the anisotropic case.

Table 7.5: Nodal output variables for isotropic cohesive surface elements.

#	**var.**	**description**	**labels**
1	$\mathbf{X}$	reference coordinates	`x` $[i] \quad i = 0, \ldots, n_{sd}$
2	$\mathbf{d}$	displacements	`D_` $[i] \quad i = \mathtt{X}, \mathtt{Y}, \ldots, n_{dof}$
3	$\|\|\Delta\|\|$	opening displacements	`jump`
4	$\|\|\mathbf{T}\|\|$	traction	`Tmag`
5		constitutive output variables	determined by model

Table 7.6: Nodal output variables for anisotropic cohesive surface elements.

#	**var.**	**description**	**labels**
1	$\mathbf{X}$	reference coordinates	`x` $[i] \quad i = 0, \ldots, n_{sd}$
2	$\mathbf{d}$	displacements	`D_` $[i]\, i = \mathtt{X}, \mathtt{Y}, \ldots, n_{dof}$
3	$\tilde{\boldsymbol{\Delta}}$	local opening displacements	2D: `d_t`, `d_n` 3D: `d_t1`, `d_t2`, `d_n`
4	$\tilde{\mathbf{T}}$	local traction	2D: `T_t`, `T_n` 3D: `T_t1`, `T_t2`, `T_n`
5		constitutive output variables	determined by model

Note that the normal component for the output of the opening displacement and the traction is always the last value. The tangent directions, denoted with the `t1` and `t2`, are determined by the coordinate mapping over surface element and generally vary from element to element.

7.5.1.2 Element output variables

The element output variables for both isotropic and anisotropic cohesive surface elements are listed in Table 7.7. Note that some of the variables are element-averaged quantities, while others are integrated over the element.

7.5.2 Constitutive models

The element formulation presented in Section 7.5 makes use of a local coordinate frame defined with respect the deforming facets of the cohesive surface. This construction allows

Table 7.7: Element output variables for cohesive surface elements.

#	var.	description	labels
1	$\overline{\mathbf{X}}$	centroid in reference coordinates	xc_$[i]$ $\quad i = 0, \ldots, n_{sd}$
2	ϕ	integrated cohesive energy	phi
3	$\bar{t}$	average traction	2D: T_t, T_n 3D: T_t1, T_t2, T_n

all surface constitutive relations to define the evolution of the local traction $\tilde{\mathbf{T}}$ with respect to the local opening displacement $\tilde{\boldsymbol{\Delta}}$, while the kinematics of large deformation and surface rotations are accounted for at the element level. For the case of an elastic, or reversible, cohesive relation, the traction is derived from a free energy potential φ as

$$\tilde{\mathbf{T}} = \frac{\partial \varphi}{\partial \tilde{\boldsymbol{\Delta}}} \tag{7.22}$$

while the surface stiffness may be expressed as

$$\frac{\partial \tilde{\mathbf{T}}}{\partial \tilde{\boldsymbol{\Delta}}} = \frac{\partial^2 \varphi}{\partial \tilde{\boldsymbol{\Delta}} \partial \tilde{\boldsymbol{\Delta}}}. \tag{7.23}$$

Motivated by the universal binding energy curves of Rose *et al.* [32] from atomistic modeling, Xu and Needleman[50] proposed the mixed-mode cohesive potential

$$\varphi\left(\tilde{\boldsymbol{\Delta}}\right) = \phi_n + \phi_n \, \exp\left(-\frac{\tilde{\Delta}_n}{\delta_n}\right) \left(\left[1 - r + \frac{\tilde{\Delta}_n}{\delta_n}\right] \frac{1-q}{r-1} - \exp\left(-\frac{\tilde{\Delta}_t^2}{\delta_t^2}\right) \left[q + \left(\frac{r-q}{r-1}\right) \frac{\tilde{\Delta}_n}{\delta_n}\right]\right). \tag{7.24}$$

From (7.22), the components of the traction are

$$\tilde{T}_n\left(\tilde{\boldsymbol{\Delta}}\right) = \frac{\phi_n}{\delta_n} \exp\left(-\frac{\tilde{\Delta}_n}{\delta_n}\right) \left(\frac{\tilde{\Delta}_n}{\delta_n} \exp\left(-\frac{\tilde{\Delta}_t^2}{\delta_t^2}\right) + \frac{1-q}{r-1} \left[1 - \exp\left(-\frac{\tilde{\Delta}_t^2}{\delta_t^2}\right)\right] \left[r - \frac{\tilde{\Delta}_n}{\delta_n}\right]\right) \tag{7.25}$$

and

$$\tilde{T}_t\left(\tilde{\boldsymbol{\Delta}}\right) = 2\,\phi_n \frac{\tilde{\Delta}_t}{\delta_t^2} \left(q + \left[\frac{r-q}{r-1}\right] \frac{\tilde{\Delta}_n}{\delta_n}\right) \exp\left(-\frac{\tilde{\Delta}_n}{\delta_n}\right) \exp\left(-\frac{\tilde{\Delta}_t^2}{\delta_t^2}\right). \tag{7.26}$$

In (7.24)–(7.26), the ratio $q = \frac{\phi_t}{\phi_n}$ relates the pure normal mode to shear mode fracture energy and $r = \frac{\tilde{\Delta}_n^*}{\delta_n}$ defines $\tilde{\Delta}_n^*$, the value of $\tilde{\Delta}_n$ after complete shear separation with $\tilde{T}_n = 0$. The effect of r vanishes when $q = 1$, which is the value used for all calculations in this study. The normal and tangential openings are defined as $\tilde{\Delta}_n = \tilde{\boldsymbol{\Delta}} \cdot \tilde{\mathbf{n}}$ and $\tilde{\Delta}_n = \tilde{\boldsymbol{\Delta}} \cdot \tilde{\mathbf{t}}$ with respect to the local normal and tangent. Xu and Needleman developed this cohesive relation in two dimensions. We extend this relation to three dimensions by assuming the response to be isotropic with respect to the tangential opening. That is, the response is expressed only in terms of the magnitude of the tangential opening, which can also be written as

$$\tilde{\Delta}_t = \left| \tilde{\mathbf{\Delta}} - \tilde{\Delta}_n \, \tilde{\mathbf{n}} \right|, \tag{7.27}$$

to emphasize that it is independent of $\tilde{\mathbf{t}}$. For the three-dimensional model, we did not attempt to incorporate any additional effects from the relative rotation of the surfaces about the normal.

7.6 Cohesive surface elements for meshfree domains

7.6.1 Constitutive models

7.6.2 Parameters for adaptive insertion

7.7 Contact

In the input section for contact elements, the user specifies a regularization constant and the entities involved with contact. Contact entities are specified in two sections, contact surfaces and contact, or striker, nodes.

7.7.1 Penalty formulation

With this contact formulation, the penetration of a striker node into a contact surface is penalized with a quadratic potential as

$$U_c = \frac{1}{2} \, k_c \, g^2, \tag{7.28}$$

where k_c is the contact stiffness, or regularization parameter, and g is the penetration distance. When used in conjunction with cohesive surface elements, a reasonable choice for the contact stiffness is

$$k_c = k_{\text{CSE}} \, A, \tag{7.29}$$

where A is the area of the cohesive surface element, and k_{CSE} is the tangent stiffness of the surface constitutive relation. For the potential proposed by Xu-Needleman potential 7.24, this stiffness is

$$k_{\text{CSE}} = \frac{\phi}{\delta_n^2}. \tag{7.30}$$

7.7.2 Augmented Lagrangian formulation

The augmented Lagrangian formulation in Tahoe is based on the formulation by Heegaard and Curnier [15].

7.8 Pair-potential

7.9 Two- and three-body potential

7.10 Surface-surface interactions

7.10.1 Element 37: two-point surface interactions

The formulation of surface-surface interactions is similar to what is described by Argento *et al.* [3]. The interaction energy of two surfaces S_1 and S_2 is given by the double-integral

$$U(S_1, S_2, t) = \alpha_{12}(t) \int_{S_1} \int_{S_2} u(|\mathbf{g}|)\, dS_1 dS_2, \tag{7.31}$$

where u is a potential function and $\mathbf{g}$ is a vector connecting a point on S_1 with a point on S_2. The function $\alpha_{12}(t)$ is introduced to allow the interaction forces to scale with simulation time. Two of differential elements dS_1 and dS_2 interact if

$$\mathbf{n}_1 \cdot \mathbf{g} > 0 \quad \text{and} \quad \mathbf{n}_2 \cdot \mathbf{g} < 0, \tag{7.32}$$

where $\mathbf{n}_1$ and $\mathbf{n}_2$ are the normals to dS_1 and dS_2, respectively. The interaction force is then given by

$$\mathbf{f} = -\frac{\partial U}{\partial \mathbf{g}}. \tag{7.33}$$

8 Constitutive models for solid mechanics

This section of the guide describes the constitutive models available for analysis of displacements and stresses. A separate section is dedicated to each material model. Each section provides some background information about the model followed by a description of the input required for three- and two-dimensional analysis. All materials support implicit as well as explicit analysis. That is, most models implement a consistent tangent matrix that should produce locally quadratic convergence when used with Newton-type, nonlinear solution algorithms (see Section 5.2.2). As will be noted in the sections that follow, some models only provide approximate tangents that will result in slower convergence rates with Newton-type solvers.

Table 8.1: Materials for analysis of displacements and stress.

code	**description**
1	small deformation Kirchhoff-St. Venant
2	finite deformation Kirchhoff-St. Venant
3	small deformation cubic
4	finite deformation cubic
5	uncoupled volumetric/deviatoric
6	quadratic-logarithmic (Simo)
7	quadratic-logarithmic (Ogden)
8	small deformation J_2 plasticity
9	finite deformation J_2 plasticity
10	finite deformation J_2 plasticity in principal stretches
11	small deformation Drucker-Prager plasticity
12	2D hexagonal Lennard-Jones lattice
13	plane strain FCC Lennard-Jones lattice
14	FCC lattice with EAM potentials
15	Stillinger-Weber diamond cubic
16	Virtual Internal Bond model
17	isotropic Virtual Internal Bond model (Simo)
18	isotropic Virtual Internal Bond model (Ogden)
19	isotropic Virtual Internal Bond model with J_2 plasticity
30	Thermo-viscoplastic model
31	Thermo-viscoplastic model
40	Finite strain viscoplastic model
45	BCJ model
46	BCJ model with damage
47	BCJ model with damage
49	Finite strain crystal elasticity

Table 8.1: Materials for analysis of displacements and stress.

code	description
50	Crystal plasticity model
51	Crystal plasticity model
52	Gradient crystal plasticity model
55	Crystal plasticity model
56	Crystal plasticity model
57	Gradient crystal plasticity model
61	Neo-Hookean viscoelastic model
62	Neo-Hookean viscoelastic model
63	Finite strain viscoelastic Kirchhoff-St.Venant
64	Small strain viscoelastic Kirchhoff-St.Venant
80	ABAQUS/Standard [16] UMAT BCJ
90	ABAQUS/Explicit VUMAT BCJ

8.0.2 Common input parameters

All constitutive models for solid mechanics share some common input parameters. An example of these parameters is shown below:

```
# Rayleigh damping parameters
0.0   # mass proportionate damping
0.0   # stiffness proportionate damping
# mass density
1.0   # mass/reference volume
# thermal expansion
0     # schedule number
0.0   # % expansion
```

The force for Rayleigh damping is defined as

$$\mathbf{F}_R = \mathbf{C}_R \, \mathbf{v} \tag{8.1}$$

where the damping matrix $\mathbf{C}_R$ is a linear combination of the mass and stiffness matrices

$$\mathbf{C}_R = a \, \mathbf{M} + b \, \mathbf{K}, \tag{8.2}$$

where a and b are the parameters specified in the input file.

Thermal expansion may be defined as a function of time with two parameters. The schedule number corresponds to the schedules defined in the time parameters block of the input file 4.2. The second parameter defines the volume change as a percentage. Thermal expansion is disabled by specifying a value of `0` for the schedule number. Thermal strains are imposed differently for small strain and finite strain material models. For small strain materials, an additive decomposition of the strain is assumed

$$\boldsymbol{\epsilon} = \boldsymbol{\epsilon}^{\theta} + \boldsymbol{\epsilon}^{\text{mechanical}}, \tag{8.3}$$

where the thermal strain is

$$\boldsymbol{\epsilon}^{\theta}(t) = \varepsilon(t)\,\mathbf{1} \tag{8.4}$$

For materials formulated at finite strains, the multiplicative split

$$\mathbf{F} = \mathbf{F}^{\text{mechanical}}\,\mathbf{F}^{\theta}, \tag{8.5}$$

of the deformation gradient is assumed, where the thermal deformation gradient is

$$\mathbf{F}^{\theta}(t) = (1 + \varepsilon(t))\,\mathbf{1}. \tag{8.6}$$

8.0.3 Common input parameters for 2D analysis

If constructed using the standard Tahoe materials model toolbox, all materials for two-dimensional analysis contain two common parameters. An example of these parameters is shown below:

```
####### common 2D parameters
1.0   # thickness
2     # constraint option
```

The thickness t is used to transform the differential volume element as

$$d\Omega = t\,dx_1\,dx_2. \tag{8.7}$$

The acceptable values for the constraint option are shown in Table 8.2. Some materials

Table 8.2: Constraint options for two-dimensional analysis.

code	**description**
1	plane stress
2	plane strain

models have been specifically formulated as plane stress or plane strain. These materials will silently override the user-specified value of the constraint option.

8.1 Material 1: small deformation Kirchhoff-St. Venant

The stress response of this isotropic model is governed by the strain energy function

$$\Phi(\boldsymbol{\epsilon}) = \frac{1}{2}\epsilon_{ij}\,\mathsf{C}_{ijkl}\,\epsilon_{kl}, \tag{8.8}$$

where $\boldsymbol{\epsilon}$ is the infinitesimal strain tensor and $\mathbf{C}$ is the isotropic tensor of elastic constants which may be written as

$$\mathbf{C} = \begin{bmatrix} \lambda+2\mu & \lambda & \lambda & 0 & 0 & 0 \\ \lambda & \lambda+2\mu & \lambda & 0 & 0 & 0 \\ \lambda & \lambda & \lambda+2\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & \mu & 0 & 0 \\ 0 & 0 & 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu \end{bmatrix} \tag{8.9}$$

for the three-dimensional case. The Lamé constants λ and μ are related to Young's modulus E and Poisson's ratio ν by

$$\lambda = \frac{\nu\,E}{(1+\nu)\,(1-2\,\nu)} \tag{8.10}$$

and

$$\mu = \frac{E}{2\,(1+\nu)}. \tag{8.11}$$

8.1.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### Kirchhoff-St.Venant
1     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν.

8.1.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### Kirchhoff-St.Venant
1     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### 2D parameters
1.0   # thickness
2     # constraint option
```

After the common parameters 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis.

8.1.3 Output values

This constitutive model does not have any output values.

8.2 Material 2: finite deformation Kirchhoff-St. Venant

This model is the finite deformation version of the material model described in Section 8.1. The stress response of this isotropic model is governed by the strain energy function

$$\Phi(\mathbf{E}) = \frac{1}{2} E_{ij}\, \mathsf{C}_{ijkl}\, E_{kl}, \tag{8.12}$$

where $\mathbf{E}$ is the Green Lagrangian strain tensor

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{1}\right), \tag{8.13}$$

where $\mathbf{F}$ is the deformation gradient and $\boldsymbol{\mathsf{C}}$ is the isotropic tensor of elastic constants given by (8.9).

8.2.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### Kirchhoff-St.Venant
2     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν.

8.2.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### Kirchhoff-St.Venant
2     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### 2D parameters
1.0   # thickness
2     # constraint option
```

After the common parameters 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis.

8.2.3 Output values

This constitutive model does not have any output values.

8.3 Material 3: small deformation cubic

The stress response of this model is governed by the strain energy function

$$\Phi(\boldsymbol{\epsilon}) = \frac{1}{2}\epsilon_{ij}\,\mathsf{C}_{ijkl}\,\epsilon_{kl}, \tag{8.14}$$

where $\boldsymbol{\epsilon}$ is the infinitesimal strain tensor and $\mathbf{C}$ is the tensor of elastic constants which may be written as

$$\mathbf{C} = \begin{bmatrix} \mathsf{C}_{11} & \mathsf{C}_{12} & \mathsf{C}_{12} & 0 & 0 & 0 \\ \mathsf{C}_{12} & \mathsf{C}_{11} & \mathsf{C}_{12} & 0 & 0 & 0 \\ \mathsf{C}_{12} & \mathsf{C}_{12} & \mathsf{C}_{11} & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathsf{C}_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathsf{C}_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathsf{C}_{44} \end{bmatrix} \tag{8.15}$$

for the three-dimensional case. The elasticity tensor in (8.15) has cubic symmetry. The degree of anisotropy is often characterized by the parameter

$$A = \frac{2\,\mathsf{C}_{44}}{\mathsf{C}_{11} - \mathsf{C}_{12}}, \tag{8.16}$$

where $A = 1$ for isotropy.

8.3.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### cubic model
3     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### cubic moduli
100.0 # C11
 10.0 # C12
 50.0 # C44
```

After the common parameters described in Section 8.0.2, the model requires specification of the cubic moduli C_{11}, C_{12}, and C_{44}.

8.3.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### cubic model
3     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### cubic moduli
100.0 # C11
 10.0 # C12
 50.0 # C44
####### orientation
0     # is rotated
0.0   # rotation angle
####### 2D parameters
1.0   # thickness
2     # constraint option
```

After the common parameters described in Section 8.0.2, the model requires specification of the cubic moduli C_{11}, C_{12}, and C_{44}. The cubic moduli are followed by specification of the lattice rotation, consisting of a boolean flag indicating whether the lattice is rotated and the angle rotation. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis.

8.3.3 Output values

This constitutive model does not have any output values.

8.4 Material 4: finite deformation cubic

This model is the finite deformation version of the material model described in Section 8.3. The stress response of this isotropic model is governed by the strain energy function

$$\Phi(\mathbf{E}) = \frac{1}{2} E_{ij}\, \mathsf{C}_{ijkl}\, E_{kl}, \tag{8.17}$$

where $\mathbf{E}$ is the Green Lagrangian strain tensor

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^{\mathrm{T}}\mathbf{F} - \mathbf{1}\right), \tag{8.18}$$

where $\mathbf{F}$ is the deformation gradient and $\boldsymbol{\mathsf{C}}$ is the tensor of elastic constants given by (8.15).

8.4.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### cubic model
4     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### cubic moduli
100.0 # C11
 10.0 # C12
 50.0 # C44
```

After the common parameters described in Section 8.0.2, the model requires specification of the cubic moduli C_{11}, C_{12}, and C_{44}.

8.4.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### cubic model
4     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### cubic moduli
100.0 # C11
 10.0 # C12
 50.0 # C44
####### orientation
0     # is rotated
0.0   # rotation angle
####### 2D parameters
1.0   # thickness
2     # constraint option
```

After the common parameters described in Section 8.0.2, the model requires specification of the cubic moduli C_{11}, C_{12}, and C_{44}. The cubic moduli are followed by specification of the lattice rotation, consisting of a boolean flag indicating whether the lattice is rotated and the angle rotation. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis.

8.4.3 Output values

This constitutive model does not have any output values.

8.5 Material 5: uncoupled volumetric/deviatoric

This model is the finite deformation, isotropic constitutive model proposed by Simo *et al.* [40] for which the response to volumetric and deviatoric deformations is uncoupled. The hyperelastic stress response is derived from the stored energy function

$$\Phi = U(J) + \bar{\Phi}(\bar{\mathbf{b}}) \tag{8.19}$$

$$U(J) = \frac{1}{2}\,\kappa \left[\frac{1}{2}\left(J^2 - 1\right) - \ln J\right] \tag{8.20}$$

$$\bar{\Phi}(\bar{\mathbf{b}}) = \frac{1}{2}\,\mu \left(\operatorname{tr}\bar{\mathbf{b}} - 3\right), \tag{8.21}$$

where κ is the bulk modulus, μ is the shear modulus, $\bar{\mathbf{b}} = J^{-2/3}\mathbf{b}$ is the deviatoric part of the left Cauchy-Green stretch tensor

$$\mathbf{b} = \mathbf{F}\,\mathbf{F}^{\mathrm{T}}, \tag{8.22}$$

and $J = \det \mathbf{F}$. As a result of the form of the volumetric portion of the stored energy function $U(J)$ (8.20), this model is stable under severe compression.

The bulk and shear moduli are related to Young's modulus E and Poisson's ratio ν by

$$\kappa = \frac{E}{3\,(1 - 2\,\nu)} \tag{8.23}$$

and

$$\mu = \frac{E}{2\,(1 + \nu)}. \tag{8.11}$$

8.5.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### uncoupled volumetric/deviatoric model
5     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν.

8.5.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### uncoupled volumetric/deviatoric model
5     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### 2D parameters
1.0   # thickness
2     # constraint option (override)
```

After the common parameters 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis. This two-dimensional model is formulated in plane strain, so other values for the constraint option are silently overridden.

8.5.3 Output values

This constitutive model does not have any output values.

8.6 Material 6: quadratic-logarithmic (Simo)

This model is the finite deformation, isotropic constitutive model proposed by Simo *et al.* [36] expressed in principal stretch space. The hyperelastic stress response is derived from the stored energy function expressed in terms of the logarithms of the principal stretches as

$$\Phi(e_1, e_2, e_3) = \frac{1}{2}\,\lambda\,(e_1 + e_2 + e_3)^2 + \mu\left(e_1^2 + e_2^2 + e_3^2\right), \tag{8.24}$$

where

$$e_A = \log \lambda_A, \qquad A = 1, 2, 3. \tag{8.25}$$

The Lamé constants λ and μ are related to Young's modulus E and Poisson's ratio ν by

$$\lambda = \frac{\nu\,E}{(1+\nu)\,(1-2\,\nu)} \tag{8.10}$$

and

$$\mu = \frac{E}{2\,(1+\nu)}. \tag{8.11}$$

The implementation of the model follows the spectral formulation developed by Simo and Taylor [39]. Notably, the principal values of the Kirchhoff stress are linearly related to the logarithmic strains

$$\begin{Bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{Bmatrix} = \begin{bmatrix} \kappa + \frac{4}{3}\mu & \kappa - \frac{2}{3}\mu & \kappa - \frac{2}{3}\mu \\ \kappa - \frac{2}{3}\mu & \kappa + \frac{4}{3}\mu & \kappa - \frac{2}{3}\mu \\ \kappa - \frac{2}{3}\mu & \kappa - \frac{2}{3}\mu & \kappa + \frac{4}{3}\mu \end{bmatrix} \begin{Bmatrix} e_1 \\ e_2 \\ e_3 \end{Bmatrix} \tag{8.26}$$

for elastic deformations. As a result of (8.26), the return mapping algorithm is readily transferred to stress space, leading to a highly efficient numerical implementation.

The model described in Section 8.7 uses the stored energy function (8.24) with the spectral formulation developed by Ogden [30]. The Ogden formulation allows more straightforward handling of repeated principal stretches.

8.6.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### quad-log (Simo)
6     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν.

8.6.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### quad-log (Simo)
6     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### 2D parameters
1.0   # thickness
2     # constraint option (override)
```

After the common parameters 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis. This two-dimensional model is formulated in plane strain, so other values for the constraint option are silently overridden.

8.6.3 Output values

This constitutive model does not have any output values.

8.7 Material 7: quadratic-logarithmic (Ogden)

This model is the finite deformation, isotropic constitutive model proposed by Simo *et al.* [36] expressed in principal stretch space. The hyperelastic stress response is derived from the stored energy function given by (8.24), expressed in terms of the logarithms of the principal stretches. This model differs from the model described in Section 8.7 only in the underlying spectral formulation. Instead of the formulation proposed by Simo and Taylor [39], this model use the spectral formulation developed by Ogden [30]. This formulation allows more straightforward handling of repeated principal stretches.

8.7.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### quad-log (Ogden)
7     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν.

8.7.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### quad-log (Ogden)
7     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### 2D parameters
1.0   # thickness
2     # constraint option (override)
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
```

The material thickness and constraint option from Table 8.2 follow the the common parameters 8.0.2. Finally, the model requires specification of Young's modulus E and Poisson's ratio ν. This two-dimensional model is formulated in plane strain, so other values for the constraint option are silently overridden.

8.7.3 Output values

This constitutive model does not have any output values.

8.8 Material 8: small deformation J_2 plasticity

This model combines the isotropic stored energy function from Section 8.1 with a J_2 yield condition incorporating linear kinematic and isotropic hardening for infinitesimal strain analysis. This model is due to Simo and Hughes [38]. The yield condition is formulated as

$$f(\boldsymbol{\sigma}, \mathbf{q}) := ||\boldsymbol{\eta}|| - \sqrt{\frac{2}{3}} K(\alpha) \leq 0, \tag{8.27}$$

where the internal variables $\mathbf{q} = \left\{ \alpha, \bar{\boldsymbol{\beta}} \right\}$ are the equivalent plastic strain α and the center of the yield surface $\bar{\boldsymbol{\beta}}$ in deviatoric stress space. The relative stress $\boldsymbol{\eta}$ is

$$\boldsymbol{\eta} = \text{dev}\,[\boldsymbol{\sigma}] - \bar{\boldsymbol{\beta}}, \tag{8.28}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress. The associative flow rule and hardening laws are

$$\dot{\boldsymbol{\epsilon}}^p = \gamma \frac{\boldsymbol{\eta}}{||\boldsymbol{\eta}||}, \tag{8.29}$$

$$\dot{\bar{\boldsymbol{\beta}}} = \gamma \frac{2}{3} H'(\alpha) \frac{\boldsymbol{\eta}}{||\boldsymbol{\eta}||}, \tag{8.30}$$

$$\dot{\alpha} = \gamma \sqrt{\frac{2}{3}}, \tag{8.31}$$

where γ is the consistency parameter from the Kuhn-Tucker complementary conditions

$$\gamma \geq 0, \qquad f(\boldsymbol{\sigma}, \mathbf{q}) \leq 0, \tag{8.32}$$

and

$$\gamma\, f(\boldsymbol{\sigma}, \mathbf{q}) = 0. \tag{8.33}$$

The combined linear kinematic and isotropic hardening functions are defined as

$$H(\alpha) = (1 - \theta)\, \bar{H}\alpha \tag{8.34}$$

and

$$K(\alpha) = \sigma_y + \theta\bar{H}\alpha, \tag{8.35}$$

where σ_y is the initial yield stress and $\bar{H}$ is the single hardening modulus. The parameter $0 \leq \theta \leq 1$ determines the mixity of isotropic to kinematic hardening, with $\theta = 1$ indicating purely isotropic hardening.

8.8.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### small strain J2 plasticity
8     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### plasticity
0     # isotropic hardening function (0:linear)
0.25  # yield stress
0.01  # hardening parameter
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. Finally, the user defines the specification for the isotropic hardening function $K(\alpha)$, the initial yield stress σ_y, and the hardening parameter $\bar{H}$. All available hardening functions are listed in Table 8.5. An example of implementing the general cubic spline is detailed in Section 8.9.

8.8.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### small strain J2 plasticity
8     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### plasticity
0     # isotropic hardening function
0.25  # yield stress
0.01  # hardening parameter
####### 2D parameters
1.0   # thickness
2     # constraint option (override)
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. Following these parameters, the user defines the specification for the isotropic hardening function $K(\alpha)$, the initial yield stress σ_y, and the hardening parameter $\bar{H}$. All available hardening functions are listed in Table 8.5. An example of implementing the general cubic spline is detailed in Section 8.9. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis. This two-dimensional model is formulated in plane strain, so other values for the constraint option are silently overridden.

8.8.3 Output values

This constitutive model has three scalar output values listed in Table 8.3.

Table 8.3: Output values for a small strain J_2 plasticity model.

var.	**description**	**labels**
α	equivalent plastic strain	`alpha`
$\|\|\text{dev}\,[\boldsymbol{\sigma}]\,\|\|$	second invariant of the deviatoric stress	`VM`
p	pressure	`press`

8.9 Material 9: finite deformation J_2 plasticity

This model is based on a formulation due to Simo [34, 35] for finite strain J_2 plasticity. The formulation includes both linear isotropic and kinematic hardening; however, the implementation allows specification of an arbitrary isotropic hardening function, but does not include kinematic hardening. The isotropic stored energy function for this model is taken from the model described in Section 8.2. The yield condition is formulated in terms of the Kirchhoff stress $\boldsymbol{\tau}$ as

$$f(\boldsymbol{\tau}, \mathbf{q}) := ||\bar{\boldsymbol{\eta}}|| - \sqrt{\frac{2}{3}} K(\alpha) \leq 0, \tag{8.36}$$

where the internal variables $\mathbf{q} = \left\{ \alpha, \bar{\boldsymbol{\beta}} \right\}$ are the equivalent plastic strain α and the center of the yield surface $\bar{\boldsymbol{\beta}}$ in deviatoric Kirchhoff stress space. The deviatoric relative stress $\bar{\boldsymbol{\eta}}$ is

$$\bar{\boldsymbol{\eta}} = \operatorname{dev}[\boldsymbol{\tau}] - \bar{\boldsymbol{\beta}}. \tag{8.37}$$

As in the small strain case, the hardening variable is assumed to evolve as

$$\dot{\alpha} = \sqrt{\frac{2}{3}} \gamma, \tag{8.38}$$

where γ is the consistency parameter in (8.32) and (8.33). The flow rule rule and kinematic hardening law are derived assuming a multiplicative split of the deformation gradient as $\mathbf{F} = \mathbf{F}^{\mathrm{e}} \mathbf{F}^{\mathrm{p}}$. The isotropic hardening functions available for this model are described in Section 8.9.4.

NOTE: The implementation of the consistent tangent moduli for this model has not been verified. The moduli are not used in the stress-update equations, but do affect the rate of convergence for Newton-type solvers. Finite strain, J_2 plasticity is also implemented in the model described in Section 8.10.

8.9.1 3D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### finite strain J2 plasticity
 9     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### plasticity
2     # isotropic hardening function
####### spline hardening function
4     # number of points
# [alpha] [K(alpha)]
0.00  0.25
0.01  0.255
0.05  0.26
0.10  0.30
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The moduli are followed by the specification of the isotropic hardening function $K(\alpha)$. In the example, the function specification `2` indicates the general cubic spline. All available hardening functions are listed in Table 8.5. The points defining the spline follow.

8.9.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### finite strain J2 plasticity
9     # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### 2D parameters
1.0   # thickness
2     # constraint option (override)
####### plasticity
2     # isotropic hardening function
####### spline hardening function
4     # number of points
# [alpha] [K(alpha)]
0.00  0.25
0.01  0.255
0.05  0.26
0.10  0.30
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis. This two-dimensional model is formulated in plane strain, so other values for the constraint option are silently overridden. The parameters for 2D analysis are followed by the specification of the isotropic hardening function $K(\alpha)$. In the example, the function specification 2 indicates the general cubic spline. All available hardening functions are listed in Table 8.5. The points defining the spline follow.

8.9.3 Output values

This constitutive model has three scalar output values listed in Table 8.4.

8.9.4 Hardening functions

The isotropic hardening function for this model is specified by the user. The functions available are listed in Table 8.5. The input parameters for each function are described in the sections that follow. For the nonlinear hardening functions, it is possible to specify the function parameters to produce hardening behavior that is not strictly convex. The

Table 8.4: Output values for a small strain J_2 plasticity model.

var.	description	labels
α	equivalent plastic strain	`alpha`
$\lvert\lvert\bar{\boldsymbol{\beta}}\rvert\rvert$	norm of kinematic hardening	`norm_beta`
$\sqrt{\frac{3}{2}}\lvert\lvert\text{dev}\left[\boldsymbol{\tau}\right]\rvert\rvert$	second invariant of the deviatoric stress	`VM_Kirch`
p	pressure	`press`

Table 8.5: Isotropic hardening functions.

code	description
0	linear
1	linear-exponential saturation
2	general cubic spline

local Newton iteration used to compute the return mapping during plastic loading is not guaranteed to converge under these conditions.

8.9.4.1 linear The linear hardening function has the form

$$K(\alpha) = \sigma_y + K\,\alpha, \tag{8.39}$$

where σ_y is the initial yield stress and K is the hardening modulus. An example of input parameters is shown below:

```
####### linear hardening function
0.25  # initial yield stress
0.01  # hardening modulus
```

8.9.4.2 linear-exponential saturation The linear-exponential saturation hardening function has the form

$$K(\alpha) = \sigma_y + K\,\alpha + \Delta K_\infty \left(1 - \exp\left[-\frac{\alpha}{\delta}\right]\right), \tag{8.40}$$

where σ_y is the initial yield stress and K is the linear hardening modulus. The saturation hardening has a magnitude of ΔK_∞ and occurs at a characteristic equivalent plastic strain of δ. An example of input parameters is shown below:

```
####### linear-exponential saturation
0.25  # initial yield stress
0.10  # linear hardening modulus
0.05  # saturation hardening
0.05  # saturation strain
```

8.9.4.3 general cubic spline

The cubic spline hardening function has the general form

$$K(\alpha) = \begin{cases} K^{(1)}(\alpha_1) + (\alpha - \alpha_1)\, K^{(1)\prime}(\alpha_1) & \text{if } \alpha < \alpha_1, \\ K^{(i)}(\alpha) & \text{if } \alpha_i < \alpha < \alpha_{i+1} \text{ for } i = 1, \dots, n-1, \\ K^{(n)}(\alpha_n) + (\alpha - \alpha_n)\, K^{(n)\prime}(\alpha_n) & \text{if } \alpha > \alpha_n, \end{cases} \tag{8.41}$$

where the n is the number of points used to define the spline. Over each sub-interval, the spline takes the form

$$K^{(i)}(\alpha) = a_0^{(i)} + a_1^{(i)} \Delta\alpha + a_2^{(i)} \Delta\alpha^2 + a_3^{(i)} \Delta\alpha^3, \tag{8.42}$$

where $\Delta\alpha = \alpha - \alpha_i$. The coefficients $\mathbf{a}^{(i)}$ are computed from the spline points under the conditions that spline pass through the data and that the first derivative of the spline is continuous between sub-intervals. Additionally, the end conditions for the spline are specified as

$$K^{(1)\prime\prime}(\alpha_1) = K^{(n)\prime\prime}(\alpha_n) = 0 \tag{8.43}$$

to match the linear extensions assumed beyond the range of the spline data. A specific example of input to define a hardening function is shown below:

```
####### spline hardening function
4     # number of spline points
# [alpha] [K(alpha)]
0.00  0.25
0.01  0.255
0.05  0.26
0.10  0.30
```

Generally, the input has the form

```
####### spline hardening function
```
[n_{pts} = *number of spline points*]

$$\begin{array}{ll} [\alpha_1] & [K(\alpha_1)] \\ \vdots & \\ [\alpha_{n_{pts}}] & [K(\alpha_{n_{pts}})] \end{array}$$

Although not required or enforced, one typically selects $\alpha_1 = 0$ in order to define the initial yield stress specifically with $K(\alpha_1)$.

8.10 Material 10: finite deformation J_2 plasticity in principal stretches

This model combines the isotropic stored energy function from Section 8.6 with a J_2 yield condition incorporating linear kinematic and isotropic hardening for finite strain analysis. This model is due to Simo [36]. The yield condition is formulated in terms of the Kirchhoff stress $\boldsymbol{\tau}$ as

$$f(\boldsymbol{\tau}, \mathbf{q}) := ||\bar{\boldsymbol{\eta}}|| - \sqrt{\frac{2}{3}} K(\alpha) \leq 0, \tag{8.44}$$

where the internal variables $\mathbf{q} = \left\{\alpha, \bar{\boldsymbol{\beta}}\right\}$ are the equivalent plastic strain α and the center of the yield surface $\bar{\boldsymbol{\beta}}$ in deviatoric Kirchhoff stress space. The relative stress $\bar{\boldsymbol{\eta}}$ is

$$\bar{\boldsymbol{\eta}} = \operatorname{dev}[\boldsymbol{\tau}] - \bar{\boldsymbol{\beta}}. \tag{8.45}$$

As in the small strain case, the hardening variable is assumed to evolve as

$$\dot{\alpha} = \sqrt{\frac{2}{3}} \gamma, \tag{8.46}$$

where γ is the consistency parameter in (8.32) and (8.33). The flow rule rule and kinematic hardening law are derived assuming a multiplicative split of the deformation gradient. These will not be outlined here; however, the linear kinematic and isotropic hardening functions

$$H(\alpha) = (1 - \theta)\, \bar{H} \alpha \tag{8.47}$$

and

$$K(\alpha) = \tau_y + \theta \bar{H} \alpha \tag{8.48}$$

functional analogously to the small strain model described in Section 8.8. τ_y is the initial Kirchhoff yield stress and $\bar{H}$ is the hardening modulus. The parameter $0 \leq \theta \leq 1$ determines the mixity of isotropic to kinematic hardening, with $\theta = 1$ indicating purely isotropic hardening. This model is formulated in principal stretches which preserves the classical return mapping schemes from the infinitesimal theory. An important results of the spectral treatment is that the consistent tangent is symmetric even during plastic loading.

8.10.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### finite strain J2 plasticity
10    # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### plasticity
0.25  # yield stress
0.01  # hardening parameter
1.0   # kinematic/isotropic mixity
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. Finally, the user defines the initial yield stress σ_y, the hardening parameter $\bar{H}$, and the kinematic/isotropic hardening mixity parameter θ, for which $\theta = 1$ produces purely isotropic hardening.

8.10.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### finite strain J2 plasticity
10    # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### 2D parameters
1.0   # thickness
2     # constraint option (override)
####### plasticity
0.25  # yield stress
0.01  # hardening parameter
1.0   # kinematic/isotropic mixity
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis. This two-dimensional model is formulated in plane strain, so other values for the constraint option are silently overridden. Following these parameters, the user defines the initial yield stress σ_y, the hardening parameter $\bar{H}$, and the kinematic/isotropic hardening mixity parameter θ, for which $\theta = 1$ produces purely isotropic hardening.

8.10.3 Output values

This constitutive model has five scalar output values listed in Table 8.6.

Table 8.6: Output values for a finite strain J_2 plasticity model.

var.	**description**	**labels**
α	equivalent plastic strain	`alpha`
$\|\|\boldsymbol{\tau}\|\|$	second invariant of the Kirchhoff stress	`VM_Kirch`
p	pressure	`press`
$\sigma_{\max}$	maximum principal Cauchy stress	`s_max`
$\sigma_{\min}$	minimum principal Cauchy stress	`s_min`

8.11 Material 11: small deformation Drucker-Prager plasticity

This model combines the isotropic stored energy function from Section 8.1 with a pressure-dependent yield condition incorporating linear kinematic hardening for infinitesimal strain analysis. The parameters for this model follow the presentation of Regueiro and Borja [31]. The pressure-dependent yield condition is formulated as

$$f(\boldsymbol{\sigma}, \alpha_1, \alpha_2) := \sqrt{\frac{3}{2}}\,||\mathrm{dev}\,[\boldsymbol{\sigma}]\,|| + \sqrt{3}\left(\zeta + \beta\,\frac{\mathrm{tr}\,[\boldsymbol{\sigma}]}{3}\right) \leq 0, \tag{8.49}$$

where $\boldsymbol{\sigma}$ is the Cauchy stress and

$$\zeta = -\bar{\alpha} + b\,\alpha_1 + \frac{\alpha_2}{\sqrt{3}}. \tag{8.50}$$

The internal variables

$$\begin{Bmatrix} \alpha_1 \\ \alpha_2 \end{Bmatrix} = -\begin{bmatrix} K' & 0 \\ 0 & H' \end{bmatrix} \begin{Bmatrix} v^p \\ e^p \end{Bmatrix} \tag{8.51}$$

are related to the total volumetric and deviatoric plastic strains v^p and e^p, respectively, through the hardening moduli K' and H'. The material constants $\bar{\alpha}$ and β may be defined in terms of the cohesive $\bar{c}$ and friction angle $\bar{\phi}$ as

$$\bar{\alpha} = \frac{6\,\bar{c}\,\cos\bar{\phi}}{\sqrt{3}\,(3 + A\,\sin\bar{\phi})} \tag{8.52}$$

and

$$\beta = \frac{6\,\sin\bar{\phi}}{\sqrt{3}\left(3 + A\,\sin\bar{\phi}\right)}, \tag{8.53}$$

where $-1 \leq A \leq 1$. The plastic potential function is

$$\varphi(\boldsymbol{\sigma}, \alpha_1, \alpha_2) = \sqrt{\frac{3}{2}}\,||\mathrm{dev}\,[\boldsymbol{\sigma}]\,|| + \sqrt{3}\left(\zeta + b\,\frac{\mathrm{tr}\,[\boldsymbol{\sigma}]}{3}\right). \tag{8.54}$$

Associative plasticity is recovered if $b = \beta$, where b is the material dilation constant, and J_2 plasticity is recovered if $b = \beta = 0$.

8.11.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### small strain Drucker-Prager plasticity
11    # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### plasticity
0.15  # cohesive-like strength parameter
0.04  # friction-like parameter
0.0   # dilation parameter
0.0   # deviatoric hardening parameter
0.0   # volumetric hardening parameter
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The cohesion-like strength parameter parameter $\bar{\alpha}$, the friction-like parameter β, the dilatation parameter b, and the hardening moduli follow.

8.11.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### small strain Drucker-Prager plasticity
11    # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### moduli
100.0 # Young's modulus
0.25  # Poisson's ratio
####### plasticity
####### plasticity
0.15  # cohesive-like strength parameter
0.04  # friction-like parameter
0.0   # dilation parameter
0.0   # deviatoric hardening parameter
0.0   # volumetric hardening parameter
####### 2D parameters
1.0   # thickness
2     # constraint option (override)
```

After the common parameters described in Section 8.0.2, the model requires specification of Young's modulus E and Poisson's ratio ν. The cohesion-like strength parameter parameter $\bar{\alpha}$, the friction-like parameter β, the dilatation parameter b, and the hardening moduli follow. The material thickness and constraint option from Table 8.2 follow the parameters for three-dimensional analysis. This two-dimensional model is formulated in plane strain, so other values for the constraint option are silently overridden.

8.11.3 Output values

This constitutive model has four scalar output values listed in Table 8.7.

8.12 Material 12: 2D hexagonal Lennard-Jones lattice

[To do: this is a pure plane stress material] The stress response of this material is calculated from an underlying lattice structure and bonding potential by making use of the Cauchy-Born rule. The method was originally described by Born [9] as a means for estimating the theoretical strength of crystals and for assessing the stability of cubic crystal configurations subject simple deformations. The Cauchy-Born rule is a multiscale assumption about how the motion of many atoms can be related to continuum deformation measures. Under the assumption, the atoms in an crystal subject to a homogeneous deformation move according

Table 8.7: Output values for the small strain Drucker-Prager plasticity model.

var.	**description**	**labels**
α_2	deviatoric part of stress-like plastic variable	`alpha_dev`
α_1	volumetric part of stress-like plastic variable	`alpha_vol`
$\lvert\lvert \mathrm{dev}\,[\boldsymbol{\sigma}]\,\rvert\rvert$	second invariant of the deviatoric stress	`VM`
p	pressure	`press`

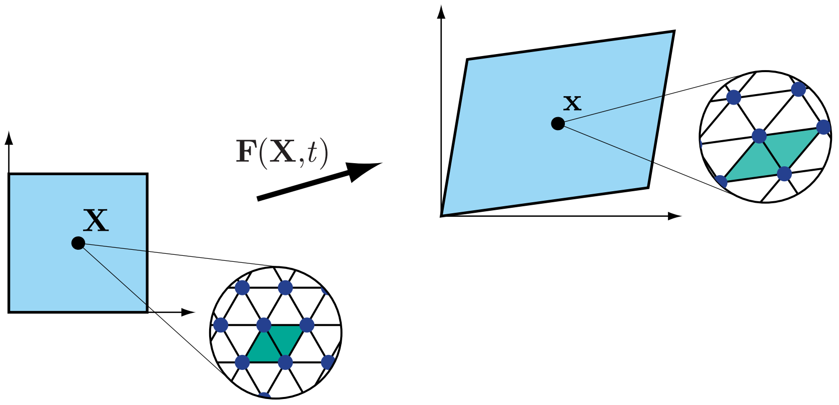


Figure 8.1: Deformation of the underlying microstructure as prescribed by the Cauchy-Born rule.

to a single mapping from the undeformed to the deformed configuration. From the continuum level, this mapping is taken to be the deformation gradient $\mathbf{F}(\mathbf{X}, t)$. As shown in Figure 8.1, the continuum region surrounding a point $\mathbf{X}$ in the undeformed configuration distorts as described by the deformation gradient. The coordinate of the continuum point in the deformed configuration is denoted by $\mathbf{x}$. At a microstructural length scale below the continuum level, we assume the deformation of an underlying crystal lattice undergoes the same homogeneous transformation. The deformation gradient at a point at the continuum length scale is assumed to be constant over a boundless crystal at the microstructural length scale. The microstructural description of a crystalline material and the corresponding continuum constitutive properties are linked using an equivalence in strain energy density. The continuum-level strain energy density is equated to the energy in a representative volume of the microstructure. Since the deformation is homogeneous, the behavior of an arbitrarily large crystal can be studied by considering only the representative volume subject to periodic boundary conditions. For crystalline materials of a single species with a primitive unit cell, this representative volume is the atomic volume. The change in energy per unit volume for a given deformation predicted by this procedure corresponds to the bulk behavior of the crystal at zero Kelvin. There are no surface or temperature-dependent effects. The atoms in the crystal interact as determined by assumed bonding potential functions. Taken from atomistic calculations, these semi-empirical relationships may be in the form of pairwise potentials, multi-body potentials, or embedded atom method potentials. The number of bonds contributing to the energy in the representative volume depends on the range of influence of the bonding potentials.

For a lattice bound by pair potentials, the strain energy density is

$$\Phi_2(\mathbf{C}) = \frac{1}{\Omega_0} \sum_{i=1}^{n_b} U^{(i)}\big(r^{(i)}(\mathbf{C})\big), \tag{8.55}$$

where $\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{F}$ is the right Cauchy-Green stretch tensor, n_b is the number of bonds, Ω_0 is the representative volume, $r^{(i)}$ is the deformed bond length, and $U^{(i)}(r)$ is a pairwise bond potential. Typically, all the bonds are assumed to be governed by the same potential function $U(r)$. Since the deformation is assumed homogeneous over the crystal, the transformation defined by $\mathbf{F}$ can be applied to vectors of finite length. For each bond in the undeformed unit cell, we define a bond vector

$$\mathbf{R} = R\,\boldsymbol{\Xi}, \tag{8.56}$$

where R and $\boldsymbol{\Xi}$ are the undeformed bond length and direction, respectively. Assuming the deformation gradient $\mathbf{F}$ is homogeneous over the representative volume, we can express the deformed bond length as

$$r(\mathbf{C}) = R\sqrt{\boldsymbol{\Xi}\cdot\mathbf{C}\,\boldsymbol{\Xi}}. \tag{8.57}$$

Given the strain energy density function (8.55), we can derive a complete description of the stress response and elastic tangent moduli using Green elastic theory [30].

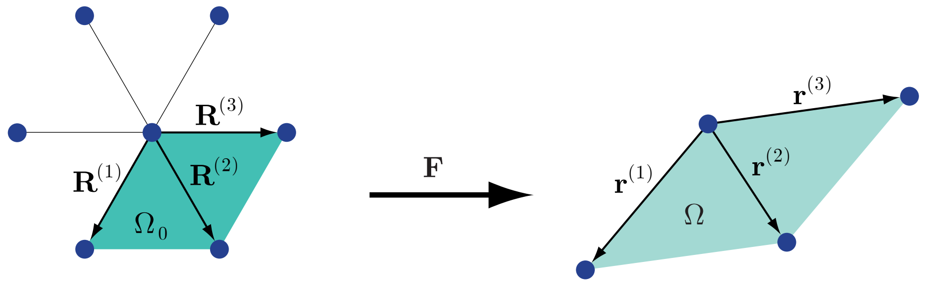


Figure 8.2: Deformation of the unit cell of a two-dimensional hexagonal lattice subject the Cauchy-Born rule.

Figure 8.2 shows the primitive unit cell of a two-dimensional, hexagonal lattice with nearest neighbor bonding for which $n_b = 3$. In the undeformed configuration, the bond vectors are given by

$$\mathbf{R}^{(1)} = a_0 \begin{Bmatrix} -1/2 \\ -\sqrt{3}/2 \end{Bmatrix} \quad \mathbf{R}^{(2)} = a_0 \begin{Bmatrix} 1/2 \\ -\sqrt{3}/2 \end{Bmatrix} \quad \mathbf{R}^{(3)} = a_0 \begin{Bmatrix} 1 \\ 0 \end{Bmatrix}, \tag{8.58}$$

where a_0 is the undeformed lattice parameter.

Following Green elastic theory, the Cauchy stress and spatial tangent modulus are given by

$$\boldsymbol{\sigma} = \sigma \mathbf{1}, \tag{8.59}$$

where the magnitude of the equibiaxial stress is

$$\sigma = \frac{3}{2}\frac{a_0}{\lambda\Omega_0} U'(\lambda a_0)\,, \tag{8.60}$$

and

$$\mathbf{c} = \mu \left[\mathbf{1} \otimes \mathbf{1} + 2\,\mathbf{I}\right], \tag{8.61}$$

where the instantaneous shear modulus is

$$\mu = \frac{3}{8\,\Omega_0}\left[a_0^2\, U''(\lambda a_0) - \frac{a_0}{\lambda}\, U'(\lambda a_0)\right]. \tag{8.62}$$

With instantaneous isotropic response under the restriction of Cauchy symmetry, the complete description of the modulus involves only the single parameter μ from (8.62). Cauchy symmetry for this two-dimensional model implies a fixed Poisson's ratio $\nu = \frac{1}{3}$.

The earliest potentials used in lattice modeling were constructed to display the essential features of cohesive interactions between atoms with parameters fitted to available experimental data, such as lattice parameters and elastic properties in the undeformed state. Born [9] employed an "inverse power" potential of the form

$$U(r) = D\frac{nm}{n-m}\left[\frac{1}{n}\left(\frac{a_0}{r}\right)^n - \frac{1}{m}\left(\frac{a_0}{r}\right)^m\right], \tag{8.63}$$

where $n > m$, D, and a_0 were chosen to match experimental data. In particular, the well-known Lennard-Jones potential results from (8.63) with $m = 6$ and $n = 12$.

8.12.1 3D input parameters

This model is formulated in two-dimensions. Other models use the Cauchy-Born rule to calculate the stress response of single crystals in three dimensions.

8.12.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

8.12.3 Output values

This constitutive model does not have any output values.

8.13 Material 13: plane strain FCC Lennard-Jones lattice

see 8.12 With instantaneous isotropic response under the restriction of Cauchy symmetry, the complete description of the modulus involves only the single parameter μ from (8.62). Cauchy symmetry for a three-dimensional lattice implies a fixed Poisson's ratio $\nu = \frac{1}{4}$.

Table 8.8: FCC lattice orientation options.

code	orientation
0	$\mathbf{e}_1$: [1 0 0] $\mathbf{e}_3$: [0 0 1]
1	$\mathbf{e}_1$: [1 0 $\bar{1}$] $\mathbf{e}_3$: [1 0 1]
2	$\mathbf{e}_1$: [1 0 $\bar{1}$] $\mathbf{e}_3$: [1 1 1]

8.13.1 3D input parameters

Although the underlying lattice for this material is three-dimensional, it has only be implemented for plane strain deformation. Other models use the Cauchy-Born rule to calculate the stress response of single crystals in three dimensions.

8.13.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

8.13.3 Output values

This constitutive model does not have any output values.

8.14 Material 14: FCC lattice with EAM potentials

The stress response of this model is calculated by applying the Cauchy-Born, described in Section 8.12, rule together with embedded atom method potentials. Application of the Cauchy-Born rule dates back to the work of Born [9]. More recently, Tadmor *et al.* [43], used the method with the embedded atom method potentials as a means to derive nonlinear, hyperelastic constitutive models for use in numerical simulations to predict stress and deformation. The embedded atom method [11] (EAM) was developed in an attempt to incorporate more physical understanding into the bond potential functions. Originally motivated by the view of metals as nuclei in a sea of electrons, the total energy of any atom in the crystal is comprised of a contribution due to the nuclear repulsion and an embedding energy representing the attraction of the nucleus to the background electron density. In fitting a set of EAM potentials to a particular material, one must define three functions: a pair potential describing nuclear interactions, the electron density distribution around each nucleus, and a function describing the interaction between a nucleus and the background electron density. Although quite different from simple pair interactions, EAM potentials may be incorporated into the procedures for deriving constitutive properties without any modifications to the assumptions of the Cauchy-Born rule.

Using EAM potentials, the strain energy density has two contributions,

$$\Phi_{\mathrm{EAM}} = \Phi_2 + \Phi_e, \tag{8.64}$$

where Φ_2 is the contribution from the nuclear interactions given by the pair potential expression (8.55) and Φ_e is the embedding energy. This additive split in the total energy results in additive contributions to the stress response and tangent moduli. Therefore, all expressions derived from (8.55) still apply, and we only need to discuss the additional contribution due to the embedding energy. The contribution to the total energy (8.64) due to the embedding energy may be written as

$$\Phi_e = \frac{1}{\Omega_0} U_e(\overline{\rho}_e) \tag{8.65}$$

where $\overline{\rho}_e$ is the background electron density, that is, the electron density at the nucleus of a particular atom resulting from all neighbors of the atom, but not the atom itself. As a first approximation, the electron density is assumed to be centrosymmetric

$$\overline{\rho}_e = \sum_{i=1}^{n_b} \rho_e\big(r^{(i)}\big)\,. \tag{8.66}$$

Table 8.9: Embedded atom method potentials.

code	**potential**
0	Al: Ercolessi and Adams [12]
1	Al: Voter and Chen [48]
2	Cu: Voter and Chen [47]
3†	Au: Foiles, Baskes, and Daw [?]

† This potential requires the file auu3 with the potential data in ParaDyn [?] format.

[To do: print tables of modulus values and units for each potential.]

8.14.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

8.14.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

8.14.3 Output values

This constitutive model does not have any output values.

8.15 Material 15: Stillinger-Weber diamond cubic

The cubic unit cell for the diamond cubic structure is shown in Figure 8.3. The unit cell

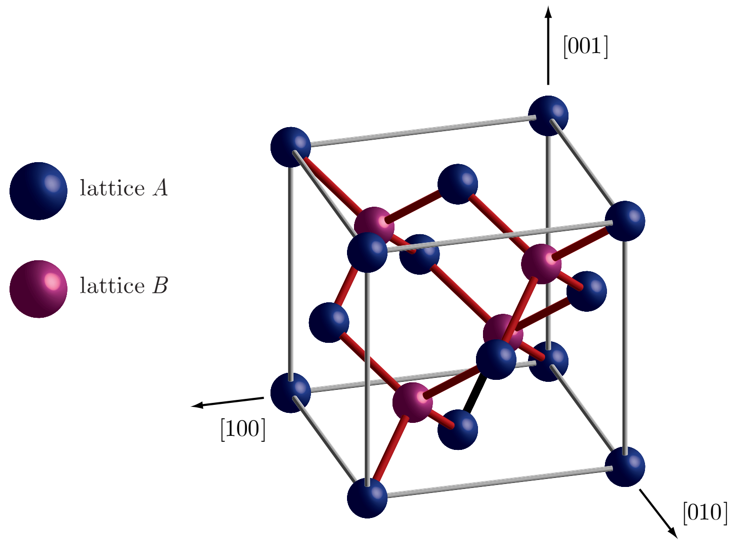


Figure 8.3: Cubic unit cell of the diamond cubic lattice.

can be constructed from two interlaced face-centered cubic lattices. The lattice B in the figure is positioned relative to lattice A by a displacement of $\{1/4, 1/4, 1/4\}$ of the cube edge dimensions. Stakgold[41] discussed how such structure, which Wiener[49] called a composite lattice, should be treated, though he limits his discussion to pair potentials and his approach does not strictly guarantee objectivity of the resulting strain energy density function. In general terms, symmetry constraints require that each constituent sub-lattice deforms homogeneously, but the sub-lattices can move relative to each other by a rigid body translation which depends on the interatomic potentials. The rigid body motion is determined by minimizing the energy of the crystal as a function of the translational degrees of freedom. By enforcing this condition, static equilibrium is satisfied for all atoms in the lattice.

Keeping in mind that the degrees of freedom will be introduced in a manner which maintains objectivity, we can express a modified strain energy density function as

$$\Phi(\mathbf{C}) = \check{\Phi}(\mathbf{C}, \boldsymbol{\Xi}(\mathbf{C}))\,, \tag{8.67}$$

where $\boldsymbol{\Xi}$ represents a vector of all the degrees of freedom of $\check{\Phi}$. The degrees of freedom are implicit functions of the stretch $\mathbf{C}$. Given a strain energy density function of the form (8.67), we can apply the standard relations from Green elastic theory to derive expressions for the stress response and tangent moduli. The internal degrees of freedom are added as

$$\mathbf{x} = \mathbf{F}\left(\mathbf{X} + \boldsymbol{\Xi}\right) \tag{8.68}$$

Though perhaps not the most accurate for all types of simulations, the original potentials by Stillinger and Weber [42] are still highly attractive due to their simplicity. With these potentials, the strain energy density is composed of two contributions,

$$\Phi_{\mathrm{SW}} = \Phi_2 + \Phi_3. \tag{8.69}$$

Here, Φ_2 represents the contribution from standard inverse-power pair potentials, and Φ_3 represents the contribution from three-body potentials which are needed to stabilize the tetrahedral structures that compose the lattice. For a range of deformations around the undeformed state, these potentials involve only nearest neighbor bonds. In the original formulation[42], the two-body potential is given by

$$U_2(r) = \begin{cases} A\left(B\, r^{-p} - r^{-q}\right) \exp\left[\frac{\delta}{r - r_{cut}}\right] & r < r_{cut} \\ 0 & r \geq r_{cut}, \end{cases} \tag{8.70}$$

and the three-body potential is

$$\begin{aligned} &U_3\big(r^{(ji)}, r^{(jk)}, \theta^{(ijk)}\big) \\ &\quad = \begin{cases} \lambda\left(\frac{1}{3} + \cos\theta^{(ijk)}\right)^2 \exp\left[\frac{\gamma}{r^{(ji)} - r_{cut}}\right] \exp\left[\frac{\gamma}{r^{(jk)} - r_{cut}}\right] & r^{(ji)}, r^{(jk)} < r_{cut} \\ 0 & r^{(ji)}, r^{(jk)} \geq r_{cut}. \end{cases} \end{aligned} \tag{8.71}$$

Both potentials make use of an exponential cut-off term which causes the potentials, and all derivatives of the potentials, to vanish smoothly when the atomic spacing r reaches r_{cut}. This cut-off term is advantageous because it clearly defines the distance over which atoms interact and provides a smooth termination for all the potentials through to the second derivatives required to compute the moduli. U_3 (8.71) is clearly designed to favor the "ideal" tetrahedral angle of $\theta_t = \cos^{-1}(-1/3)$, though the functional form is not motivated by any detailed attributes of the sp^3 orbitals which produce this configuration. In terms of (8.70) and (8.71), we can express the strain energy density (8.69) as

$$\Phi_{\mathrm{SW}} = \frac{1}{\Omega_0}\left[\sum_{(ij)=1}^{8} U_2\big(r^{(ij)}\big) + \sum_{(ijk)=1}^{12} U_3\big(\tilde{\mathbf{r}}^{(ijk)}\big)\right], \tag{8.72}$$

where we represent the arguments to the three-body term as a vector composed of the length of two bond vectors and the cosine of the angle they subtend,

$$\tilde{\mathbf{r}}^{(ijk)} = \left\{\begin{matrix} r^{(ji)} \\ r^{(jk)} \\ \cos\theta^{(ijk)} \end{matrix}\right\}. \tag{8.73}$$

Table 8.10: Diamond cubic lattice orientation options.

code	orientation
0	$\mathbf{e}_1$: [1 0 0] $\mathbf{e}_3$: [0 0 1]
1	$\mathbf{e}_1$: [1 0 $\bar{1}$] $\mathbf{e}_3$: [1 0 1]
2	$\mathbf{e}_1$: [1 0 $\bar{1}$] $\mathbf{e}_3$: [1 1 1]

[To do: with and without internal equilibration.]

8.15.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

8.15.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

8.15.3 Output values

This constitutive model does not have any output values.

8.16 Material 16: Virtual Internal Bond model

The general features of the VIB model, as well as more detailed study of its localization behavior under conditions of dynamic crack propagation, are described in greater detail elsewhere[13, 19, 20, 21]. The VIB model is developed within the framework of hyperelasticity. The current, or deformed, configuration of a body is described as $\mathbf{x} = \boldsymbol{\varphi}(\mathbf{X})$, by a mapping $\boldsymbol{\varphi}$ of the undeformed configuration $\mathbf{X}$. The arrangement of cohesive interactions among material particles is described by a spatial bond density function. The strain energy density is computed by integrating the bond density in space in a continuous analog to the sum over discrete lattice neighbors for the case of crystalline materials. The VIB form of the strain energy density function is

$$\Phi = \frac{1}{\Omega_0} \int\limits_{\Omega_0^*} U(l)\, D_\Omega \, d\Omega, \tag{8.74}$$

where Ω_0 is the undeformed representative volume, l is the deformed virtual bond length, $U(l)$ is the bonding potential, D_Ω is the volumetric bond density function, and Ω_0^* is the integration volume defined by the range of influence of U.

[To do: show results for the fracture energy, cohesive strength, and modulus in terms of potential, and then show specific results for the Smith-Ferrante potential. Is the potential selectable?]

Unlike constitutive models based on the Cauchy-Born rule for crystalline materials described in Section 8.14, the VIB model may suffer significant numerical errors when evaluating integrals of the bond distribution to determine the strain energy density (8.74) and the resulting stress response and moduli. Many different schemes could be developed for evaluating the VIB model integrals. As with any numerical procedure, the optimal scheme represents a compromise between accuracy and speed of execution. The primary concern with the implementation of the isotropic VIB model is that the response in numerical calculations is indeed lacking in any orientational dependence resulting from the integration procedure. For the plane stress, isotropic VIB model, the integral of the volumetric bond density function reduces to an integral over the interval $-\pi \leq \phi \leq \pi$. A simple scheme for evaluating these integrals is shown in Figure 8.4. The uniform bond distribution is sampled

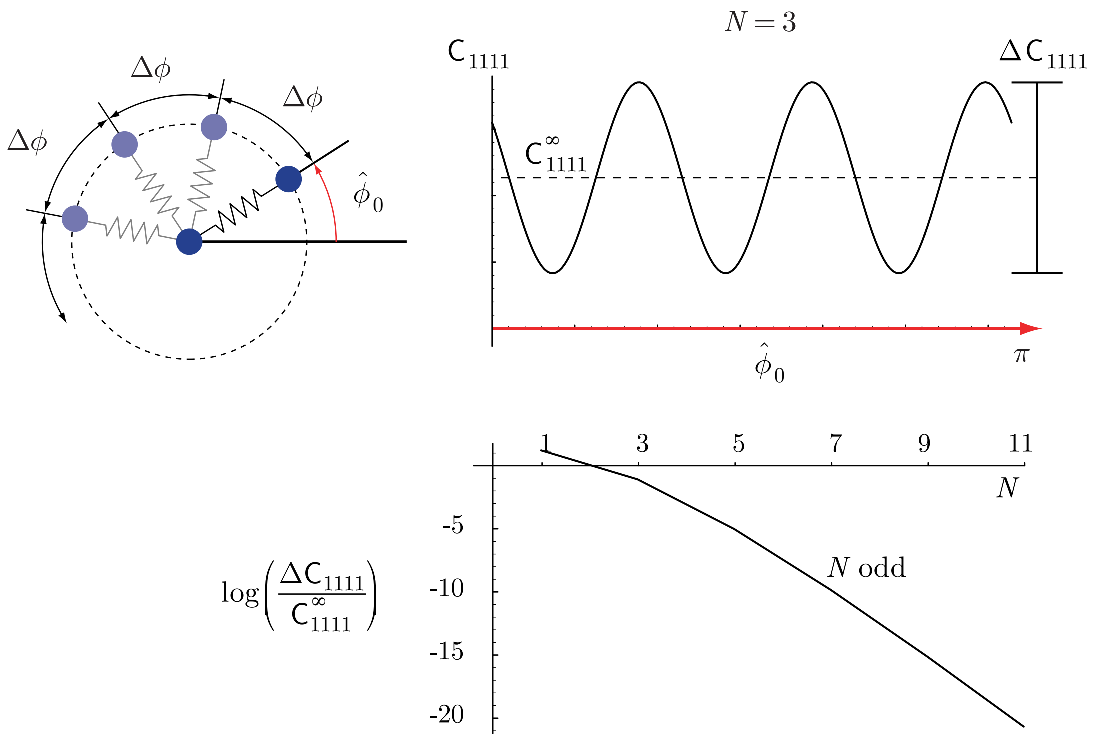


Figure 8.4: Convergence to isotropic response in the numerical implementation of the plane stress, isotropic VIB model.

at fixed intervals $\Delta\phi$. The resulting values of the stress and modulus should be independent of the orientation of the "microstructure", represented by the offset angle $\hat{\phi}_0$ in positioning the integration points. Clearly, the value of the VIB integrals for the stress and modulus will not be independent of $\hat{\phi}_0$ if an insufficient number of sampling points is used. The variation of the C_{1111} component of the spatial tangent modulus for an arbitrarily chosen state of deformation is shown in Figure 8.4 for the case of $N = 3$ sampling points. As is evident from the figure, the value displays a six-fold symmetric variation. The $N = 3$ configuration corresponds to the bonding arrangement in a two-dimensional, close-packed lattice. This is the lattice arrangement on the close-packed planes of the face-centered cubic (FCC) lattice and on the basal plane of the hexagonal close-packed (HCP) lattice. For calculations at infinitesimal deformations, these planes are commonly assumed to be isotropic. However, only the instantaneous response of the undeformed lattice lacks orientational dependence. For deformations at finite strains, this crystal plane displays six-fold symmetric anisotropy. Figure 8.4 also shows how the numerical response converges toward isotropy as the number of integration points is increased. The magnitude of the variation in the modulus $\Delta\mathsf{C}_{1111}$ relative to the "exact" value for the applied deformation $\mathsf{C}^{\infty}_{1111}$ drops to below machine precision with only 11 integration points. Therefore, the computational effort to produce essentially isotropic response in the numerical implementation does not present significant difficulties. The evenly-spaced integration point arrangement seems to be a highly efficient scheme for evaluating the VIB integrals. For a given number of points, the results using integration schemes based on evenly-spaced points display much smaller dependencies on orientation than do the results using integration schemes based on Gaussian quadrature.

For the three-dimensional isotropic VIB model, the task of generating isotropic response is significantly more difficult. This case requires evaluating the VIB integrals over a sphere. Unlike the two-dimensional case, there are no schemes for arranging an arbitrary number of points on a sphere in a evenly-spaced manner. The icosahedron with its 20 equilateral triangular faces presents a starting point from which more dense arrangements of points can be constructed. Integration rules with 80 and 320 points in arrangements which are symmetric about the "equator" are shown in Figure 8.5. These arrangements are created by

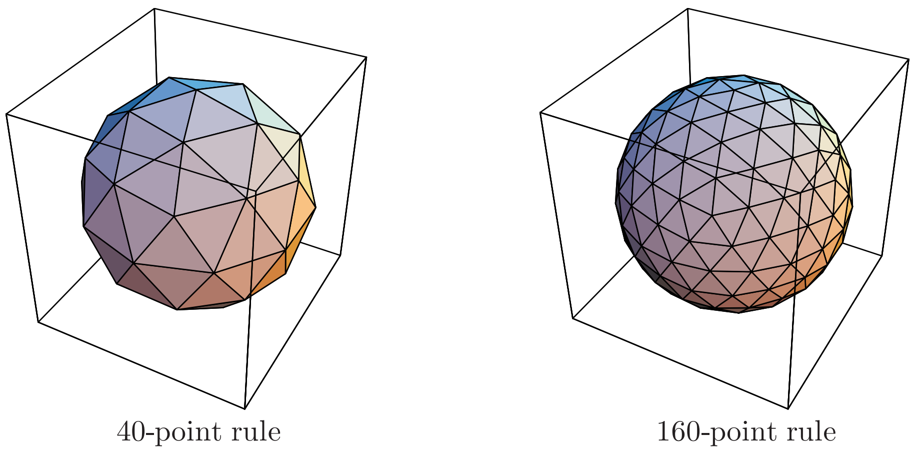


Figure 8.5: Integration point arrangements in three dimensions based on the icosahedron. *[To do: get picture of lat-long points.]*

successively bisecting the edges of the triangular facets and projecting the vertex points onto the sphere. The center of each facet is used as an integration point with a weight selected to produce the correct result when integrating a constant over the surface of the sphere. For a rotation about an arbitrarily selected axis, the variation in the C_{1111} component of the spatial tangent modulus is shown in Figure 8.6 for the integration schemes depicted

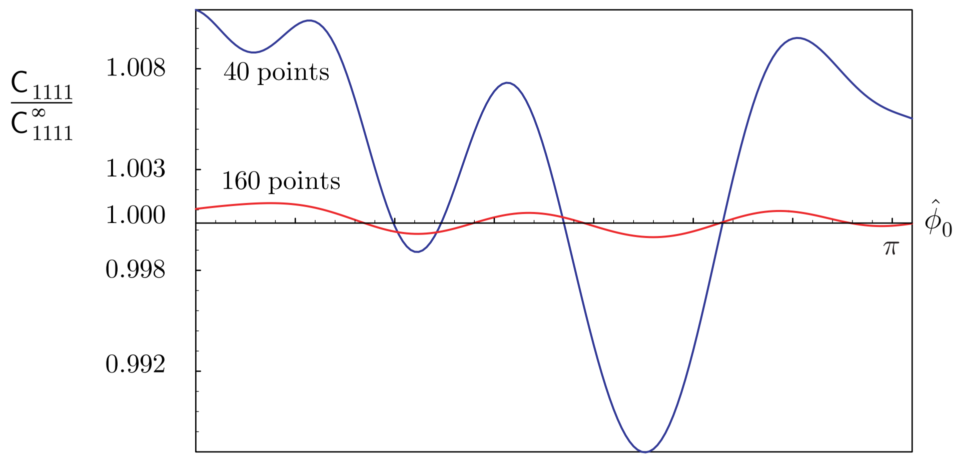


Figure 8.6: Orientational dependence of C_{1111} for the three-dimensional isotropic VIB model in a deformed state.

in Figure 8.5. The figure shows that the modulus displays noticeable anisotropic behavior although a much larger number of points is used than would be required in two dimensions to produce essentially no error. Bažant[6] has developed more sophisticated schemes for numerical integration over the surface over of a sphere, but even these schemes produce a significant increase in computational effort with the extension from two to three dimensions.

8.16.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### VIB
16    # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### potential parameters
1     # potential code
4.775 # A
0.0643# B
####### integration parameters
1     # integration rule
40    # number of integration points
```

8.16.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

```
####### VIB
16    # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### 2D parameters
1.0   # thickness
1     # constraint option (override)
####### potential parameters
1     # potential code
4.775 # A
0.0643# B
####### integration parameters
0     # integration rule
11    # number of integration points
```

8.16.3 Output values

This constitutive model does not have any output values.

8.17 Material 17: isotropic Virtual Internal Bond model (Simo)

This model combines the VIB strain energy density from Section 8.16 with the spectral formulation developed by Simo and Taylor [39]. Expressed in principal stretches, the model displays exactly isotropic response regardless of the accuracy of the scheme used to integrate the bond distribution. *[To do: However, the model will not display the modulus, fracture energy, or cohesive strength developed from analysis of the model.]*

8.17.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

8.17.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

8.17.3 Output values

This constitutive model does not have any output values.

8.18 Material 18: isotropic Virtual Internal Bond model (Ogden)

This model combines the VIB strain energy density from Section 8.16 with the spectral formulation developed by Ogden [30]. Expressed in principal stretches, the model displays exactly isotropic response regardless of the accuracy of the scheme used to integrate the bond distribution. *[To do: However, the model will not display the modulus, fracture energy, or cohesive strength developed from analysis of the model.]*

8.18.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

8.18.2 2D input parameters

An example of input parameters for two-dimensional analysis is shown below:

8.18.3 Output values

This constitutive model does not have any output values.

8.19 Material 19: isotropic Virtual Internal Bond model with J_2 plasticity

This model combines the VIB strain energy density from Section 8.16 with the spectral formulation developed by Simo and Taylor [39] and the finite strain framework of J_2 plasticity from Simo [36] described in Section 8.10. Expressed in principal stretches, the model displays exactly isotropic response regardless of the accuracy of the scheme used to integrate the bond distribution. *[To do: However, the model will not display the modulus, fracture energy, or cohesive strength developed from analysis of the model.]*

The yield condition is formulated in terms of the Kirchhoff stress $\boldsymbol{\tau}$ as

$$f(\boldsymbol{\tau}, \mathbf{q}) := ||\boldsymbol{\eta}|| - \sqrt{\frac{2}{3}} K(\alpha) \leq 0, \tag{8.75}$$

where the internal variables $\mathbf{q} = \left\{\alpha, \bar{\boldsymbol{\beta}}\right\}$ are the equivalent plastic strain α and the center of the yield surface $\bar{\boldsymbol{\beta}}$ in deviatoric Kirchhoff stress space. The relative stress $\boldsymbol{\eta}$ is

$$\boldsymbol{\eta} = \operatorname{dev}[\boldsymbol{\tau}] - \bar{\boldsymbol{\beta}}. \tag{8.76}$$

As in the small strain case, the hardening variable is assumed to evolve as

$$\dot{\alpha} = \sqrt{\frac{2}{3}}\, \gamma, \tag{8.77}$$

where γ is the consistency parameter in (8.32) and (8.33). The flow rule rule and kinematic hardening law are derived assuming a multiplicative split of the deformation gradient. These will not be outlined here; however, the linear kinematic and isotropic hardening functions

$$H(\alpha) = (1 - \theta)\, \bar{H} \alpha \tag{8.78}$$

and

$$K(\alpha) = \tau_y + \theta \bar{H} \alpha \tag{8.79}$$

functional analogously to the small strain model described in Section 8.8. τ_y is the initial Kirchhoff yield stress and $\bar{H}$ is the hardening modulus. The parameter $0 \leq \theta \leq 1$ determines the mixity of isotropic to kinematic hardening, with $\theta = 1$ indicating purely isotropic hardening.

8.19.1 Output values

[To do: output values]

8.80 Material 80: ABAQUS/Standard UMAT BCJ

This model makes of the the ABAQUS/Standard [16] UMAT interface in Tahoe. This model is due to Bammann, Chiesa, and Johnson [?]. The ABAQUS/Standard UMAT interface in Tahoe parses ABAQUS-format input. The specifications for UMAT's in Tahoe may be taken directly from an ABAQUS Standard input deck.

Like all material models, the materials using the ABAQUS/Standard UMAT interface may be used for analysis types not supported by ABAQUS/Standard. Namely, the materials may be applied to dynamic analysis using explicit time integration. Using this time integration scheme, the UMAT's will calculate both the stress and modulus, unless specifically rewritten to detect whether both or just one is actually needed for the analysis. Unmodified UMAT's will perform unnecessary calculations, leading to longer simulations times. This model requires an installation of Tahoe that include the f2c [?] module.

8.80.1 3D input parameters

An example of input parameters for three-dimensional analysis is shown below:

```
####### ABAQUS/Standard BCJ
80    # material code
####### common material parameters
0.0    0.0    1.0
0      0.0
####### ABAQUS input deck
*****************************************
*MATERIAL, NAME=A356
**
** user subroutine is dmg1.f using metric units
** material name = alum  356
** creation date = 08/08/96
** data fit by   =  Jim Lathrop for uscar A356 temp-strain rate
*USER MATERIAL, CONSTANTS=52
**
**G, a,Bulk,b,melttemp,C1, C2, C3
** C4, C5, C6, C7, C8, C9, C10, C11
**C12,C13,C14,C15,C16,C17, C18, C19
**C20,Ca,Cb,init.temp,heat gen.coeff,void growth exp,initial rad,tors constant a
**nuc const b,nuc const c,nuc coeff,fract.tough,part.size,part.vol.fract,cd1,cd2
**
 2.592E+04 1.000E+00 6.763E+04 0.000E+00 5.556E+03 5.309E+01 9.453E+02 1.559E+02
 1.105E+02 1.000E-05 0.0000000 1.128E-03 -1796.200 4.820E+03 1.094E+01 2.385E-03
 1.441E+03 1.674E-03 0.000E+00 2.818E+03 4.622E+00 0.000E+00 0.000E+00 0.000E+00
 0.000E+00 -5.000000 -0.389490 297.00000 0.0000000 0.3000000 0.0002000 615460.00
 58640.000 30011.000 86.600000 17.300000 4.000E-06 0.0700000 4.5000000 40.000000
 20.00E-00 20.00E-00 0.0509000 0.0010000 0.0000000 0.0000000 0.0000000 0.0000000
 0.0000000 0.0000000 0.0035000 0.0090000
*DEPVAR
 25
```

[To do: This is for A356]

8.80.2 2D input parameters

The parameters for using the *[To do: same as for three-dimensional analysis. Since this model is formulated in three-dimensions, the thing will be plane strain when used for two dimensional analysis. Mention this above in the general description of the UMAT interface.]*

8.80.3 Output values

[To do: output values determined by the implementation of the UMAT. Show output parameters for A356.]

9 Constitutive models for cohesive surface elements

This section of the guide describes the constitutive models that may be used as material models specifying the traction vector as a function of the displacement discontinuity for the cohesive surface elements described in Section 7.5.2. Each of the following subsections provides some background information about its respective material model followed by a description and example of the input required for analysis. All cohesive surface models have been implemented in two dimensions, and most also have three-dimensional implementations. All surface materials support implicit and explicit analysis; in other words, all models implement consistent tangent matrices that produce quadratic convergence near local minima when used with the Newton-type, nonlinear solution algorithms (see Section 5.2.2).

Table 9.1: Constitutive models for cohesive surface elements

code	**description**
0	Xu-Needleman (elastic)
1	Tvergaard-Hutchinson (elastic)
2	Linear Damage
3	Tvergaard-Hutchinson (elastic + viscous dissipation)
4	Tijssens (elastic/viscoplastic)
5	trilinear rate-dependent
6	“Tied” potential (elastic)
7	Yoon, Allen, and Searcy (linear viscoelastic)
8	thermomechanical model (not implemented)
9	rate-based ductile fracture model
10	elastoplastic model for geomaterials
11	rigid-plastic model for geomaterials

9.0.4 Background

The Tahoe input for the cohesive surface elements described in Section 7.5.2 specify the traction-separation relation(s) for a given cohesive surface element block by specifying a list of material codes ranging from 0 to 7 along with any required model parameters. In the following sections, “Surface Material x” refers to the constitutive relation specified by a material code of “x” for a given element block. In these constitutive models, the displacement discontinuity between two opposite crack faces is quantified by the gap vector $\mathbf{\Delta}$ whose components Δ_t and Δ_n in two dimensions are specified for directions along and normal to the midplane between the two crack faces. In three dimensions, the components are Δ_{t1},

Δ_{t2}, and Δ_n. Each constitutive model described below returns the traction vector $\mathbf{T}$ as a function of $\boldsymbol{\Delta}$ and any internal state variables $\{\mathbf{q}_i\}$ the model has. Also, many of these models specify length scales $\delta_i, i = n, t$ in the normal and tangential directions directions ($i = n, t1, t2$ in 3D), and one defines a dimensionless vector $\boldsymbol{\lambda}$ related to the gap vector $\boldsymbol{\Delta}$ via

$$\lambda \equiv \frac{\Delta_i}{\delta_i}. \tag{9.1}$$

With this notation many models define the traction vector as a function of the λ_i or indeed of the scalar opening parameter $\lambda \equiv \sqrt{\boldsymbol{\lambda} \cdot \boldsymbol{\lambda}}$.

In addition, many of these cohesive models have a penalty stiffness mulitplier that stiffens the material under compression in order to improve the poor behavior of cohesive surface elements in this regime. The penalty stiffness multiplier is a single number multiplying the natural stiffness that would be calculated by the cohesive surface model.

9.1 Surface Material 0: Xu-Needleman (elastic)

This model due to Xu and Needleman [50] is based on the universal binding energy curve and exhibits complete reversibility; consequently, the traction vector can be derived from a potential $\Phi(\boldsymbol{\Delta})$ via

$$\mathbf{T} = \frac{\partial \Phi}{\partial \boldsymbol{\Delta}}. \tag{9.2}$$

For the Xu-Needleman model, Φ is a function of the dimensionless gap vector $\boldsymbol{\lambda}$ described above and is given by

$$\Phi(\boldsymbol{\Delta}) = \phi_n(1 + e^{-\lambda_n})(1 - r + \lambda_n)\frac{1-q}{r-1} - e^{-\lambda_t^2}(q + \frac{r-q}{r-1}\lambda_n). \tag{9.3}$$

Here ϕ_n is an input parameter specifying the work to fracture and one has the identity

$$\phi_n = \int \frac{\partial \phi}{\partial \boldsymbol{\Delta}} \cdot \delta \boldsymbol{\Delta}. \tag{9.4}$$

Two additional dimensionless parameters appearing in the expression for Φ are $q = \frac{\phi_t}{\phi_n}$ relating the normal to shear fracture energies and $r = \frac{\Delta_n^*}{\delta_n}$ which defines the normal opening Δ_n^* after pure shear separation at zero normal traction. The exponential decay of this potential implies an infinite range of interaction between two crack faces, but the Tahoe implementation of this model truncates the potential at a distance $r_{fail}\,\delta_n$. A plot of the normal traction as a function of Δ_n for pure mode I loading is shown in Figure **??** .

9.1.1 Input parameters

The Xu-Needleman model is implemented for both two- and three-dimensionsal simulations, and the input parameters are independent of the dimensionsality of the problem. An example of these input parameters is shown below:

```
####### Xu Needleman
0     # surface material code
####### model parameters
1     # q
0     # r
0.1   # dn
0.1   # dt
1.0   # phi_n
100.0 # r_fail
10.0  # penetration stiffness multiplier
```

9.1.2 Output values

This constitutive model does not have any output values.

9.2 Surface Material 1: Tvergaard-Hutchinson (elastic)

As with the Xu-Needleman model, the model of this section due to Tvergaard and Hutchinson [45] is compeletely reversible and uses the dimensionless gap vector $\boldsymbol{\lambda}$ to characterize the opening displacement. Unlike the Xu-Needleman model, the Tvergaard-Hutchinson model first specifies a potential density ϕ shown in Figure **??** and given by

$$\phi(\boldsymbol{\Delta}) = \left\{ \begin{array}{rcl} \sigma_{max} \frac{\lambda}{\Lambda_1} & : & \lambda < \Lambda_1 \\ \sigma_{max} & : & \Lambda_1 < \lambda < \Lambda_2. \\ \sigma_{max} \frac{1-\lambda}{1-\Lambda_2} & : & \Lambda_2 < \lambda < \Lambda_{fail} \end{array} \right. \tag{9.5}$$

The potential Φ can be obtained from the potential density via

$$\Phi = \delta_n \int_0^{\lambda} \phi(\tilde{\lambda}) \; \mathrm{d}\tilde{\lambda}, \tag{9.6}$$

and the traction may be concisely expressed as

$$T_i = \frac{\partial \Phi}{\partial \Delta_i} = \frac{\delta_n}{\delta_i} \Lambda_i \phi(\lambda). \tag{9.7}$$

9.2.1 Input parameters

This cohesive model is implemented in two- and three-dimensions, and it should be noted that the zero-stiffness plateau in the traction-separation relation when $\Lambda_1 \neq \Lambda_2$ can lead to poorer-than-quadratic or convergence or no convergence at all in certain problems. An example of the input parameters for the Tvergaard-Hutchinson model follows:

```
####### Tvergaard-Hutchinson
1     # material code
####### model parameters
100.0 # sigma
0.01  # dn
0.01  # dt
0.1   # Lambda_1
0.9   # Lambda_2
1.0   # Lambda_fail (set to 1 if lambda_fail < 1)
0.    # penetration stiffness multiplier
```

9.2.2 Output values

This constitutive model has a single nodal output value; it prints the value of λ.

9.3 Surface Material 2: Linear Damage Model

This model is not currently implemented in three-dimensions.

9.3.1 Input parameters

An example of input parameters for the linear damage model is shown below:

```
####### linear damage code
2     # surface material code
####### model parameters
0.01  # dn
0.01  # dt
0.0   # penetration stiffness multiplier
```

9.3.2 Output values

This constitutive model does not have any output values.

9.4 Surface Material 3: Tvergaard-Huthcinson with viscous dissipation

This model generalizes the Tvergaard-Hutchinson model described in Section 9.2 to include the effects of viscous dissipation. Consequently, this model does not exhibit reversability upon unloading, and may not be completely described by a potential. The traction **T** is now a function of $\dot{\mathbf{\Delta}}$, the rate of change of the gap vector, as well as a function of $\mathbf{\Delta}$ itself. It may be expressed as

$$\mathbf{T} = \mathbf{T}^{tv} + \eta_0(1 - \lambda)\dot{\mathbf{\Delta}} \tag{9.8}$$

where $\mathbf{T}^{tv}$ denotes the traction as a function of $\mathbf{\Delta}$ specified by the Tvergaard-Hutchinson model in Equation (9.7).

9.4.1 Input parameters

Sample input parameters are shown below:

```
####### Tvergaard-Hutchinson with viscous dissipations
3     # surface material code
####### model parameters
100.0 # sigma
0.01  # dn
0.01  # dt
0.10  # lambda_1
0.90  # lambda_2
1.0   # lambda_fail (set to 1 if lambda_fail < 1)
0.01  # eta_0
0.0   # penalty stiffness multiplier
```

9.4.2 Output values

This constitutive model outputs nodal values of λ and the energy dissipated by the viscous term in the traction-separation relation.

9.5 Surface Material 4: Tijssens' crazing model (elastic-plastic)

This model is due to Tijssens *et al.* [44], and models crazing in amorphous polymers. This cohesive law has three distinct regimes corresponding to elastic deformation until craze initiation, plastic deformation as the craze fibrils lengthen, and finally failure as the crazes lose their load-carrying capacity. The elastic regime is linear elastic with traction-separation relation

$$T_i = k_i \Delta_i \tag{9.9}$$

where the normal and tangential stiffnesses $k_i; i = n, t$ are parameters of the model and summation over repeated indices is not implied. Elastic deformation continues until the stress state in the bulk material surrounding the cohesive layer satisfies a yield condition. For mode I loading, the yield condition reduces to specifying a critical normal traction for craze initiatiation. For more general loading, the stress tensor may be sampled in the surrounding bulk, and the initiation criterium is given by a function of T_n and the bulk stress tensor σ

$$f(\sigma, T_n) = -\frac{1}{2}\sigma_{ii} - \frac{1}{2}A_\theta - \frac{B_\theta}{2\sigma_{ii}} - T_n = 0 \tag{9.10}$$

The parameters A_θ and B_θ are temperature dependent according to

$$X_\theta = X_0 e^{-\frac{Q_X}{k_B\theta}}, X = A, B. \tag{9.11}$$

After craze initiation, the traction-separation relation is specified in rate form as

$$\dot{T}_i = k_i(\dot{\Delta}_i - \dot{\Delta}_i^{cr}) \tag{9.12}$$

The components Δ_i^{cr} represent the span of craze fibrils between the surfaces of the cohesive layer and evolve according to

$$\dot{\Delta}_i = \dot{\Delta}_i^0(e^{-\frac{K^*}{k_B\theta}(\tau_i - T_i)} - \delta_{it}e^{-\frac{K^*}{k_B\theta}(\tau_t - T_t)}). \tag{9.13}$$

Here, $k_B\theta$ is Boltzmann's constant multiplied by the absolute temperature (assumed constant throughout the simulation), K^* and the τ_i are input parameters governing the kinetics of craze growth, and the Kronecker delta, δ_{it}, is included to ensure that T_t is an even function of Δ_t. It is assumed that $\tau_n = \sqrt{3}\tau_t$ and $\dot{\Delta}_t^0 = \sqrt{3}\dot{\Delta}_n^0$ in analogy to von Mises' stresses. Also, the normal elastic stiffness k_n is constant while the tangential elastic stiffness k_t softens according to

$$k_t = k_{t0}\, e^{-c_t\Delta_n^{cr}/\Delta^{crit}}. \tag{9.14}$$

Once the craze width Δ_n^{cr} reaches a critical opening Δ^{crit}, the craze is assumed to have failed and lost its load-carrying capacity. In the Tahoe implementation of this model, the traction is reduced to zero over n_{fail} timesteps with tangential and normal stiffnesses given by k_t and k_n, respectively. The result for pure mode I loading at constant rate yields the traction-separation curves shown in Figure **??**. The hardening behavior is seen when $\dot{\Delta}_n < \dot{\Delta}_n^0$; otherwise softening behavior is observed.

9.5.1 Input Parameters

The input parameters for this model are as shown below:

```
####### Tijssens
4     # surface material code
####### model parameters
100.0 # k_t^0
100.0 # k_n
50.0  # decay constant for k_t^0
0.005 # critical craze opening width
1.0   # A_0 - mixed-mode initiation parameter
1.0   # B_0 - mixed-mode initiation parameter
1.0   # Q_A - mixed-mode initiation parameter
1.0   # Q_B - mixed-mode initiation parameter
0.001 # deltadot_n^0 - should be of order the opening rate, Delta_dot
10.0  # tau_n
1.0   # K^star
273.0 # absolute temperature
1     # integer > 0 giving bulk element group to obtain stress from
10    # integer number of timesteps n_fail
```

9.5.2 Output Values

This cohesive model outputs four nodal values. The first is the local normal opening rate $\dot{\Delta}_n$. The second is the tangential craze width, Δ_t^{cr}, and the third is the normal component of the same, Δ_n^{cr}. The final output parameter is the initiation function $f(\sigma, T_n)$.

9.6 Surface Material 5: trilinear rate dependent

This model is a generalization of the Tvergaard-Hutchinson model of Section 9.2; its primary distinction is that the critical normal length scale δ_n varies as a function of the normal opening rate $\dot{\delta}_n$. The material is assumed to yield when it first reaches the plateau traction (for pure mode I loading) σ^m, and at this yield point δ_n is set as a function of rate given by

$$\delta_n = \delta_n^b + \delta_n^m \log \dot{\Delta}_n \tag{9.15}$$

where log denotes the natural logarithm. Before the critical normal traction is reached, the normal length scale is rate-independent and given as an input paramter. Once the normal length scale has been set for a particular integration point, the parameter is finalized there and that point is no longer sensitive to variations in the local opening rate.

A second distinction between this trilinear model and Tvergaard and Hutchinson's is that the traction-separation relation is no longer constrained to have a plateau (region of zero stiffness); there is an additional parameter m_p allowing nonzero stiffness and better numerical performance. The traction on the plateau is determined by m_p via

$$\mathbf{T}(\lambda) = \mathbf{T}(\Lambda_1)(1 + m_p(\lambda - \Lambda_1)), \tag{9.16}$$

where Λ_1 has the same meaning as in the Tvergaard-Hutchinson model of Section **??**. Likewise, $\mathbf{T}(\Lambda_1)$ is idential to that determined by Surface Material 1 since the loading rate effects the plastic failure portion of the model and not the elastic loading portion.

9.6.1 Input parameters

An example of the input parameters for the trilinear, rate-dependent model is shown below:

```
####### rate-dependent Tvergaard-Hutchinson
5     # surface material code
####### model parameters
100.0 # sigma
0.25  # rate-independent normal length scale, dn
0.25  # dt
0.1   # Lambda_1
0.9   # Lambda_2
1.0   # Lambda_fail (set to 1 if lambda_fail < 1)
0.0   # penalty stiffness multiplier
1.0   # delta_n^b (intercept of rate function)
1.0   # delta_n^m (slope or rate functions)
0.0   # m_p
```

Setting “delta_n_b” equal to “d_n” and “delta_n_m” and “m_p” to zero in the input file reduces this model to the rate-independent Tvergaard-Hutchinson one of Section 9.2.

9.6.2 Output values

This model outputs nodal values for λ, each component of the local opening rate $\dot{\mathbf{\Delta}}$, and the rate-dependent normal length scale δ_n.

9.7 Surface Material 6: Tied Potential

This material surface model works in conjunction with the TiedNodes KBC Controller 3, and its behavior is unpredictable unless the TiedNodes controller is activated. This codependency is used in an attempt to remedy some shortcomings of cohesive surface models. To begin with, cohesive surface models are in general designed to model the loss of a material's load-carrying capacity across a crack face, yet by necesity the traction-separation relation must also specify the response of this region of material to loading before this capacity is assumed to have been lost. Consequently, there is *a priori* no relation between the bulk material and the initial (presumably) elastic loading portion of the traction separation relation. The tied nodes KBC Controller constrains the two faces separated bridged by a cohesive surface to admit no displacement discontinuity across them until an initiation criterium is reached, and consequently, the cohesive surface element is inactive until its nodes are "untied" at initiation. To this end, the tied potential provides a function TiedPotentialT::InitiationQ that determines if a critical traction has been reached for yield to occur and for the material to exhibit cohesive failure. Although a minimal initiation function is provided and will be described below, this function should in general be re-implemented by the end user. Since the material response is that of the bulk until the nodes are untied, this model should provide greater numerical stability in situations where the cohesive layer may reasonably be expected to undergo severe compression or other deformations for which a cohesive surface element may be ill-equipped.

Currently, two traction-separation laws may be used after the nodes are untied. The first is the Xu-Needleman model of Section 9.1 and the second is the Tvergaard-Hutchinson model of Section 9.2. The Xu-Needleman implementation begins the traction separation law at the peak stress of $\phi_n/(e\delta_n)$ in the notation of Section 9.1, so that the materials yield criterium is taken to be one of maximum normal traction.

The Tvergaard and Hutchinson tied-potential implementation allows the user to specify the initiation point in terms of the scalar opening parameter λ and the model parameters given in Section 9.2. More concretely, a given value of λ may be inverted to determine a specific traction that acts as the initiation traction. Since the initiation traction is not constrained to be the maximum traction, this choice of tied potential is possibly more robust than the Xu-Needleman choice, but it is plagued by poorer numerical performance.

9.7.1 Input parameters

The tied potential input parameters contain a flag that is set to zero to choose the Tvergaard-Huthcinson model and to one to choose the Xu-Needleman model. Examples of both sets of input will be shown.

Below is shown sample input for a tied Xu-Needleman model:

```
####### Tied potential
6     # surface material code
0. 1. # normal direction to sample stress to obtain traction
####### surface material code for Xu-Needleman
0
####### model parameters
1.0   # q
0.0   # r
0.001 # dn
0.001 # dt
10.0  # phi
10.0  # r_fail
```

For the tied Tvergaard-Hutchinson model:

```
####### Tied potential
6     # surface material code
0. 1. # normal direction to sample stress to obtain traction
####### surface material code for Tvergaard-Hutchinson
1
####### model parameters
100.0 # sigma
0.001 # dn
0.001 # dt
0.25  # lambda_1
0.75  # lambda_2
1.0   # lambda_3
0.25  # lambda_0, initiate at T = T(lambda_0)
```

Although the input parameters have been ordered to correspond exactly with the ones for their respective untied cohesive laws, it should be noted that there are no penalty stiffness multipliers for interpenetration, and that the mode I formalism currently used overrides any effects in the Xu-Needleman specification other than $q = 1$, $r = 0$.

9.7.2 Output values

The tied potentials ouput a single nodal value that reflects a given nodes status as tied ($\neq 0$) or untied (0.) and may be useful in conjunction with visualization tools.

9.8 Material 7: Yoon, Allen, and Searcy (linear viscoelastic)

This model presented initially by Yoon and Allen and later by Searcy and Allen models a cohesive layer as a linear viscoelastic material that undergoes damage as it loses its load-carrying capacity. An undamaged material will respond exactly as a linear viscoelastic material, while a material undergoing damage will require the specificaton of a damage evolution law. The details of the physical interpretation of the damage parameter α are given in Reference [51], and the traction-separation law is given as

$$\mathbf{T} = \boldsymbol{\lambda}(1-\alpha(t)) \left[T_0 + \int_0^t \left(E_\infty + \sum_i^{n_e} (1 - E_i e^{-\frac{t}{\tau_i}}) \right) \frac{\partial \lambda(\tau')}{\partial \tau'} \,\mathrm{d}\tau' \right] \tag{9.17}$$

where the E_i and τ_i are elastic moduli and relaxation times corresponding to relaxation process i, and E_∞ is the equilibrium elastic modulus as for a Prony series in a standard linear solid. The model as published contains an initiation traction T_0, but calculations using the model set T_0 to zero, the same convention currentely used in Tahoe. There is a choice of damage evolution laws where $\dot{\alpha}$ is specified as a function of λ, $\dot{\lambda}$, and α itself. In all the damage evolution laws, it is assumed that α does not evolve for closing cohesive surfaces, i.e when $\dot{\lambda}$ is non-negative.

The first damage law is given in Reference [1] as

$$\dot{\alpha} = \alpha_0 \lambda^m, \tag{9.18}$$

and the response of this evolution law to cyclic loading is shown in Figure . A second evolution law is given in Reference [51] as

$$\dot{\alpha} = \alpha_0 (1-\alpha^n)(1-b\lambda^m), \tag{9.19}$$

and the response of this law to identical cyclic loading as before is shown in Figures . A third and for the moment final damage evolution law is given by

$$\dot{\alpha} = \alpha_0 \dot{\lambda}^m, \tag{9.20}$$

and its response to cyclic loading is shown in Figure .

9.8.1 Input parameters

An example of input parameters for two- or three-dimensional analysis is shown below:

```
######## Yoon-Allen-Searcy
7       # potential code
0.      # fsigma_0
1.      # d_c^t - critical tangential length scale
1.      # d_c^n - critical normal length scale
10.     # E_infinity - equilibrium modulus
2       # number of elements in prony series (nprony)
10.     # first relaxation modulus
10.     # first relaxation time
100.    # second relaxation modulus
2.      # second relaxation time
 .
 .
 .
100.    # nprony_th relaxation modulus
100.    # nprony_th relaxation time
1       # damage evolution law code
        # 1: alpha_dot = alpha_0*l^alpha_exp
        # 2: alpha_dot =
        #     ((alpha_0(1-alpha)^alpha_exp)(1-lambda_0)^lambda_exp)
        # 3: alpha_dot = alpha_0*lambda_dot^alpha_exp
2.      # alpha_exp
10.     # alpha_0
#-3.    # lambda_exp # only for damage code 2
#10.    # lambda_0  # only for damage code 2
10.     # penetration stiffness multiplier
```

9.8.2 Output values

This constitutive model has three nodal output values. It outputs values of λ, $\dot{\lambda}$, and α.

10 Geometry and results files

The user input consists of analysis parameters and a geometry specification. Analysis parameters are specified in a plain text file. Several options exist for the specification of the geometry. Table 10.1 lists the file types available to specify the geometry. These will be described in greater detail in Section 10.1. Table 10.2 lists the file types available for the output of results. Note that some of the file types may be used for both input and results, while others are limited to one or the other.

Table 10.1: Geometry input file formats.

code	**description**
0	TahoeI
1	TahoeII
3	EnSight [10] Gold ASCII version 6
4	EnSight Gold Binary version 6
5	ExodusII [33]
6	ABAQUS ASCII [?]
7	ABAQUS Binary
10	PATRAN [28]

Table 10.2: Result output file formats.

code	**description**
0	TahoeI
2	TecPlot [2] version 7
3	EnSight [10] Gold ASCII version 6
4	EnSight Gold Binary version 6
5	ExodusII [33]
6	ABAQUS ASCII [?]
7	ABAQUS Binary
10	PATRAN [28]

10.1 Geometry file formats

Input data is allowed to be one of three database formats listed in Table 10.1:

(1) TahoeI: With this format, geometry data may be supplied embedded in the parameters file or in files separate from, or external to, the parameters files. When specified in external files, the block of geometry data in the parameters file is replaced by the name of the external file where the data resides. The format of the data itself is the same in both cases.

(2) TahoeII: This format allows random access to the geometry data which is stored in one or more plain text files.

(3) ExodusII: This format allows random access to the geometry data which is stored in platform independent binary file.

All Tahoe plain text input files are white space delineated, that is, values are separated by white space, but the arrangement of white space is ignored. The # symbol denotes that the remainder of a given line of text should be ignored, or treated as a comment.

10.1.1 TahoeI format

The TahoeI format includes specifications for model geometry as well as kinematic and force boundary conditions. Each of these input types is described in greater detail in the sections that follow. All of the input types precede the actual input data with some indication of the dimensions of the data. When beginning to read a block of input data, Tahoe will determine whether the block begins with an integer or with a character string. A leading integer denotes the dimensions of an embedded block of input data. If Tahoe finds a character string, it assumes the string is the name of an external file in which the input data resides. Tahoe cannot distinguish file names beginning with an integer from the dimensions of an embedded data block.

10.1.1.1 Coordinate list The format of coordinate data is shown below:

`# dimensions`

$[n_{nd} = \textit{number of points}]$

$[n_{sd} = \textit{number of spatial dimensions}]$

`# coordinates`

$$\begin{array}{llll} \texttt{1} & [x_{11}] & \dots & [x_{1n_{sd}}] \\ \vdots & & & \\ [n_{nd}] & [x_{n_{nd}1}] & \dots & [x_{n_{nd}n_{sd}}] \end{array}$$

The coordinate of each point is preceded by its index. Coordinates may appear in any order; however, Tahoe does not verify that all n_{nd} points have been assigned coordinates exactly once.

10.1.1.2 Element block

The format of element block data is shown below:

```
# dimensions
```
[n_{el} = *number of elements*]
[n_{en} = *number of nodes per element*]
```
# connectivities
```
1 [n_{11}] ... [$n_{1n_{en}}$]
$\vdots$
[n_{el}] [$n_{n_{el}1}$] ... [$n_{n_{el}n_{en}}$]

The connectivity of each element is preceded by its index. Connectivities may appear in any order; however, Tahoe does not verify that all n_{el} elements have been assigned connectivities exactly once.

10.1.1.3 Node set

The format of node set data is shown below:

```
# dimension
```
[n_{nd} = *number of nodes*]
```
# list
```
[n_1] ... [$n_{n_{nd}}$]

10.1.1.4 Side set

The format of side set data is shown below:

```
# dimension
```
[n_s = *number of facets*]
```
# facet list
```
[e_1] [s_1]
$\vdots$
[e_{n_s}] [s_{n_s}]

Facets are specified by two numbers, the element number e and the side s. The side numbering convention for the various integration domain geometries are shown in Figures 7.1 through 7.6.

10.1.2 TahoeII format

The TahoeII format allows random access to geometry data in a plain text file format. The file format uses the keywords listed in Table 10.3. The keywords must preceded by an asterisk (`*`). Aside from the keyword `set`, each keyword denotes a separate type of data. These keywords may appear in any order in a TahoeII file. However, since the file is scanned whenever data is requested, it is more efficient to place longer blocks of data toward the end

Table 10.3: Keywords for the TahoeII file format.

`version`	`title`
`dimensions`	`nodes`
`elements`	`nodesets`
`sidesets`	`set`

of the file. Alternately, blocks of data may be specified in files separate from the primary TahoeII file. When specified in an externally, the block of geometry data in the file is replaced by the path of the external file where the data resides. The format of the data itself is the same in both cases.

A TahoeII file contains a single set of coordinates and an arbitrary number of element sets, node sets, and side sets. The format of the sets is similar to the TahoeI format described in Section 10.1.1. The sets are identified by an ID. ID's may be any positive integer and do not need to be consecutive. ID's for each type of set are stored separately. As mentioned above, TahoeII files are plain text and white space delineated, that is, values are separated by white space, but the arrangement of white space is ignored. The `#` symbol denotes that the remainder of a given line of text should be ignored, or treated as a comment. The current TahoeII version is `1.0`. The format of each type of data in a TahoeII file is described in the sections that follow.

10.1.2.1 Version The `version` keyword is used to check the compatibility of the database with the current version of the TahoeII code. The current version is `1.0`. An example of this section is shown below:

```
# TahoeII file version
*version 1.0
```

10.1.2.2 Title This section begins with the `version` keyword. A one line title may be assigned to the database:

```
# database title
*title
```
[*one line title of 254 characters or less*] (10.1)

10.1.2.3 Dimensions This section begins with the `dimensions` keyword. It contains the dimensions of the nodal coordinate data, and declares the ID and dimension of each element, node, and side set. This sequence of data in this section must follow the order shown in the example below:

```
# dimensions and IDs
*dimensions
# dimension of coordinate data
$[n_{nd} = \textit{number of nodes}]$
$[n_{sd} = \textit{number of spatial dimensions}]$
# element set declarations
$[n_{es} = \textit{number of element sets}]$
 $[ID_1]$   $[n_{el1} = \textit{number of elements}]$   $[n_{en1} = \textit{number of element nodes}]$
   $\vdots$
$[ID_{n_{es}}]$              $[n_{el\,n_{es}}]$                              $[n_{en\,n_{es}}]$
# node set declarations
$[n_{ns} = \textit{number of node sets}]$
 $[ID_1]$   $[n_{nd1} = \textit{number of set nodes}]$
   $\vdots$
$[ID_{n_{ns}}]$              $[n_{nd\,n_{ns}}]$
# side set declarations
$[n_{ss} = \textit{number of side sets}]$
 $[ID_1]$   $[n_{s1} = \textit{number of facets}]$
   $\vdots$
$[ID_{n_{ss}}]$             $[n_{s\,n_{ss}}]$
```

If the number of element sets n_{es}, node sets n_{ns}, or side sets n_{ss} is zero, the subsequent list of ID's and dimensions is empty.

10.1.2.4 Node sets This section begins with the `nodesets` keyword. The node sets must appear in the order they have been specified in the dimensions section 10.1.2.3. The beginning of the data for each node set is marked by the `set` keyword. The data for each set is comprised of the number of nodes in the set and list of nodes themselves. This data for each set may be defined in line or in an external file. If the data is defined in an external file, the path to the file replaces the data itself in the TahoeII file. In either case, the format of the data itself is the same. The format of each node set specification is the same as the

TahoeI format for node sets 10.1.1.3. An example appears below:

```
# node set specifications
*nodesets
*set
```
[n_{nd1} = *number of nodes in set* 1]

[n_1] ... [$n_{n_{nd1}}$]

$\vdots$

```
*set
```
[$n_{nd\,n_{ns}}$ = *number of nodes in set* n_{ns}]

[n_1] ... [$n_{n_{nd\,n_{ns}}}$]

The set subscripts for the nodes in each set have been omitted for clarity. Note that the dimensions of the set are from the from the dimensions section are repeated but that the ID of each set is not repeated. When specifying the data for a node set in an external file, the primary TahoeII contains the path to the external file, as shown below:

`*set` [*path to external file*]

10.1.2.5 Side sets This section begins with the `sidesets` keyword. The side sets must appear in the order they have been specified in the dimensions section 10.1.2.3. The beginning of the data for each side set is marked by the `set` keyword. The data for each set is comprised of the number of each in the set and list of facet specifications themselves. Facets are specified by two numbers, the element number e and the side s. The side numbering convention for the various integration domain geometries are shown in Figures 7.1 through 7.6. This data for each set may be defined in line or in an external file. If the data is defined in an external file, the path to the file replaces the data itself in the TahoeII file. In either case, the format of the data itself is the same. The format of each side set specification is the same as the TahoeI format for side sets 10.1.1.4. An example appears below:

```
# side set specifications
*sidesets
*set
```
$[n_{s1} = \textit{number of facets in set } 1]$

$[e_1] \quad \ldots \quad [s_1]$

$\vdots$

$[e_{n_{s1}}] \quad \ldots \quad [s_{n_{s1}}]$

$\vdots$

```
*set
```
$[n_{s\,n_{ss}} = \textit{number of facets in set } n_{ss}]$

$[e_1] \quad \ldots \quad [s_1]$

$\vdots$

$[e_{n_{s\,n_{ss}}}] \quad \ldots \quad [s_{n_{s\,n_{ss}}}]$

The set subscripts for the element and sides in each set have been omitted for clarity. Note that the dimensions of the set are from the from the dimensions section are repeated but that the ID of each set is not repeated. When specifying the data for a side set in an external file, the primary TahoeII contains the path to the external file, as shown below:

$$\texttt{*set} \;\; [\textit{path to external file}] \tag{10.2}$$

10.1.2.6 Element sets This section begins with the `elements` keyword. The element sets must appear in the order they have been specified in the dimensions section 10.1.2.3. The beginning of the data for each element set is marked by the `set` keyword. The data for each set is comprised of the number of each in the set and list of facet specifications themselves. This data for each set may be defined in line or in an external file. If the data is defined in an external file, the path to the file replaces the data itself in the TahoeII file. In either case, the format of the data itself is the same. The format of each element set specification is the same as the TahoeI format for element sets 10.1.1.2. An example appears below:

```
# element set specifications
*elements
*set
```
[n_{el1} = *number of elements in set* 1]

$$
\begin{array}{llll}
\texttt{1} & [n_{11}] & \ldots & [n_{1n_{en1}}] \\
\vdots & & & \\
[n_{el1}] & [n_{n_{el1}1}] & \ldots & [n_{n_{el1}n_{en1}}]
\end{array}
$$

$\vdots$

```
*set
```
[$n_{el\,n_{es}}$ = *number of elements in set* n_{es}]

$$
\begin{array}{llll}
\texttt{1} & [n_{11}] & \ldots & [n_{1n_{en\,n_{es}}}] \\
\vdots & & & \\
[n_{el\,n_{es}}] & [n_{n_{el\,n_{es}}1}] & \ldots & [n_{n_{el\,n_{es}}n_{en\,n_{es}}}]
\end{array}
$$

The set subscript on the nodes in each element has been omitted for clarity. The connectivity of each element is preceded by its index. Connectivities may appear in any order; however, Tahoe does not verify that all n_{el} elements have been assigned connectivities exactly once. Note that the dimensions of the set are from the from the dimensions section are repeated but that the ID of each set is not repeated. When specifying the data for a side set in an external file, the primary TahoeII contains the path to the external file, as shown below:

`*set` [*path to external file*]

10.1.2.7 Nodal coordinates The coordinate specification is marked by the `nodes` keyword. The format of coordinate data is the TahoeI format described in Section 10.1.1.1. An example is shown below:

```
# nodal coordinates
*nodes
# dimensions
```
[n_{nd} = *number of points*]

[n_{sd} = *number of spatial dimensions*]

```
# coordinates
```

$$
\begin{array}{llll}
\texttt{1} & [x_{11}] & \ldots & [x_{1n_{sd}}] \\
\vdots & & & \\
[n_{nd}] & [x_{n_{nd}1}] & \ldots & [x_{n_{nd}n_{sd}}]
\end{array}
$$

The coordinate of each point is preceded by its index. Coordinates may appear in any order; however, Tahoe does not verify that all n_{nd} points have been assigned coordinates exactly once. Coordinate data may be specified in the primary TahoeII file or in an external file. When specifying the data for a side set in an external file, the primary TahoeII contains the path to the external file, as shown below:

```
# nodal coordinates
*nodes  [path to external file]                    (10.3)
```

10.1.3 EnSight Gold format

EnSight Gold [10] files can be read in version 6 Gold plain text or binary format. The EnSight binary files are portable between some platforms. When an EnSight database is read, the user should enter the case file name, including the extension, when prompted for a database filename.

10.1.4 ExodusII/Genesis format

The ExodusII [33] finite element geometry and results file format was developed at Sandia National Laboratories. This file format is export-controlled, and therefore is not available to all users. A given database should contain a coordinate list, element blocks, and any node set or sides sets references from the parameters file. Tahoe does not read material data from the ExodusII database. Node and element maps are currently ignored and items are assumed to be consecutively numbered. To refer to a specific set or block in the database use the ID value. The blocks and sets do not need to have consecutive ID values.

10.1.5 ABAQUS Results Files format

ABAQUS [?] results files in ASCII or binary (*.fil/*.fin) are provided as an input option. Typically as an input option for translating database information. The Tahoe interface supports 5.8 and 6.2 versions and maybe others, depending upon version variations. A given database should contain a coordinate list, element list, node sets, and element sets. Nodes and elements do not have to be consecutively numbered. ABAQUS results files have sets for both elements and nodes. The toolbox interface only reads the first eight characters of the set name. No spaces are allowed. ABAQUS does not have database records pertaining to side sets.

10.1.6 MSC PATRAN Neutral format

MSC PATRAN [28] neutral files are created using the export command within PATRAN. A given database should contain a coordinate list, element list, named component groups that contain element blocks and/or node sets. Nodes and elements do not have to be consecutively numbered. To refer to an element block or node set, use the group name. PATRAN group names are created from the Group pull down menu and are typically used for posting different layers. They are exported to a Neutral file as Named Components. When writing the Neutral file, PATRAN will truncate the group name to 12 characters and write it in upper case. The group names must not contain spaces. The Neutral files do not have anything explicitly pertaining to side sets, but if desired, a pseudo side set could be established using contact surface boundaries.

10.1.7 Tahoe Results format

Tahoe results files are the output files written in Tahoe format. This format differs from Tahoe I or Tahoe II input file format and contains variable results. When prompted for the database filename, the user should supply the geometry (*.geo) or results (*.run) file including the extension.

10.2 Results file formats

Output data is associated with an output group. An output group is not necessarily the same as the element group. Element groups can have more than one connectivity set for post-processing purposes. Some element groups have no output. These connectivity sets are not the element blocks listed in the parameters file. For instance, to visualize an internal 3D crack, those elements that represent the crack (localized elements or cohesive surfaces) are separated into an additional connectivity within the element group. This allows post-processors to easily show the crack without having to create isosurfaces or other body representations.

Variable output data is either associated with a node or an element. Nodal data are variable values defined at the nodes within the output group. Element data are variables values associated with the elements. Each element variable has one value per element. All element quadrature data is projected to the nodes using the smoothing algorithm proposed by Zienkiewicz [?].

For the purposes of writing output files, the output print increment is a value starting at one and is consecutively incremented. Thus the print increment is not the same as the computational increments used by the program. This accounts for changes in the frequency with which computational increments are printed and some I/O databases demand consecutive numbering.

Depending upon the type of output database chosen, different output files are created. File extensions are listed in Table 10.4. The log/output file is always created and contains an echo of the input data. If multiple loading sequences are used, the extensions for geometry and variable data will be lengthened to include sequence numbers. EnSight uses the variable labels as file extensions on its variable files. For ExodusII and TecPlot, the variable labels are used as variable names. The nodes are broken up into those used by the connectivities. Node numbers coincide with the internal consecutive node numbering. As described in Section 2.2, Tahoe has two output modes for parallel execution. Tahoe can join the results data from all the processors at run time, or the distributed results data may be joined during a separate

Table 10.4: File extensions for results files.

file type	**extension**	**description**
plain text	`.out`	input echo/runtime log
	`.io`[*group*]`.fracture`	fracture surface area
	`.nd`[*node*]`.hst`	nodal history
	`.rs`[*step*]	restart data
TahoeI	`.geo`	output geometry
	`.run`	output results
TecPlot	`.io`[*group*]`.dat`	output results
EnSight	`.io`[*group*]`.case`	case file
	`.io`[*group*]`.geo`	output geometry
	`.io`[*group*]`.ps`[*step*]`.geo`	changing output geometry
	`.io`[*group*]`.ps`[*step*]`.`[*label*]	output variable
ExodusII	`.io`[*group*]`.exo`	output results
	`.io`[*group*]`.ps`[*step*]`.exo`	changing geometry output

post-processing step. With distributed output files, the extension `.p`[*processor*] is prepended to each of the extensions listed in Table 10.4.

10.2.1 TecPlot

The TecPlot [2] files are currently written in version 7.5 plain text format. Element data is ignored and not written to an output file. Element data must be averaged or projected to the nodes. TecPlot plain text files can be converted to binary files, which are generally portable, using the PrePlot executable supplied by TecPlot.

10.2.2 EnSight

EnSight [10] files are written in version 6 Gold plain text or binary format. The EnSight binary files are generally portable between some platforms. Users should conduct a preliminary test to verify this portability. Version 6 Gold files are readable by EnSight version 6 Gold and by EnSight version 7 or later. The EnSight case file is written at each print increment to account for unexpected machine crashes. Each output group is treated as a part and is written to a separate geometry and case file. Separate files are produced so only the part that changes will be printed at each print increment. Thus reducing the amount of file space needed for large meshes. Element and node tags are not given. Any vector variable internally named with the {`_X`, `_Y`, `_Z`} convention will be written to a vector variable file instead of a scalar variable file.

10.2.3 ExodusII

ExodusII files are portable to all platforms that have the library installed on them. Each block within an output group is treated as a block. Separate files are written for each output group. Separate files are produced so only the block that changes will be printed at each print increment. Separate files are also produced so that nodal data from different output groups does not need to be merged. Element numbering and ordering maps are not given. The nodes are broken up into those used by the connectivities. ExodusII files, from multiple restarts, can be joined together using the `conex` executable.

10.2.4 ABAQUS Results

ABAQUS results files are created for version 5.8. These files could be used in Post. Each block withan an output group is treated as an element set. Separate files are written for each output group.

10.2.5 PATRAN Results

PATRAN results files are created following the format for version 2001. These files could be used in Post. Each block withan an output group is treated as a Named Component. For each output group, a geometry file, displacement file, force file, node variable file and element variable file is written (if variable data exists).

10.2.6 Fracture surface area

The fracture surface area file contains columnar data, in the order listed as follows:

(1) time

(2) fracture surface area

10.2.7 Node history output

The history node file contains columnar data, in the order listed as follows.

(1) time

(2) displacement vector

(3) velocity vector (if dynamic or transient)

(4) acceleration vector (if dynamic)

(5) force vector (if prescribed DOF exists)

11 History of updates to the Tahoe User Guide

11.1 History

[To do: brief history:
originated by Stephanie Wimmer
updated to Version 3.4 by Patrick Klein.]

11.2 Changes since Version 3.4

This section summarizes changes to the Tahoe input format since Version 3.4.

11.2.1 update 3.4.1

Two additional output variables have been added to the element output for solid deformation elements listed in Table 7.4. This brings the number of element output flags from 5 to 7. This change is version aware. Only 5 flags are read if the input file version is older than 3.4.1.

References

[1] D.H. Allen and C.R. Searcy, *A micromechanical model for a viscoelastic cohesive zone*, International Journal of Fracture **107** (2001), 159–176.

[2] Amtec Engineering, Inc., *TecPlot User Manual*, 8.0 ed., 1999.

[3] C. Argento, A. Jagota, and W.C. Carter, *Surface formulation for molecular interactions of macroscopic bodies*, Journal of the Mechanics and Physics of Solids **45** (1997), 1161–1183.

[4] C. Ashcraft, D. Pierce, D.K. Wah, and J. Wu, *The reference manual for* ***SPOOLES***, Boeing Shared Services Group, 2.2 ed., 1999.

[5] S. Atluri and T. Zhu, *A new meshless local Petrov-Galerkin approach in computational mechanics*, Computational Mechanics **22** (1998), 117–127.

[6] Z.P. Bažant, *Efficient numerical integration on the surface of a sphere*, Zeitschrift für Angewandte Mathematik und Mechanik **66** (1986), 37–49.

[7] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl, *Meshless methods: an overview and recent developments*, Computer Methods in Applied Mechanics and Engineering **139** (1996), 3–47.

[8] T. Belytschko, Y.Y. Lu, and L. Gu, *Element-free Galerkin methods*, International Journal for Numerical Methods in Engineering **37** (1994), 229–256.

[9] M. Born, *On the stability of crystals,* I., Proceedings of the Cambridge Philosophical Society **36** (1940), 160–172.

[10] Computational Engineering International, Inc., *EnSight User Manual*, 6.2 ed., 1998.

[11] M.S. Daw and M. Baskes, *Embedded atom method:derivation and application to impurities, surfaces, and other defects in metals*, Physical Review B **29** (1984), 6443–6453.

[12] F. Ercolessi and J.B. Adams, *Interatomic potentials from first-principals calculations: the force-matching method*, Europhysics Letters **28** (1994), 583–588.

[13] H. Gao and P. Klein, *Numerical simulation of crack growth in an isotropic solid with randomized internal cohesive bonds*, Journal of the Mechanics and Physics of Solids **46** (1998), 187–218.

[14] R.A. Gingold and J.J. Monaghan, *Smoothed particle hydrodynamics: theory and application to non-spherical stars*, Monthly Notices of the Royal Astronomical Society **181** (1977), 375–389.

[15] J.-H. Heegaard and A. Curnier, *An augmented Lagrangian method for discrete large-slip contact problems*, International Journal for Numerical Methods in Engineering **36** (1993), 569–593.

[16] Hibbit, Karlsson & Sorenson, Inc., *ABAQUS/Standard*, 5.6 ed., 1996.

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

[18] S.A. Hutchinson, J.N. Shadid, and R.S. Tuminaro, *Aztec user's guide version 1.1*, Tech. Report SAND95-1559, Sandia National Laboratory, 1995.

[19] P. Klein and H. Gao, *Crack nucleation and growth as strain localization in a virtual-bond continuum*, Engineering Fracture Mechanics **61** (1998), 21–48.

[20] P.A. Klein, *A virtual internal bond approach to modeling crack nucleation and growth*, Ph.D. thesis, Stanford University, 1999.

[21] P.A. Klein and H. Gao, *Study of crack dynamics using the Virtual Internal Bond model*, James R. Rice 60th Anniversary Volume of Solid Mechanics and its Applications (G.M.L. Gladwell, ed.), Kluwer Academic Publishers, in press.

[22] P. Lancaster and K. Salkauskas, *Surfaces generated by moving least squares methods*, Mathematics of Computation **37** (1981), 141–158.

[23] W.K. Liu and Y. Chen, *Reproducing kernel particle methods*, International Journal for Numerical Methods in Fluids **20** (1995), 1081–1106.

[24] W.K. Liu, Y. Chen, C.T. Chang, and T. Belytschko, *Advances in multiple scale kernel particle methods*, Computational Mechanics **18** (1996), 73–111.

[25] W.K. Liu, Y. Chen, R.A. Uras, and C.T. Chang, *Generalized multiple scale reproducing kernel particle methods*, Computer Methods in Applied Mechanics and Engineering **139** (1996), 91–158.

[26] W.K. Liu, S. Hao, T. Belytschko, S. Li, and C.T. Chang, *Multiple scale meshfree methods for damage fracture and localization*, Computational Materials Science **16** (1999), 197–205.

[27] ———, *Multi-scale methods*, International Journal for Numerical Methods in Engineering **47** (2000), 1343–1361.

[28] MSC Software Corporation, *MSC PATRAN: User Manual*, 2001 ed., 2001.

[29] B. Nayroles, G. Touzot, and P. Villon, *Generalizing the finite element method: Diffuse approximation and diffuse elements*, Computational Mechanics **10** (1992), 307–318.

[30] R.W. Ogden, *Non-linear elastic deformations*, John Wiley and Sons, Inc., New York, 1984.

[31] R.A. Regueiro and R.I. Borja, *A finite element model of localized deformation in frictional materials taking a strong discontinuity approach*, Finite Elements in Analysis and Design **33** (1999), 283–315.

[32] J.H. Rose, J. Ferrante, and J.R. Smith, *Universal binding energy curves for metals and bimetallic interfaces*, Physical Review Letters **47** (1981), 675–678.

[33] L.A. Schoof and V.R. Yarberry, *ExodusII: A finite element data model*, Tech. Report SAND92-2137, Sandia National Laboratory, 1992.

[34] J.C. Simo, *A framework for finite strain elastoplasticity based on maximum plastic dissipation and the multiplicative decomposition: part I. continuum formulation*, Computer Methods in Applied Mechanics and Engineering **66** (1988), 199–219.

[35] ______, *A framework for finite strain elastoplasticity based on maximum plastic dissipation and the multiplicative decomposition: part II. computational aspects*, Computer Methods in Applied Mechanics and Engineering **68** (1988), 1–31.

[36] ______, *Algorithms for static and dynamic multiplicative plasticity that preserve the classical return mapping schemes of the infinitesimal theory*, Computer Methods in Applied Mechanics and Engineering **99** (1992), 61–112.

[37] J.C. Simo, F. Armero, and R.L. Taylor, *Improved versions of assumed enhanced strain tri-linear elements for 3D finite deformation problems*, Computer Methods in Applied Mechanics and Engineering **110** (1993), 359–386.

[38] J.C. Simo and T.J.R. Hughes, *Computational inelasticity*, Springer, New York, 1998.

[39] J.C. Simo and R.L. Taylor, *Quasi-incompressible finite elasticity in principal stretches. continuum basis and numerical algorithms*, Computer Methods in Applied Mechanics and Engineering **85** (1991), 273–310.

[40] J.C. Simo, R.L. Taylor, and K.S. Pister, *Variational and projection methods for the volume constraint in finite deformation elastoplasticity*, Computer Methods in Applied Mechanics and Engineering **51** (1995), 177–208.

[41] I. Stakgold, *The Cauchy relations in a molecular theory of elasticity*, Quarterly of Applied Mechanics **8** (1950), 169–186.

[42] F.H. Stillinger and T.A. Weber, *Computer simulation of local order in condensed phases of silicon*, Physical Review B **31** (1985), 5262–5271.

[43] E.B. Tadmor, M. Ortiz, and R. Phillips, *Quasicontinuum analysis of defects in solids*, Philosophical Magazine A **73** (1996), 1529–1563.

[44] M.G.A Tijssens, E. van der Giessen, and L.J. Sluys, *Modeling of crazing using a cohesive surface methodology*, Mechanics of Materials **32** (2000), 19–35.

[45] V. Tvergaard and J.W. Hutchinson, *The relation between crack growth resistance and fracture process parameters in elastic-plastic solids*, Journal of the Mechanics and Physics of Solids **40** (1992), 1377–1397.

[46] P. Underwood, *Dynamic relaxation*, Computational methods for transient analysis (T. Belytschko and T.J.R. Hughes, eds.), Elsevier Science, 1983, pp. 245–265.

[47] A.F. Voter, *The embedded atom method*, Intermetallic Compounds: Principles and Practice (J.H. Westbrook and R.L. Fleischer, eds.), John Wiley and Sons, Ltd., 1994, p. 77.

[48] A.F. Voter and S.P. Chen, *Accurate interatomic potentials for Ni, Al, and Ni_3Al*, Proceedings of the 1987 MRS Fall Symposium, Characterization of Defects in Materials, vol. 82, Materials Research Society, 1987, pp. 175–180.

[49] J.H. Weiner, *Statistical mechanics of elasticity*, John Wiley and Sons, Inc., New York, 1983.

[50] X.-P. Xu and A. Needleman, *Numerical simulations of fast crack growth in brittle solids*, Journal of the Mechanics and Physics of Solids **42** (1994), 1397–1434.

[51] C. Yoon and D.H. Allen, *Damage dependent constitutive behavior and energy release rate for a cohesive zone in a thermoviscoelastic solid*, International Journal of Fracture **96** (1999), 55–74.