SAND2007-5784P

Sandia National Laboratories

Solid Mechanics Training:

Course Material

Picatinny Arsenal
September 24-28, 2007
Picatinny, NJ

Course Contacts:
Arne Gullerud, (505) 844-4326, asgulle@sandia.gov
Martin Heinstein, (505) 844-1257, mwheins@sandia.gov
Sandia National Labs
Department 1542

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy under contract DE-AC04-94AL85000.

Sandia National Laboratories

Outline of Training Material

1. Sierra Mechanics
   a. Resources
   b. Physics modules
   c. Analysis process

2. Presto: explicit transient dynamics module
   a. Capability overview
   b. Command summary

Outline of Training Material

3. Example: Rod Impacting Plate
   a. w/ Hexes
   b. w/ Tets
   c. w/ Element death
   d. w/ Element–to–particle conversion
   e. w/ various Materials

4. Example: Colliding rings
   a. w/ Hexes
   b. w/ contact options

Sierra Mechanics

Sierra Mechanics: Outline of Presentation

1. Resources
   a. Sierra specific resources
   b. Related resources (meshing, data viewers, etc.)
2. Physics modules
3. Analysis process
   a. Pre-processing (create mesh, create input file)
   b. Analysis execution (running Presto)
   c. Post-processing (visualizing/processing results)
4. Input file anatomy

Sierra Mechanics: Resources

Provided to Sandia internal user community:

Sierra user support: (505)-845-1234

for obtaining help, reporting issues/bugs

Sierra mechanics web page: http://sierra.sandia.gov

ESP100 solid mechanics short course:

http://www.sandia.gov – webfileshare

Presto web page: http://presto.sandia.gov

Sierra Mechanics: Resources

Provided to External user communities: (case-by-case basis, what we can provide you here/now)

Presto training material

Presto users manual

Test suites (Performance, Regression)

Example problems

ESP100 solid mechanics short course notes

Cubit “Cheat Sheet”

Paraview “Cheat Sheet”

Blot “Cheat Sheet”

Sierra Mechanics: Related Resources

Cubit : web page, http://cubit.sandia.gov (SNL internal)

SIMBA : http://simba.ca.sandia.gov (SNL internal)

Ensight : http://www.ensight.com ,
http://www.ran.sandia.gov/ensight (SNL Internal)

Patran : http://www.mscsoftware.com/products/patran.cfm

Paraview: http://www.paraview.org

SEACAS : http://jal.sandia.gov/SEACAS/SEACAS.html
(SNL Internal)

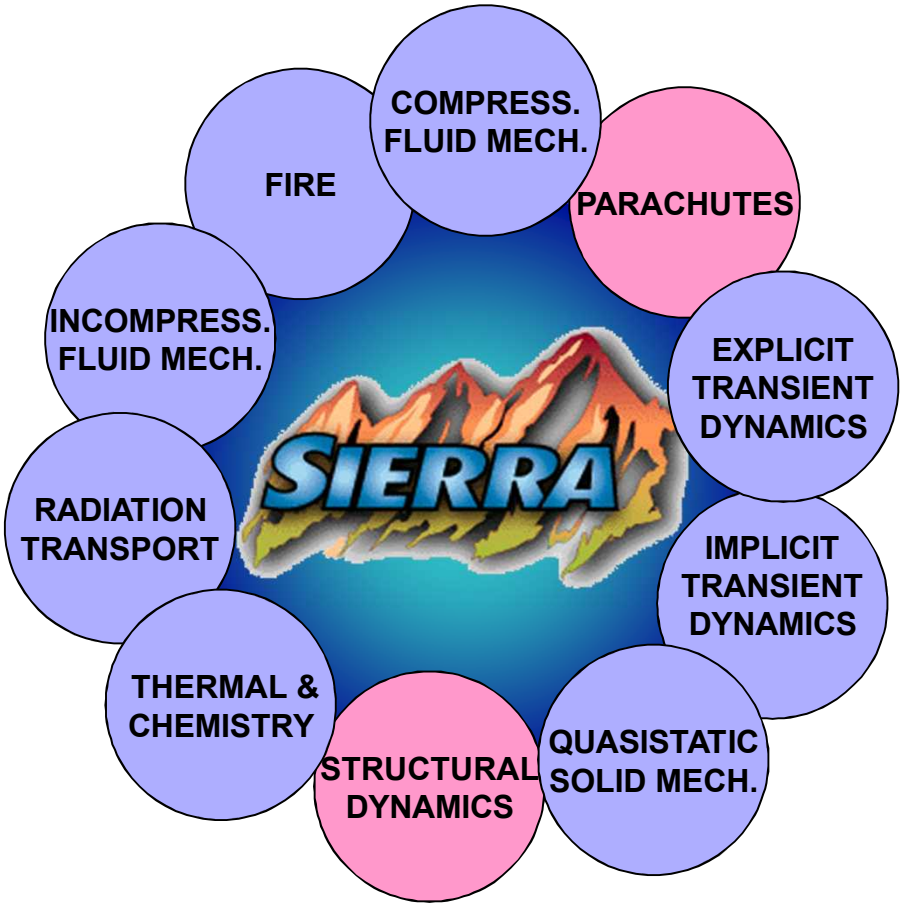
Sierra Mechanics: Physics Modules

Solids Codes:

- Presto: Transient Dynamics
- Adagio: Quasistatics
- Salinas: Structural Dynamics
- Tempo: Solids handoff

Fluids/Thermal Codes:

- Calore: Thermal
- Calagio: Thermal/Structural
- Fuego: Low Mach
- Premo: Compressible fluid dynamics
- Aria: Incompressible/coupled mechanics



Analysis Process: Overview

As in all analysis processes, there are three steps:

1. Pre-processing: Mesh & input file creation
2. Analysis execution: Running the analysis
3. Post-processing: Visualizing the results

These steps are *not* inside one package or under one interface when using Presto

Analysis Process: Pre-Processing

Pre-processing must produce two inputs for analysis execution:

1. mesh file
   …is a binary file(s) in the Genesis/Exodus format derived off of HDF5

2. input file
   …is a text file

Analysis Process: Pre-Processing

Mesh file generation

- ✓ Cubit (licensed)
  Sandia's current meshing tool
- Patran (licensed)
  can convert from other codes: LSDyna, Abaqus
  Sandia's "exodus preferences" are available
- SEACAS (available from Sandia)
  Sandia legacy meshing tools – Fastq, Gen3D, Grepos, SPHGEN, etc.

Analysis Process: Pre-Processing

Input file generation

- ✓ Typical approach is to use an existing example and modify (w/ text editor) as appropriate
- SIMBA is a GUI-based input file generator that understands how to generate a Presto (or other Sierra) input file (it also generates LSDyna, Abaqus, Pronto input files)
- Converter script exists to convert from Pronto input file

Analysis Process: Analysis Execution (running Presto)

Running Presto is typically done through the “Sierra” script:

sierra -V <version**> presto -i <**input file**> -j <**num procs**>**

-V <version>	must be a known version that was part of the install, for this training, e.g. : PRESTO2.7beta4
presto	refers to the Sierra mechanics (physics) module to execute
-i <input file>	refers to the ASCII text file
-j <num procs>	refers to the number of processors the code execution will use

Analysis Process: Analysis Execution (running Presto)

Running Presto is typically done through the “Sierra” script:

sierra -V <version**> presto -i <**input file**> -j <**num procs**>**

Other flags:

-Q <queue>

-T <time for queue>

-X <location of exe>/<name of exe>

-x <project with exe>

-h (man page)

Analysis Process: Analysis Execution (running Presto)

Sierra script does several operations:

1. Takes a single mesh file and splits it (slice & spread) into many (one for each processor specified)
2. Writes a submission script to run the code (sierra.sh)
   - If there is a queuing system, submits submission script to queue
   - If there is not, runs script
3. Upon completion, combines parallel results files into a single file

Analysis Process: Analysis Execution (running Presto)

In the event of queuing systems w/ locally provided features, these operations can be done ‘manually’ by power users:

- The “LOADBAL” tool splits the mesh for parallel (command line driven)
- The “EPU” tool re-combines the distributed results files (command line driven)
- The submission script can be written and executed OR submitted to queue (usually done based on one that already exists)

Analysis Process: Analysis Execution (running Presto)

During execution, four output files are written

1. Log file (text format, *.log)
   logs execution progress, error messages, etc.
2. Results file (Exodus format, typically *.e)
   mesh with node/element variables at user specified time interval
3. History file (Exodus format, typically *.h)
   global quantities, e.g. KE, or node/element values at user specific points in time
4. Restart file (Exodus format, typically *.rst)
   like results file, but intended to restart/continue the simulation if needed

Analysis Process: Analysis Execution (running Presto)

Note: During execution, additional files may be written

1. Queuing system (text format, usually *.e<jobid>, *.o<jobid>)
   contains messages from the queueing system and possibly error output (stderr)
2. Machine output file (*.output)
   Same kind of information as from the queuing system for machine without a queue

Analysis Process: Post-Processing

A variety of tools exist for viewing EXODUS files

- Ensight (licensed)
  Commercial, GUI driven
  available on many platforms, e.g. Linux workstations, windows
- ✓Paraview (no license needed)
  Partially Sandia funded, GUI driven
  available on many platforms, e.g. Linux workstations, windows
- BLOT (part of SEACAS, Sandia legacy tools)
  text input driven
- Patran (licensed)
  Commercial, GUI driven

Analysis Process: Input File Anatomy

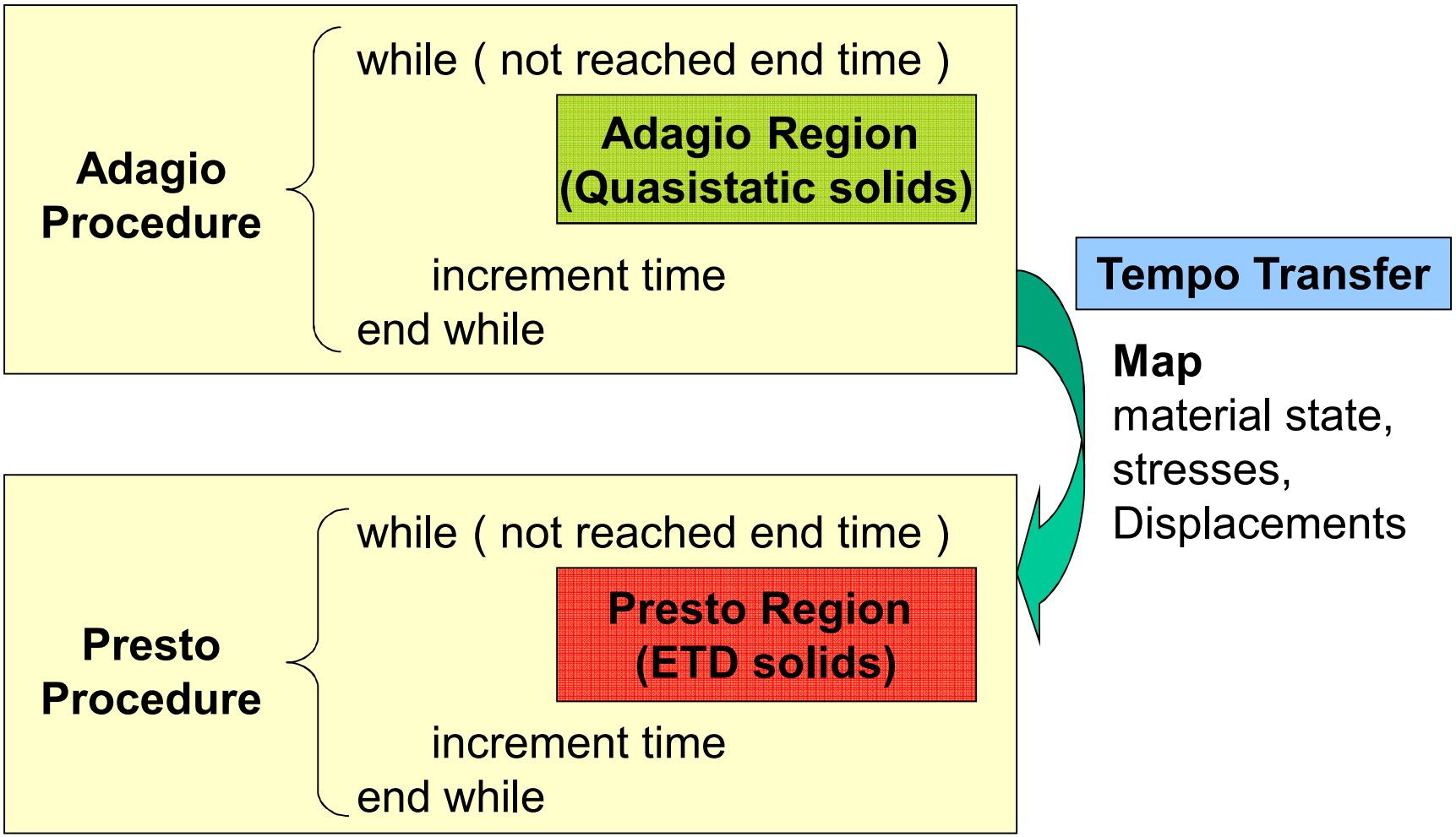
The input file for all Sierra codes follow a structure adopted to facilitate loose multi-physics coupling

Primary parts:

- a **procedure** advances time from a start to end time by driving **regions** and **transfers** data between them (loose coupling)
- a **region** computes physics for a single time increment – can be single-physics or multi-physics (tight coupling)
- a **transfer** maps nodal/element state data between **regions**

Analysis Process: Input File Anatomy

Example: Pre-load analyses (a one-time “hand-off”)

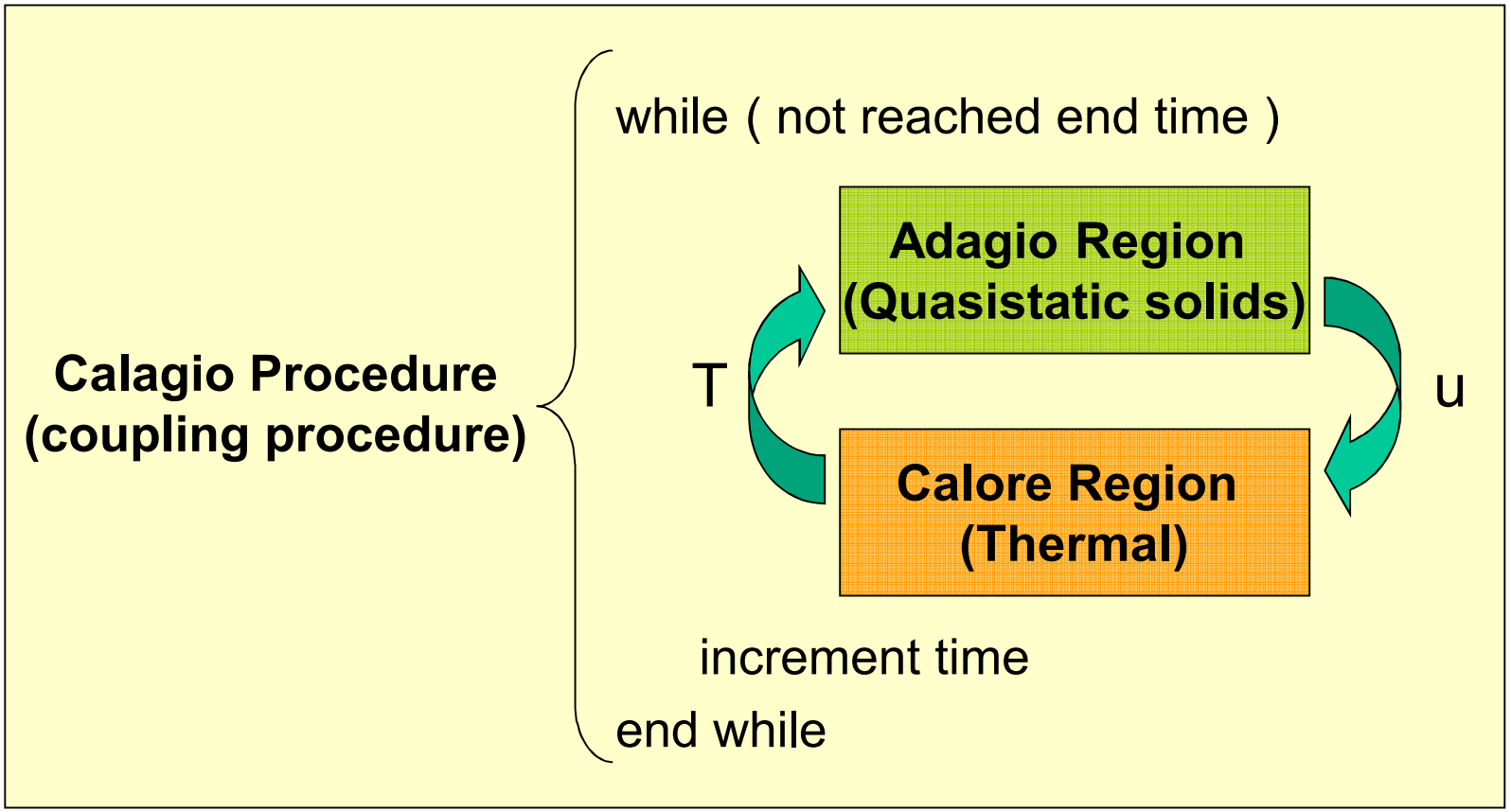


Analysis Process: Input File Anatomy

```
begin sierra <my_analysis_name>
  … functions, materials, mesh file to read …
  begin adagio procedure <my_procedure_name>
    … time control definition …
    begin adagio region <my_region_name>
      … adagio physics definition  …
    end
  end
  begin presto procedure <my_procedure_name>
    … time control definition …
    … transfer description …
    begin presto region <my_region_name>
      … presto physics definition  …
    end
  end
end
```

Analysis Process: Input File Anatomy

Example: Thermal-Structural modeling (a loose coupling)



Analysis Process: Input File Anatomy

```
begin sierra <my_analysis_name>
  … functions, materials, mesh file to read …
  begin calagio procedure <my_procedure_name>
    … time control definition …
    begin adagio region <my_adagio_region_name>
      … adagio physics definition  …
    end
    … transfer description …
    begin calore region <my_calore_region_name>
      … calore physics definition  …
    end
  end
end
```

Analysis Process: Input File Anatomy

Understanding scope

- Information applicable potentially to multiple physics goes in **sierra** scope
- Time control goes in **procedure** scope
- Physics-specific information goes in **region** scope

A **procedure** and a **region** is needed even if you are only using a single physics

We will only focus on a Presto single-physics analysis

Hands-on Session 1: Rod-impacting-plate problem

Introduce the problem
Cubit session
Review already created input file
Paraview session

Sandia National Laboratories

Presto:
Sierra mechanics explicit transient dynamics module

Sandia National Laboratories

Assumptions in this Training

You are familiar with explicit transient dynamic analysis, e.g. LSDyna, Abaqus explicit, EPIC, etc.
(see ESP100 short course notes for refresher)

You are just looking into how to get up and running as quick as possible

You already know or are willing to learn how to do mesh generation and visualization using Cubit, Paraview

Presto: Outline of Presentation

1. Overview
   a. Standard modeling features
   b. Advanced modeling features

2. Input file command syntax
   a. Basics
   b. Sierra scope commands
   c. Presto procedure scope commands
   d. Presto region scope commands

Presto: Overview

Standard modeling features

- Critical time step: Element-based
- Elements: Uniform-Gradient hex8, tet4, Key-Hoff shell4, Beam, Truss, Spring, SPH
- ICs: Displacements, Velocity, Acceleration
- BCs: Force, Moments, Displacements, Velocity, Acceleration
- Contact/impact: skinning, two-pass algorithm, friction models
- Materials: standard

Presto: Overview

Advanced modeling features

- Critical time step: Node-based, Global estimator
- Elements: SD hex8, Nodal-Based tet4 w/ remeshing, GPA, HPM
- BCs: Periodic, User-defined
- Contact/impact: iterative (implicit) enforcement of momentum balance, MP-scalable, user-defined friction models
- Materials: advanced

Presto: Overview

Advanced modeling features (cont'd)

- Element death + element-to-particle conversion
- Remeshing
- CTH coupling
- Embedded fiber-in-matrix (rebar)
- Multi-length scale

Presto: Input File Structure

Basics:

- A modeling feature is typically specified in a **Begin/End** block
- **Begin/End** blocks define a “scope”
- Lines within **Begin/End** block define required/optional input for the modeling feature

Example: specifying a boundary condition

```
begin fixed displacement
    node set = nodelist_1
    components = X Z
end
```

Presto: Input File Structure

Basics (cont'd):

- Syntax is free-format and verbose
- Most command lines fit "keyword = value" style
- Comment characters are "#" and "$"
- Single lines can span multiple lines if followed by a "\#"
- Syntax is case insensitive (except for filenames)
- Commands within their respective scope are order-independent
- Certain names are reserved for referring to mesh parts in the genesis file: "block_<id>", "surface_<id>", "nodelist_<id>

Presto: Input File Structure

Basics (cont'd):

- The end statement at the conclusion of a command block can either be just "end" or can mirror the begin block

```
begin fixed displacement
      … data …
end
```

```
begin fixed displacement
      … data …
end fixed displacement
```

- Presto as a single physics requires three major nested **Begin/End** blocks:
  1. Sierra block
  2. Presto procedure block
  3. Presto region block

Presto: Input File Structure

Understanding scope

- Sierra block defines a **sierra** scope, where the FE model (materials, mesh) is defined
- Procedure block defines a **procedure** scope, where the simulation (time control) is defined
- Region block defines a **region** scope, where all physics-related information is defined

A **procedure** and a **region** is needed even if you are only using a single physics

Presto: Input File Structure

```
begin sierra <name>
   … functions, directions …
   … materials …
   … mesh file to read …
   begin presto procedure <procedure_name>
      … time control definition  …
      begin presto region <region_name>
         … boundary conditions …
         … contact …
         … output …
      end
   end
end
```

Sierra scope

procedure scope

region scope

Presto: Input File
Sierra Scope Commands

Sierra scope commands:

- Functions
- Directions, axes, points
- Restart control
- Materials
- Sections
- Finite Element Model

} Mesh description

Presto: Input File
Sierra Scope Commands

Functions

can be specified via types ‘constant’, ‘piecewise linear’, or ‘analytic’

syntax:

```
begin definition for function <name>
  type = [constant | piecewise linear |
          analytic]
  begin values
      …
  end
end
```

Presto: Input File
Sierra Scope Commands

Functions

Example: ‘piecewise linear’

```
begin definition for function ramp
  type = piecewise linear
  begin values
    0.0       0.0
    1.0e-3    1.0
  end
end
```

Presto: Input File
Sierra Scope Commands

Functions

Example: ‘analytic’

```
begin definition for function cosine_ramp
   type = analytic
   evaluate expression = \#
      "(1-cos(x * 1.0e+04 * 1 * pi));"
end
```

Note: “\#” means line continuation

Presto: Input File
Sierra Scope Commands

Directions, axes, points

are defined with line commands:

```
define point <name> with coordinates \#
  <real> <real> <real>
define direction <name> with vector \#
  <real> <real> <real>
define axis <name> with point \#
  <point1 name> point <point2 name>
define axis <name> with point \#
  <point1 name> direction <dir name>
```

Presto: Input File
Sierra Scope Commands

Materials

are characterized by (possibly many) stress-strain models within individual material **begin/end** blocks

- Multiple stress-strain model descriptions are allowed, e.g. the material ‘tungsten’ could be described by the stress-strain models ‘elastic_plastic’ and ‘johnson_cook’ (as well as others)
- Multiple stress-strain model feature is useful for coupling, exploring effect of constitutive assumptions on the solution, or utilization of a pre-characterized database of stress-strain models for materials

Presto: Input File
Sierra Scope Commands

Materials

syntax

```
begin property specification for \#
  material <name>
  density = <real>
  begin parameters for model <model_type>
       parameter = <value>
       …
  end
  begin parameters for model <model_type>
       parameter = <value>
       …
  end
end
```

Presto: Input File
Sierra Scope Commands

Sections

define parameters specific for a type of element, e.g.

for Solid elements,

- Strain incrementation (MI, SO)
- Formulation (MQ, SD)

for Shell elements,

- Thru-thickness integration
- Thickness approach (constant for all elements in a block, element attribute from mesh file)

for Beam elements,

- Cross section type
- Dimensions

All sections lie in the sierra scope

Presto: Input File
Sierra Scope Commands

Sections

Examples:

```
begin beam section rod_block
  section = rod
  width = 0.01
  height = 0.01
  t axis = 0.0 0.0 1.0
end beam section rod_block
```

```
begin solid section so_solid
 strain incrementation = strongly_objective
end solid section so_solid
```

Presto: Input File
Sierra Scope Commands

Finite element model

specifies the connection between the mesh and element block parameters/properties, e.g.

- A material and one of it's stress-strain models
- A section
- Element block numerical parameters, e.g. hourglass stiffness values

Presto: Input File
Sierra Scope Commands

Finite element model

syntax:

```
begin finite element model <name>
  database name = <mesh_file>
  begin parameters for block <block_id>
       material = <material_name>
       solid mechanics use model <model_name>
       section = <section_name>
       … element block numerical parameters …
  end
  …
end
```

Presto: Input File
Sierra Scope Commands

… element block numerical parameters …

are any number of element block related quantities, e.g.

- Linear bulk viscosity
- Quadratic bulk viscosity
- Hourglass stiffness
- Hourglass viscosity
- Effective moduli model

Hands-on Session 2: Rod-impacting-plate problem

Cubit: change element type from hex to tet

Presto execution

Paraview session

Presto: Input File
Procedure Definition

Procedure

contains description of simulation time control

- drives time incrementation
- calls the Presto **region** for a time step

Note: Sierra scope commands are not allowed inside the **procedure** scope, e.g.

- no function, direction, section, material definitions, … inside **procedure begin/end** block

Presto: Input File
Procedure Scope Commands

Presto procedure

syntax:

```
begin presto procedure <procedure name>
   … time control definition …
   begin presto region <region name>
      … presto physics definition …
   end
end
```

Presto: Input File
Procedure Scope Commands

Time control

defines the end time of the analysis, and enables the subdivision of time into identified chunks. These chunks can be used to turn on and off bounday conditions, element blocks, etc. For each block, we can define parameters to be used by Presto region, e.g.

- initial time step
- time step scale factor
- time step increase factor
- step interval for analysis “heartbeat” to log file

Presto: Input File
Procedure Scope Commands

Time control

syntax:

```
begin time control
  begin time stepping block <name>
    start time = <time>
    begin parameters for presto \#
      region <region_name>
      ... presto region parameters ...
    end
  end
  termination time = <time>
end
```

Presto: Input File
Region Scope Commands

Region definition

holds the physics-specific Presto specific information, including:

- Which mesh description to use
- Initial conditions
- Initial values (of element/nodal variables)
- Boundary conditions
- Contact
- I/O definitions
- Element death

Presto: Input File
Region Scope Commands

Which mesh description to use

is specified through the “use finite element model command”

```
begin sierra demo
  …
  begin finite element model mesh_1
    …
  end
  begin presto procedure my_proc
    …
    begin presto region my_region
      use finite element model mesh_1
      …
    end
  end presto procedure my_proc
end sierra demo
```

Presto: Input File
Region Scope Commands

Initial conditions

can be specified in multiple ways, e.g.

- using different coordinate systems, (cartesian, cylindrical, etc.) defined in the sierra scope
- with user subroutines

Presto: Input File
Region Scope Commands

Initial velocity

can be specified on node sets, side sets, element blocks

syntax:

```
begin initial velocity
  [node set | surface | block] = <name>
  remove [node set | surface] = <name>
  component = [x | y | z]
  direction = <dir name>
  magnitude = <value>
end
```

Hands-on Session 3: Rod-impacting-plate problem

Changing the initial velocity

Paraview session

Sandia National Laboratories

Presto: Input File
Region Scope Commands

Prescribed kinematic BCs

syntax:

```
begin prescribed \#
    [displacement|velocity|acceleration]
  [node set|surface|block] = <name>
  component = [x|y|z]
  direction = <dir name>
  function = <function name>
  scale factor = <value>
end
```

Presto: Input File
Region Scope Commands

Fixed displacement

syntax:

```
begin fixed displacement
  [node set | surface | block] = <name>
  component = [x | y | z]
end
```

Presto: Input File
Region Scope Commands

Force BCs

can be specified with many options, e.g.

- Nodal force vector
- Gravity
- Cavity expansion
- Silent BC
- Spot-Weld
- Line-Weld
- Viscous Damping

Presto: Input File
Region Scope Commands

Force BC via nodal force vector

syntax:

```
begin prescribed [force | moment]
  [node set | surface ] = <name>
  direction = <dir name>
  function = <function name>
  scale factor = <value>
end
```

Presto: Input File
Region Scope Commands

Force BC via pressure

syntax:

```
begin pressure
  surface = <surface id>
  function = <function name>
end
```

Can also read in pressure per face from variable defined on the mesh file, variable defined on a coupled region (e.g. CTH), or by user subroutine. Also, can define as tractions.

Presto: Input File
Region Scope Commands

Contact

command block layout:

```
begin contact definition <name>
   … contact surface definitions …
   … remove initial overlap …
   … shell lofting …
   … friction models …
   … search options …
   … enforcement options …
   … interaction defaults …
   … interaction definitions …
end
```

Presto: Input File
Region Scope Commands

Contact surface definitions

are specified as element blocks to be skinned, defined surfaces, or defined nodesets

- Example: skin all element blocks

```
skin all blocks = on
```

- Example: using specific surfaces

```
contact surface <name> contains <blocks, surfaces>
```

- Example: node set

```
contact nodeset <name> contains <nodeset>
```

- Can define hybrid surface (e.g. block minus a surface), use analytic surfaces, or do node sets

Presto: Input File
Region Scope Commands

Contact remove initial overlap

move contact nodes to avoid initial penetration

- Meshes frequently have a little bit of overlap in the parts that can cause problems with contact (loss of contact, excessive forces)
- Can cause element inversion if there is too much overlap or if the tolerances are too large
- Special options are available for shells to iteratively remove overlap

Presto: Input File
Region Scope Commands

Contact remove initial overlap

syntax:

```
begin remove initial overlap
  overlap normal tolerance = <value>
  overlap tangential tolerance = <value>
end
```

Presto: Input File
Region Scope Commands

Contact shell lofting

is done to define topologically unique sides of a shell

- Can specify how lofting is done in the shell lofting section
- Can turn shell lofting off
- Has a number of options for dealing with coincident shells and hexes or shells and shells

Presto: Input File
Region Scope Commands

Contact shell lofting

syntax

```
begin shell lofting
  lofting algorithm = [on | off]
  coincident shell treatment = \#
    [disallow | ignore | simple ]
  coincident shell hex treatment = \#
    [disallow | ignore | tapered ]
end
```

Presto: Input File
Region Scope Commands

Contact friction models

are used in contact interactions

- 11 friction models are available, including:
  - phenomenological failure (weld) models
  - cohesive laws
  - Adhesion
  - advanced friction models
  - user subroutine models
- A number of models can fail and change to a different friction model after failure (e.g., weld models, cohesive models)

Presto: Input File
Region Scope Commands

Contact friction models

examples:

- Frictionless

```
begin frictionless model <name>
end
```

- Tied

```
begin tied model <name>
end
```

- Constant Friction

```
begin constant friction model <name>
  friction coefficient = <value>
end
```

Presto: Input File
Region Scope Commands

Contact search options

- Global search

  By default, a global search is done each time step; best for most complex contact situations

- Tracking

  Local search every time step w/ a periodic global search every n-steps; best for persistent contact

- Search tolerances

  Must specify contact tolerances – physical distance to search for overlaps

  We are finalizing a capability to automatically determines tolerances

Presto: Input File
Region Scope Commands

Contact search options

syntax:

```
begin search options
  global search increment = <num steps>
  normal tolerance = <value>
  tangential tolerance = <value>
end
```

Presto: Input File
Region Scope Commands

Contact enforcement options

- Properly solving enforcement requires an implicit solution on contact constraints
- We do an iterative solution; user picks accuracy versus CPU cost balance
- We by default do a momentum balance solution, but are testing a penalty enforcement (that may be faster)

Presto: Input File
Region Scope Commands

Contact enforcement Options

Syntax:

```
begin enforcement options
    momentum balance iterations = <int>
end
```

Presto: Input File
Region Scope Commands

Contact interaction defaults

- Default default is that no surfaces interact with any other surfaces
- Can specify that all surfaces interact by default
- Can request contact algorithm to automatically determine kinematic partition factor
  (by default, it assumes same stiffness between contacting surfaces)
- Can specify default friction model

Presto: Input File
Region Scope Commands

Contact interaction defaults

- Syntax:

```
begin interaction defaults
  general contact = [on | off]
  self contact = [on | off]
  automatic kinematic partition = [on | off]
  friction model = <name>
end
```

Presto: Input File
Region Scope Commands

Contact interaction definitions

specify how a pair of surfaces should interact

- Any of the defaults can be overridden (e.g. tolerances, friction model)
- Interaction can also be turned off
- Can define master/slave enforcement or symmetric contact

Presto: Input File
Region Scope Commands

Contact interaction definitions

Syntax:

```
begin interaction <name>
  surfaces = <surface 1> <surface 2>
  kinematic partition = <value>
  master = <surface>
  slave = <surface>
  normal tolerance = <value>
  tangential tolerance = <value>
  friction model = <name>
  interaction behavior = [sliding |
                          no_interaction]
end
```

Choose one

Hands-on Session 4: Rod-impacting-plate problem

Changing contact interaction options
Changing contact enforcement options
Paraview session

Sandia National Laboratories

Presto: Input File
Region Scope Commands

Output

There are three kinds of output, each of which is separately controlled in the region scope

- Results – full mesh file with requested variables on each element/node
- History – output global variables or values at specific points
- Restart – suitable for a checkpoint restart in case of machine crash, etc.

Many options are available to control the frequency of output (output on signal, output schedulers, etc…)

Presto: Input File
Region Scope Commands

Output

Examples of variables names (full list in Presto manual)

nodal	element
force_external	stress
force_internal	log_strain
force_contact	von_mises
velocity	…
displacement	

Can have user-defined variables derived from existing variables (summations, etc.)

Presto: Input File
Region Scope Commands

Results output

Syntax:

```
begin results output <name>
  database name = <name>
  at time <value> increment = <value>
  at step <value> increment = <value>
  node variables = <var name> as <out name>
  element variables = <var name> as <outname>
end
```

Presto: Input File
Region Scope Commands

History output

Syntax:

```
begin history output <name>
  database name = <name>
  at time <value> increment = <value>
  at step <value> increment = <value>
  variable = [node|element] <var name>
    at [node|element] <int> as <out name>
end
```

Presto: Input File
Region Scope Commands

Restart output

Syntax:

```
begin restart data <name>
  database name = <name>
  at time <value> increment = <value>
  at step <value> increment = <value>
end
```

NOTE: Restarting at a specific time or last restart time instead of at start time is specified in the **sierra** scope

```
restart time = <time>
restart = automatic
```

Hands-on Session 5: Rod-impacting-plate problem

Requesting nodal, element output
Changing the frequency of output, restart
Paraview session

Presto: Input File
Region Scope Commands

Element death capability

involves killing (deleting) elements based on

- Material damage
- Element inversion
- Deformation measure
- Time step
- Other variables

“or” condition is used when multiple criterion are specified

Presto: Input File
Region Scope Commands

Element death capability

Once death criterion is met, element deletion occurs over a specified number of steps

Death criterion is checked at a user specified step/time interval

Newly created surfaces as a result of element death can be included in contact

Killed elements can be converted into particles

Presto: Input File
Region Scope Commands

Element death

Syntax:

```
begin element death <name>
  block = <list of blocks>
  criterion is element value of <var name>
              [<|<=|=|>=|>] <value>
  material criterion = <material model name>
  death on inversion
  convert elements to particles with \#
     section = <section name>
  death steps = <value>
  check [time|step] interval = <value>
end
```

Hands-on Session 6: Rod-impacting-plate problem

Specifying element death

Paraview session

Sandia National Laboratories

Hands-on Session 7: Rod-impacting-plate problem

Converting dead elements to particles

Ensight session

Hands-on Session 8: Rod-impacting-plate problem

Attaching particles to intact elements

Ensight session

Sandia National Laboratories