SAND2014-15912PE

Exceptional service in the national interest

Sandia National Laboratories

Overview of Eagle3 (SIERRA Based) Sandia National Laboratories

GIC*L

July , 2014

Presented by:
Vicki Porter, (505) 844-4326, vlporte@sandia.gov
Sandia National Labs
Department 1542

GOODYEAR

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

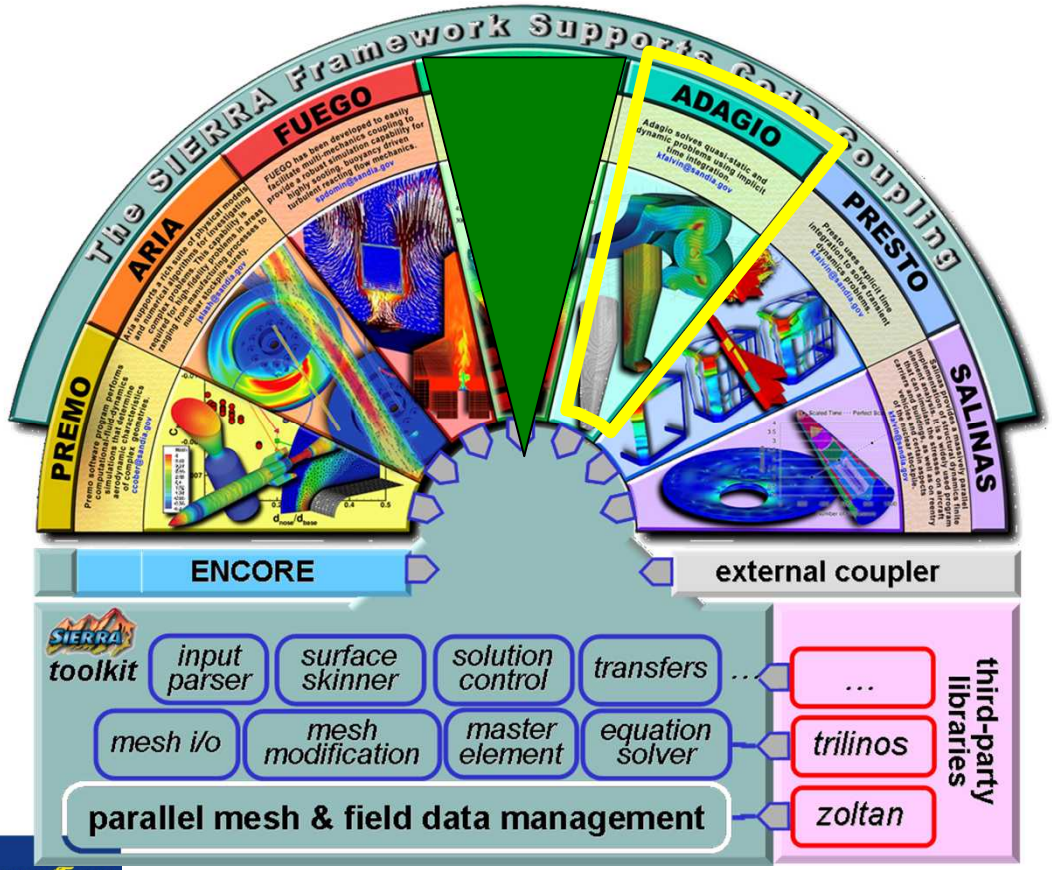
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

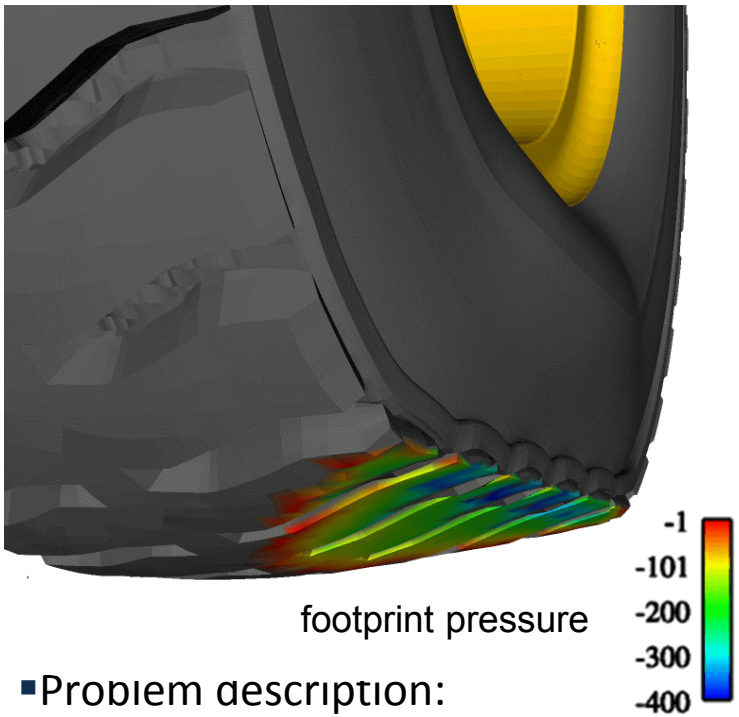
Eagle3 Physics Modules (Regions)

- Quasistatics and Implicit Dynamics – Adagio
- Explicit Dynamics – Presto
- Structural Dynamics – Salinas
- Thermal -- Aria

SIERRA Mechanics modules

SIERRA_Adagio
Non-linear Solid mechanics



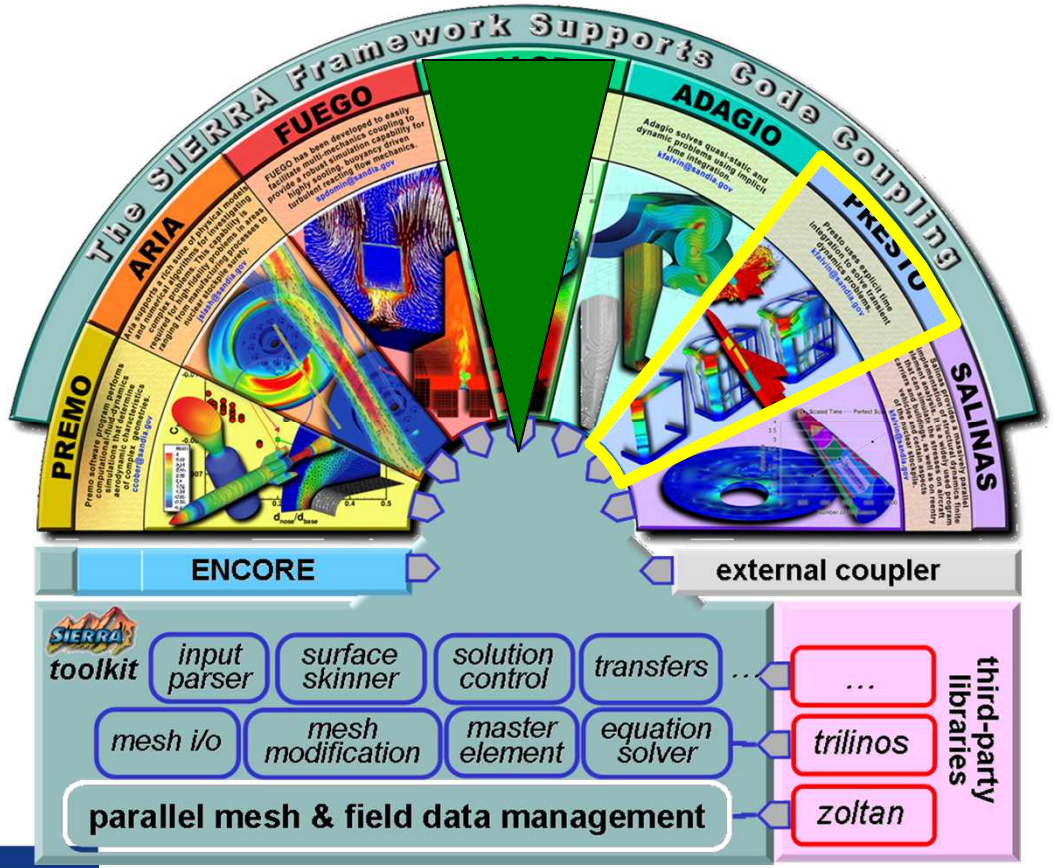


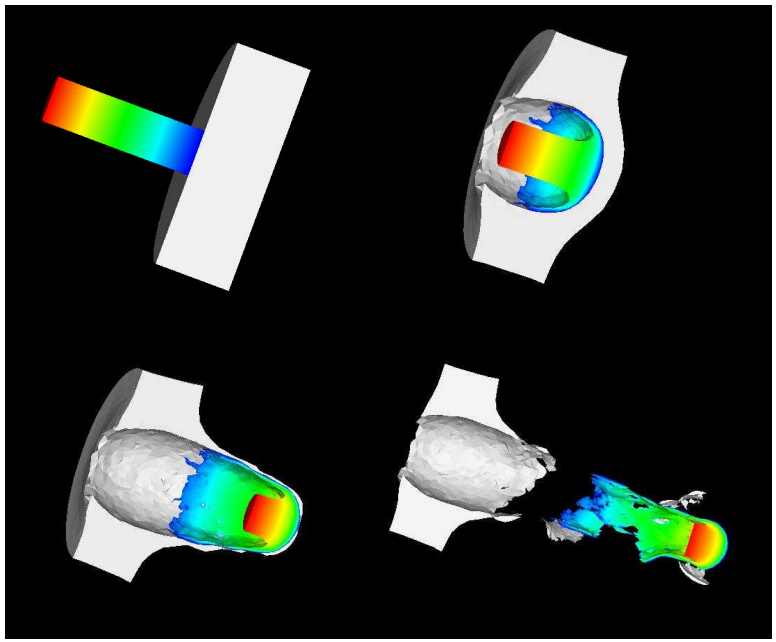
footprint pressure

- Problem description:
- Tire performance modeling, camber angle = 18°
- Capabilities:
- Quasistatics, implicit dynamics, Parallel non-linear implicit solvers, contact, >50 material models, failure & tearing

SIERRA Mechanics modules

SIERRA_Presto
Non-linear Solid transient dynamics

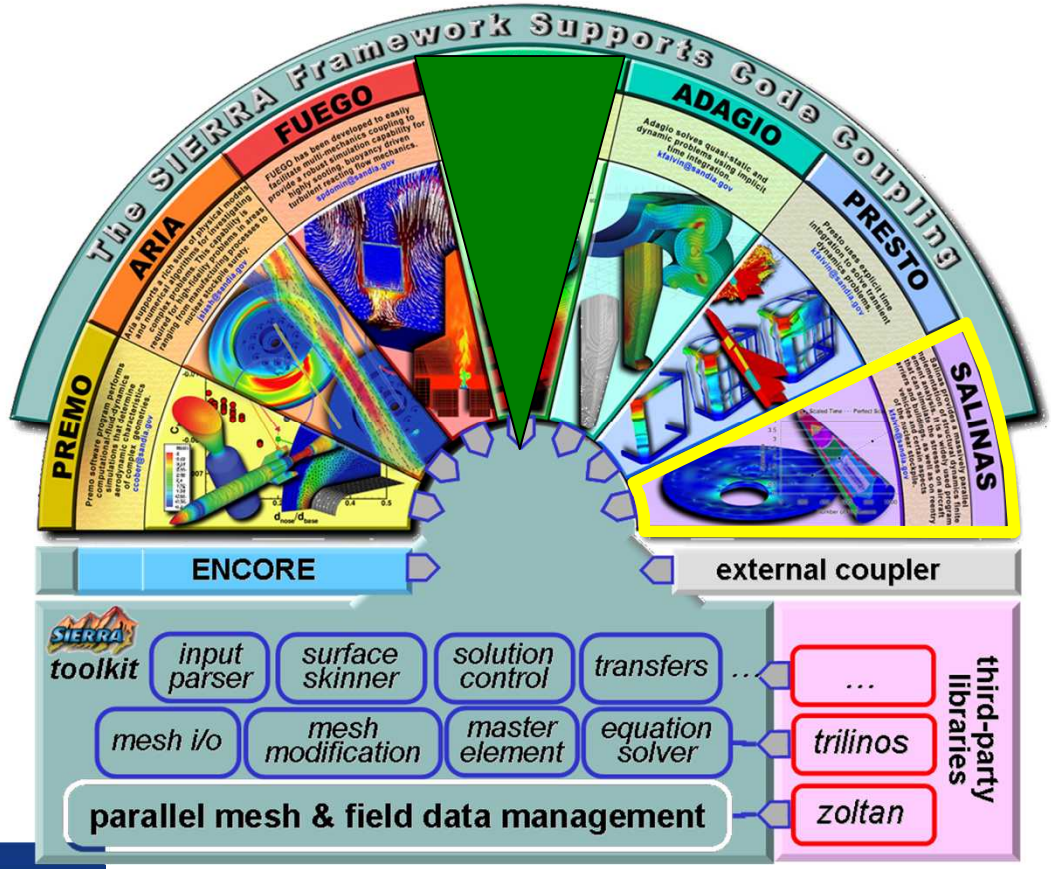


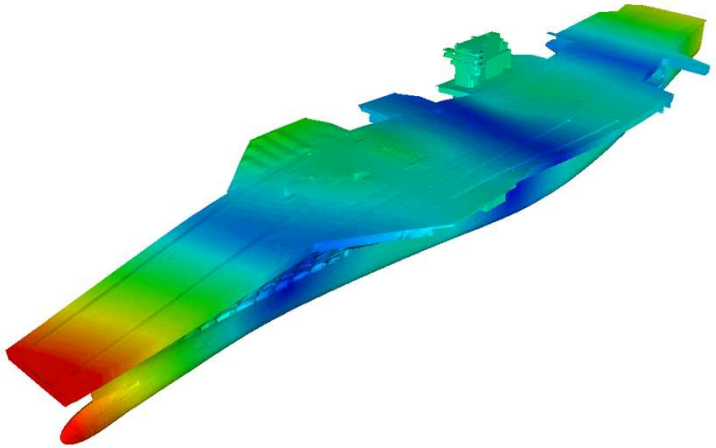


- Problem description:
- Tungsten rod impacting steel plate @2500 m/s
- Capabilities:
- Explicit time integrator, Hexes, shells, beams, Nodal-based tet w/ remeshing, particle methods, cohesive surface elements, spot welds, contact, material failure

SIERRA Mechanics modules

SIERRA_Salinas
Linear structural dynamics

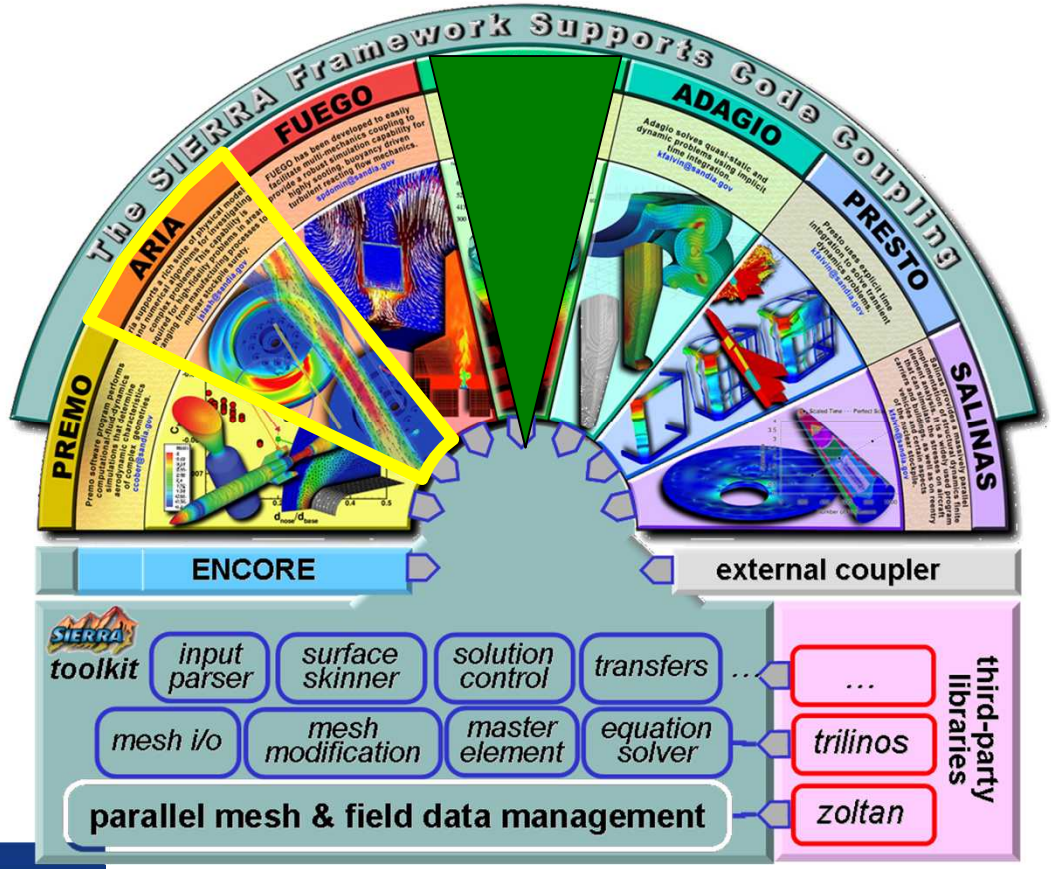




- Problem description:
- Modal solution of complex structures (1000's of material regions, offset shells and beams)
- 2.0M DOFs, solved on 64 processors
- Acoustics

SIERRA Mechanics modules

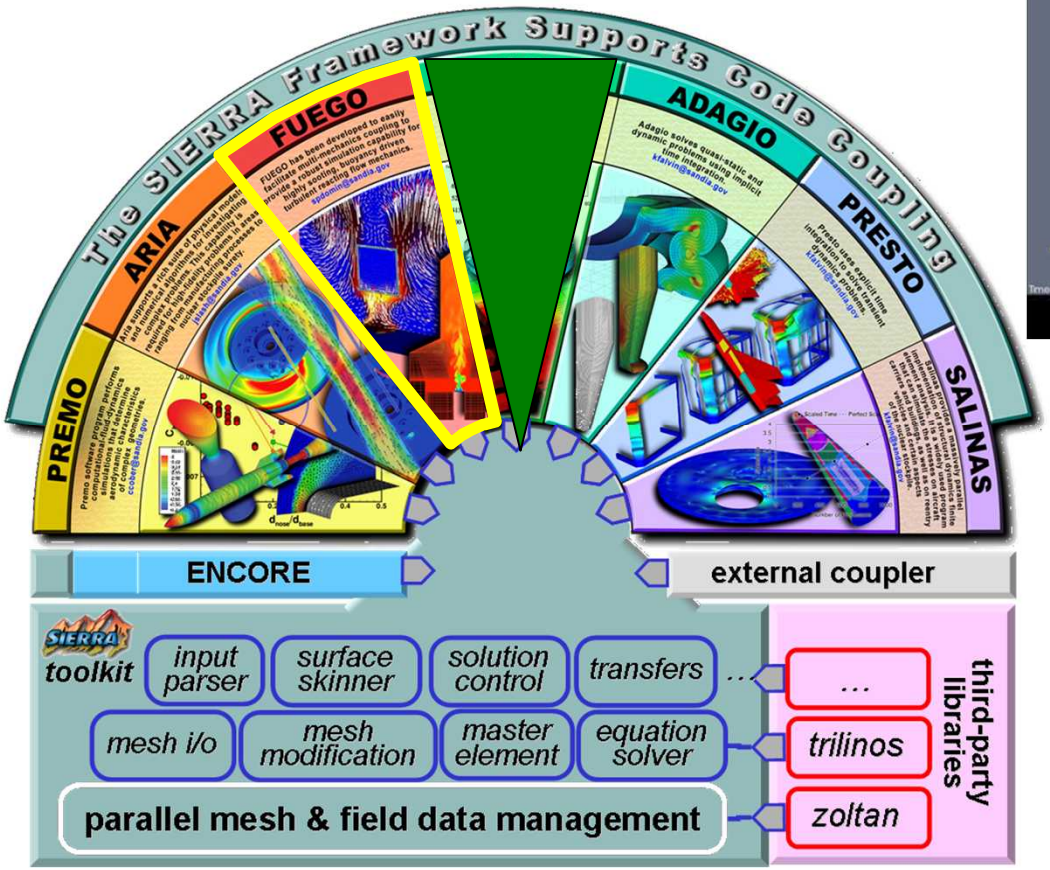
SIERRA_Aria
Thermal Conduction
Non-Newtonian flow

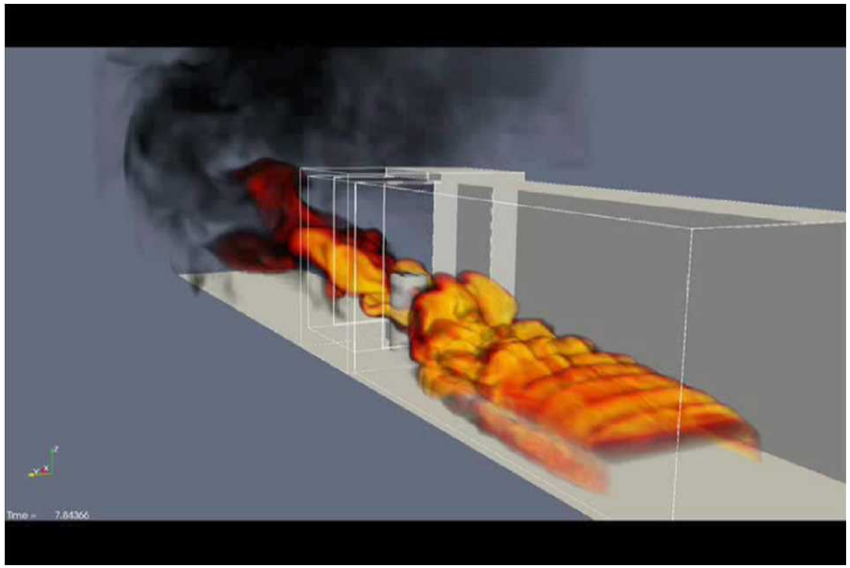


- Capabilities:
- Thermal Conduction/Radiation
- Chemically Reacting flows, Level sets for free-surface tracking
- Complex material response
- Basic Physics: Navier-Stokes (variable ρ), Energy, Species, Electrostatics

SIERRA Mechanics modules

SIERRA_Fuego
Low Mach number, finite-volume fluid dynamics





- Problem description:
- Hydrocarbon pool fire, 400M DOFs, 5000 procs. on Red Storm
- Capabilities:
- Turbulent reacting flow with coupling to participating media radiation and heat conduction, RANS and LES-based turbulence models

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
  - a **transfer** maps nodal/element state data between **regions**

Input File Structure

```
begin sierra <my_name>
    … functions, directions …
    … materials …
    … mesh file to read …

    begin adagio procedure <my_procedure_name>
        … time control definition  …

        begin adagio region <my_region_name>
            … boundary conditions …
            … contact …
            … output …

        end
    end
end
```

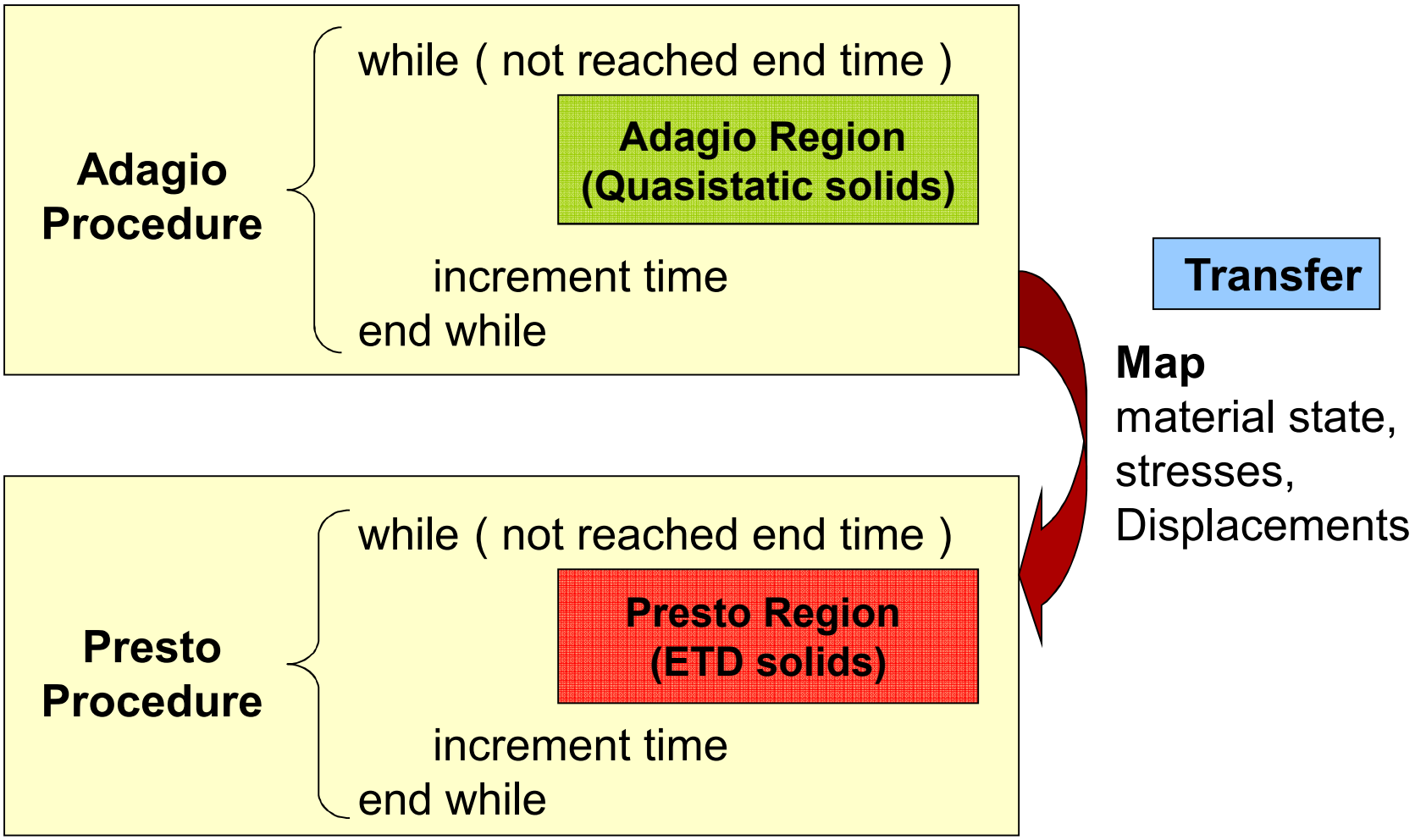
Sierra scope

procedure scope

region scope

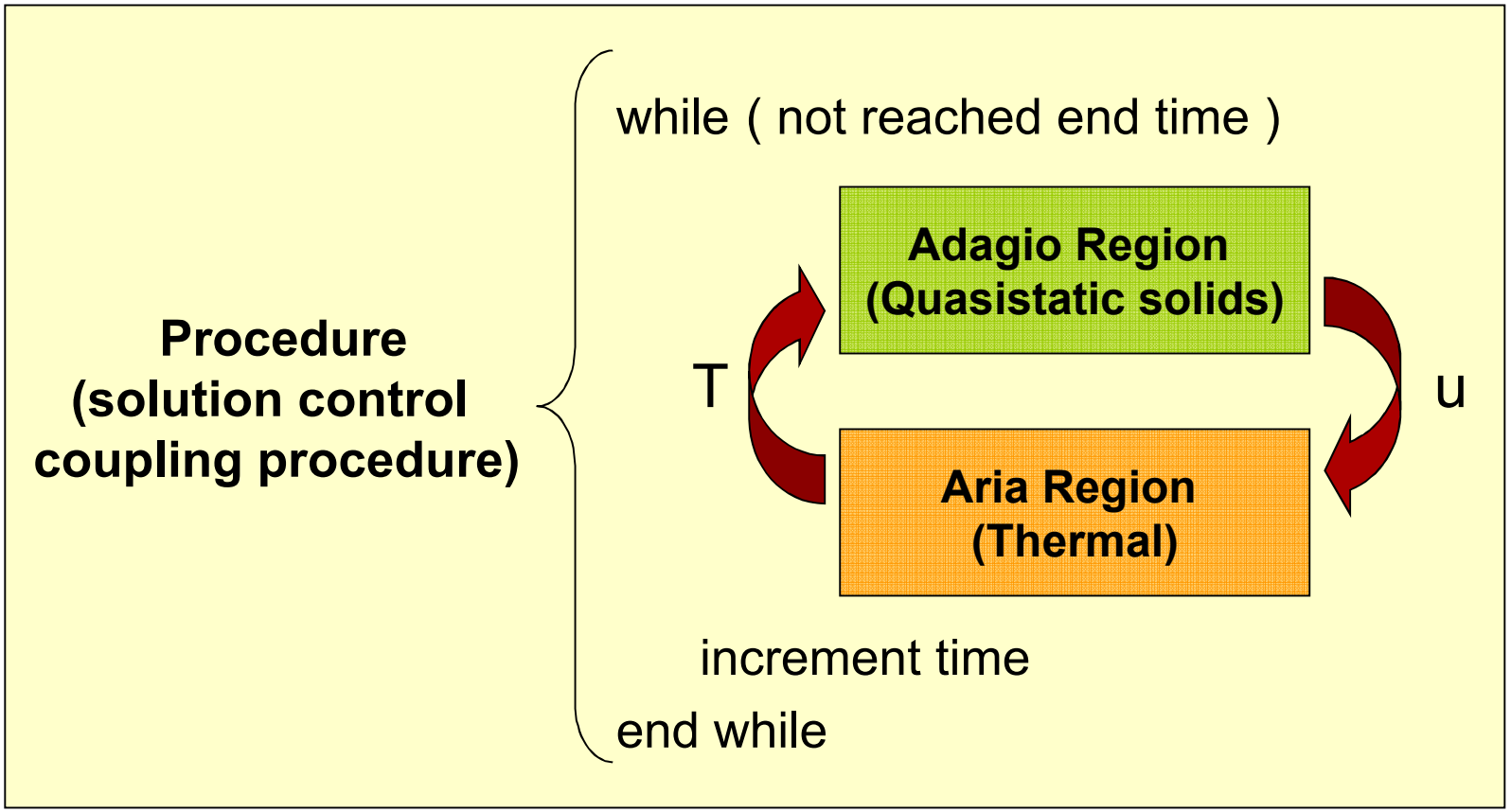
Analysis Process: Input File Anatomy

- Example: Inflate analysis with a one-time "hand-off" to explicit rolling.



Analysis Process: Input File Anatomy

- Example: Thermal-Structural modeling (a loose coupling)



Goodyear Applications

Sandia National Laboratories

- Inflate – Serial quasistatic (adagio procedure and region) pressurization of wedge.
- Revolve – Serial wedge-to-3D transfer to 3D ribbed mesh (adagio procedure with adagio region).
- Deflect – Parallel quasistatic deflection of ribbed tire (adagio procedure and adagio region).
- Re-equilibriation – Parallel restart then transfer to treaded tire (adagio procedure with adagio region).
- Lagrangian Quasistatic Rolling – Parallel restart then transfer to a new adagio region (adagio procedure and two adagio regions).
- SST – Parallel restart then transfer to a new adagio region (adagio procedure with two adagio regions).
- Explicit Rolling – Parallel restart then transfer to presto region (adagio procedure with adagio region and presto region).
- Thermal SST – Parallel restart from deflect for adagio followed by running a generic procedure with transfers between aria and adagio regions for loose coupling (generic procedure with aria and adagio regions).
- Acoustics/EigenMode Analysis – Inflate-Deflect-Re-equilibriate followed by explicit rolling and transfer to structural dynamics (Several adagio procedures and adagio/presto regions followed by transfer to salinas region).

GOODYEAR

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 information is defined related to the physics that is being solved
- A **procedure** and a **region** are needed even if you are only using a single physics

Input File
Sierra Scope Commands

Sandia National Laboratories

- Sierra scope commands:
  - Functions
  - Directions, axes, points
  - Restart control
  - Materials
  - Sections
  - Finite Element Model

Mesh description

GOODYEAR

Input File
Procedure Definition

Sandia National Laboratories

- **Procedure**
- contains description of simulation time control
  - drives time incrementation
  - calls the Adagio **region** for a time (load) 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

GOODYEAR

Input File
Region Scope Commands

- The **region** holds the physics-specific information:
  - Which mesh description to use
  - Output definitions
  - Initial conditions
  - Initial values (of element/nodal variables)
  - Boundary conditions
  - Contact
  - Element death

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 Y Z
end
```

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
```

  - A solid mechanics single physics analysis requires three major nested **Begin/End** blocks:
    1. Sierra block
    2. Adagio Procedure block
    3. Adagio Region block

Input File
Sierra Scope Commands

Sandia National Laboratories

- Sierra scope commands:
  - Functions
  - Directions, axes, points
  - Restart control
  - Materials
  - Sections
  - Finite Element Model

Mesh description

GOODYEAR

Input File
Sierra Scope Commands

- Functions
- can be specified via types ‘constant’, ‘piecewise linear’,
  or ‘analytic’
- syntax:

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

Input File
Sierra Scope Commands

- Functions
- Example: ‘piecewise linear’

```
begin definition for function load
  type = piecewise linear
  begin values
     0.0       0.0
     1.0       0.0
    10.0       1.0
  end
End
```

Note: text in dark blue means a name you can choose (and can refer to later)

Input File
Sierra Scope Commands

Sandia National Laboratories

- 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

GOODYEAR

Input File
Sierra Scope Commands

- Materials syntax
- **begin property specification for \# material <my_name>**

```
    density = <real>

    begin parameters for model <model_type>
        parameter = <value>
        …
    end

    begin parameters for model <model_type>
        parameter = <value>
        …
    end

end
```

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

Input File
Sierra Scope Commands

- BEGIN SOLID SECTION <string>solid_section_name
  FORMULATION = <string>MEAN_QUADRATURE|
  SELECTIVE_DEVIATORIC (MEAN QUADRATURE)
  DEVIATORIC PARAMETER = <real>deviatoric_param
  STRAIN INCREMENTATION = <string>MIDPOINT_INCREMENT|
  STRONGLY_OBJECTIVE|NODE_BASED
  (MIDPOINT_INCREMENT)
  NODE BASED ALPHA FACTOR = <real>bulk_stress_weight (0.01)
  NODE BASED BETA FACTOR = <real>shear stress_weight (0.35)
  HOURGLASS FORMULATION = <string>TOTAL|INCREMENTAL
  (TOTAL)
  RIGID BODY = <string>rigid_body_name
  USE LAME
- END [SOLID SECTION <string>solid_section_name]

Input File
Sierra Scope Commands

- BEGIN SHELL SECTION <string>shell_section_name
  - THICKNESS = <real>shell_thickness
  - THICKNESS MESH VARIABLE = <string>THICKNESS|<string>var_name
  - THICKNESS TIME STEP = <real>time_value
  - THICKNESS SCALE FACTOR = <real>thick_scale_factor (1.0)
  - INTEGRATION RULE = TRAPEZOID|GAUSS|LOBATTO|SIMPSONS|USER (TRAPEZOID)
  - NUMBER OF INTEGRATION POINTS = <integer>num_int_points (5)
  - BEGIN USER INTEGRATION RULE
    - <real>location_1 <real>weight_1
    - <real>location_2 <real>weight_2
    - …
    - <real>location_n <real>weight_n
  - END [USER INTEGRATION RULE]
  - LOFTING FACTOR = <real>lofting_factor (0.5)
  - ORIENTATION = <string>orientation_name
  - RIGID BODY = <string>rigid_body_name
- END [SHELL SECTION <string>shell_section_name]

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

Input File
Sierra Scope Commands

- Finite element model syntax:

```
begin finite element model <my_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
```

Input File
Sierra Scope Commands

Element block numerical parameters include:

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

Input File
Procedure Definition

- The **procedure** contains a description of simulation time control
  - drives time incrementation
  - calls the Adagio **region** for a time (load) 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

Input File
Procedure Scope Commands

- Procedure syntax:

```
begin adagio procedure <procedure_name>
   … time control definition …
   begin adagio region <region_name>
      … adagio physics definition …
   end
end
```

Input File
Procedure Scope Commands

- Time control
- Defines the start & end time of the analysis, and enables the subdivision of time into identified chunks. These chunks can be used to turn on and off boundary conditions, element blocks, etc. For each block, we can define parameters to be used by Adagio region, e.g.
  - start time (of the time block)
  - time increment
  - initial time step
  - time step scale factor
  - time step increase factor
  - step interval for analysis “heartbeat” to log file

Input File
Procedure Scope Commands

- Time control syntax:

```
begin time control
  begin time stepping block <my_name>
    start time = <time>
    begin parameters for adagio \#
        region <region_name>
        … adagio region parameters …
    end
  end
  termination time = <time>
end
```

Input File
Region Scope Commands

The **Region** definition holds the physics-specific Adagio specific information including:

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

Input File
Region Scope Commands

- Which mesh description to use
- Specified through the “use finite element model “ line

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

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…)

Input File
Region Scope Commands

- Output
- Examples of variable names (full list in Adagio 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.)

Input File
Region Scope Commands

- Results output syntax:

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

Input File
Region Scope Commands

- History output syntax:

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

Input File
Region Scope Commands

- Restart output syntax:

```
begin restart data <my_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
```

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
```

Input File
Region Scope Commands

Sandia National Laboratories

- Fixed displacement syntax:

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

GOODYEAR

Input File
Region Scope Commands

Sandia National Laboratories

- 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
```

GOODYEAR

Input File
Region Scope Commands

- Force BC via pressure syntax:

```
begin pressure
  surface = <surface id>
  function = <function name>
  scale factor = <value>
end
```

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

Input File
Region Scope Commands

- Contact command block layout:

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

only functional w/ ETD

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 specify default friction model

Input File
Region Scope Commands

- Contact surface definitions are specified as element blocks to be skinned, defined surfaces, and/or defined nodesets
  - Example: skin all element blocks

```
skin all blocks = on
```

  - Example: using specific blocks and 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

Input File
Region Scope Commands

- Contact friction models
  - Example: Frictionless

```
begin frictionless model <my_name>
end
```

  - Example: Tied

```
begin tied model <my_name>
end
```

  - Example: Constant Friction

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

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 (AL, Explicit)

Input File
Region Scope Commands

- Contact interaction definition syntax:

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

Input File
Region Scope Commands

- Mulit-Level Solver: Iterative solver that Adagio uses to handle nonlinearities such as contact, material incompressibilty, and SST nested outside of the core CG solver.
- Command block layout:

```
begin solver <multilevel_solver_name>
   … core CG solver …
   … control contact solver …
   … control stiffness solver …
   … control element death solver …
end
```

Input File
Region Scope Commands

- Core CG solver definition
- Specified as the "core non-linear equation solver" in a "Newton-like" manner. Various pre-conditioners (PC) are available:
  - Example: Nodal PC (using the Nodal PC will result in a non-linear equation solver that is NLPCG)
  - Example: FETI full tangent PC (using the FETI PC will result in a non-linear equation solver that looks most like Newton's method
  - Preconditioners are either formed semi-analytically or via nodal probing (to handle material constitutive models that do not have analytic tangents)

Input File
Region Scope Commands

- ‘Control contact’ solver definition
- Specified to treat contact in an iterative manner nested outside the core CG solver:
  - ‘Level 1’ contact iterations converge w/ specified residual tolerances

```
begin control contact
   target relative residual      = 1.0e-4
   acceptable relative residual = 1.0e-3
   maximum iterations = 10
end
```

  - Active set approach: constraint set is updated before/after the core solver converges @ ‘Level 1’

Input File
Region Scope Commands

- Diagnosing convergence issues, e.g with contact solver parameters for the active set (‘capturing’ or ‘releasing’ nodes in contact) or the Stick-slip set (‘sticking’, ‘slipping’, or ‘unslipping’ nodes in contact)

```
begin control contact
   target relative residual      = <value>
   acceptable relative residual = <value>
   maximum iterations = <value>
   iteration plot = 1
end
```

← Every iteration

Goodyear Uses of Contact

- Typically two kinds of contact:

1. Analytic Rigid Surface for rims and road (can include SST).
2. Augmented Lagrange (QS) and Dash/lclib (ETD) for groove sipe closure.

If ONLY using ARS or lclib:

```
Begin adagio region adagio
 ….
   jas mode
….
End adagio region adagio
```

If using Augmented Lagrange (QS) or Dash (ETD) with or without ARS (no lclib):

```
Begin adagio region adagio
 ….
   jas mode solver
   jas mode output
   jas mode reactions
….
End adagio region adagio
```

Multiple Contact (ARS with Augmented Lagrange)

Sandia National Laboratories

Contact Definition for ARS:

```
begin contact definition NativeARS

    enforcement = ARS

    tied search configuration = current

    contact surface surf_301 contains surface_301
    contact surface surf_302 contains surface_302
    contact surface surf_1000 contains  surface_1000

    begin analytic general surface skin_901
      analytic surface     = 901
      reference rigid body = block_901
    end analytic general surface skin_901

    begin interaction int_1
        name = RIM
        master = skin_901
        slave = surf_301
        normal tolerance = 1.0
        tangential tolerance = 1e-08
        capture tolerance = 0.01
        slip predictor = 0.0
        pushback factor = 0.1
        tension release = 0.0
        friction model = frictionRim
    end interaction int_1

    begin constant friction model frictionRim
        friction coefficient = 0.8
    end constant friction model frictionRim

end contact definition NativeARS
```

GOODYEAR

Multiple Contact (ARS with Augmented Lagrange)

Sandia National Laboratories

- Contact Definition for ARS:

```
begin contact definition AL

   contact formulation type = DASH
   enforcement = AL
   skin all blocks = off
   contact surface groove1 contains surface_801
   contact surface groove2 contains surface_802

   begin interaction groove
#        surfaces = groove1 groove2
      master = groove1
      slave = groove2
      friction model = frictionGroove
   end interaction groove

   begin constant friction model frictionGroove
      friction coefficient = 0.2
   end constant friction model frictionGroove

end contact definition AL
```

GOODYEAR

Multilevel Solver Syntax

```
begin solver

  begin loadstep predictor
    type = scale_factor
    scale factor = 0.0 0.0
    scale factor = 0.0 0.0 during p3
  end loadstep predictor

  begin control contact
        level = 1
        target relative residual = 0.02
        acceptable relative residual = 0.2
        target relative residual = 0.01 during  p3
        acceptable relative residual = 0.1 during  p3
        target relative contact residual = 0.002 during p1 p2
        acceptable relative contact residual = 0.02 during p1 p2
        target relative contact residual = 0.001 during  p3
        acceptable relative contact residual = 0.01 during  p3
        maximum iterations = 1000
        minimum iterations = 10
        lagrange initialize = none
        lagrange adaptive penalty = off
      end control contact
```

Multilevel Solver Syntax

```
begin control stiffness
      level = 1
      target relative residual = 0.02
      acceptable relative residual = 0.2
      target relative strain increment = 2.0
      acceptable relative strain increment = 20.0
      target relative residual = 0.01 during  p3
      acceptable relative residual = 0.1 during  p3
      target relative strain increment = 1.0 during p1 p2
      target relative strain increment = 0.5 during  p3
      acceptable relative strain increment = 2.0 during p1 p2  p3
      maximum iterations = 1000
      minimum iterations = 10
 end control stiffness
```

Multilevel Solver Syntax

```
    begin cg
       target relative residual = 0.01
       acceptable relative residual = 0.1
       target relative residual = 0.005 during  p3
       acceptable relative residual = 0.05 during  p3
       iteration print = 5 during p1 p2  p3
       line search tangent during  p3
       maximum iterations = 1000
       minimum iterations = 10
       reset limits 70 30 10.0 0.5
       iteration print = 1
       line search actual
       preconditioner = block
    end cg

end solver
```

Solution Control for Region Coupling (Thermal SST)

```
#-----Time Step Control  in Procedure ----
begin solution control description
      use system main
      begin system main

      # --- p0 --- #  (user defined time period)
      begin transient p0_01
        advance region_1
          transfer adagio-Aria-qdot
          transfer adagio-Aria-membrane-ends
          transfer adagio-Aria-membrane-ax
          transfer adagio-Aria-membrane-ay
          transfer adagio-Aria-deform
        advance Aria_1
      end transient p0_01
```

```
# --- p1 --- #
      begin transient p1_01
          transfer Aria-adagio-temperature
        advance region_1
          transfer adagio-Aria-qdot
          transfer adagio-Aria-membrane-ends
          transfer adagio-Aria-membrane-ax
          transfer adagio-Aria-membrane-ay
          transfer adagio-Aria-deform
        advance Aria_1
      end transient p1_01

      simulation termination time = 5.0

end system main
```

Transfer Example

```
begin transfer adagio-Aria-qdot
    copy  volume elements from region_1 to Aria_1
    send block block_1  to block_1
    send block block_2  to block_2
    send block block_4  to block_4
    send block block_5  to block_5
    from elements to elements
   send field qdot state none to user_var_qdot state none
  end transfer adagio-Aria-qdot
```

User Defined Output

- Generally consists of three parts:
  1. Functions in Sierra scope.
  2. User defined output blocks in region scope – actually creates the variable.
  3. Results block in region scope that includes the user defined variables as output – tells the code to compute and output the variable at certain times.

- Parts 2 and 3 are always required. Step 2 can used functions defined in one or defined expressions directly.

User Defined Variables

Step 1: Function Definitions in Sierra scope.

```
begin function sine
   type is analytic
   evaluate expression is "amplitude=2;    \#
        frequency=2*pi; \#
        phase=0;        \#
        amplitude * sin(frequency*x + phase)"
end

begin definition for function globalKE
   type = analytic
   expression variable: v = global velocityMag
   expression variable: m = global totalMass
   evaluate expression = "0.5*m*v*v”
end
```

User Defined Variables

Sandia National Laboratories

More Function Definitions Examples:

```
begin definition for function nodalVmagVersion2
   type = analytic
   expression variable: v = nodal_vector velocity
   evaluate expression = "sqrt( v_x*v_x + v_y*v_y + v_z*v_z )"
 end

 begin definition for function elementKE
   type = analytic
   expression variable: vol = element volume
   expression variable: v = global velocityMag
   evaluate expression = " 0.5 * ( 100.0 * vol ) * v * v "
 end
```

GOODYEAR

User Defined Variables

Sandia National Laboratories

Step 2: User Output Block to Define Variable

```
begin user output
    include all blocks                                  (Can include specific blocks)
    compute element sxx as function rateEval
    compute element von as function vonEval
    compute global analyticKE as function analytic_KE
    compute global totalMass as sum of nodal mass
    compute global velocityMax_x as max of nodal velocity(x)
    compute nodal vMag as function nodalVmagVersion2 $ nodalVmag
    compute nodal nKE as function nodalKE
    compute global nodalKE as sum of nodal nKE
    compute global globalExpressionKE from expression " 0.5 * totalMass * vMag * vMag"
    compute global globalFunctionKE as function globalKE
    compute element eKE as function elementKE
    compute global elementKE as sum of element eKE
    compute at every step
  end
```

GOODYEAR

User Defined Variables

Sandia National Laboratories

Step 3: Results Output Block (User defined output can be placed in same results block and output to the same exodus file as intrinsic variables.)

```
begin results output fred
     database name = outputExpression.e
     database type = exodusII
     at time 0.0, interval = 0.005
     nodal variables = displacement
     nodal variables = velocity
     nodal variables = nKE
     element variables = eKE
     global variables = velocityMax_x
     element variables = von
     element variables = sxx
 end
```

GOODYEAR