PennState
College of Engineering

NucE 497: Reactor Fuel Performance

Lecture 16: 2D and 3D Numerical solutions of Thermomechanics

February 15, 2017
Michael R Tonks
Mechanical and Nuclear Engineering

Material is taken from Dr. Motta's book, chapter 6

Today we will finish talking about the coupling of temperature and stress and start numerical solutions

- Module 1: Fuel basics
- Module 2: Heat transport
- Module 3: Mechanical behavior
  - Introduction to solid mechanics
  - Analytical solutions of the mechanics equations
  - Thermomechanics, thermal expansion
  - Solving equations in 1D numerically
  - **Solving in multiple dimensions with FEM**
  - Summary of fuel performance codes
- Module 4: Materials issues in the fuel
- Module 5: Materials issues in the cladding
- Module 6: Accidents, used fuel, and fuel cycle

Here is some review from last time

- What is unique about the stresses and strains in axisymmetric coordinates?
  a) They are not axisymmetric
  b) The rr component of the strain is $u_{r,r}/r$
  c) There are a nonzero stresses and straines in the rr, zz, and θθ directions
  d) They are equal

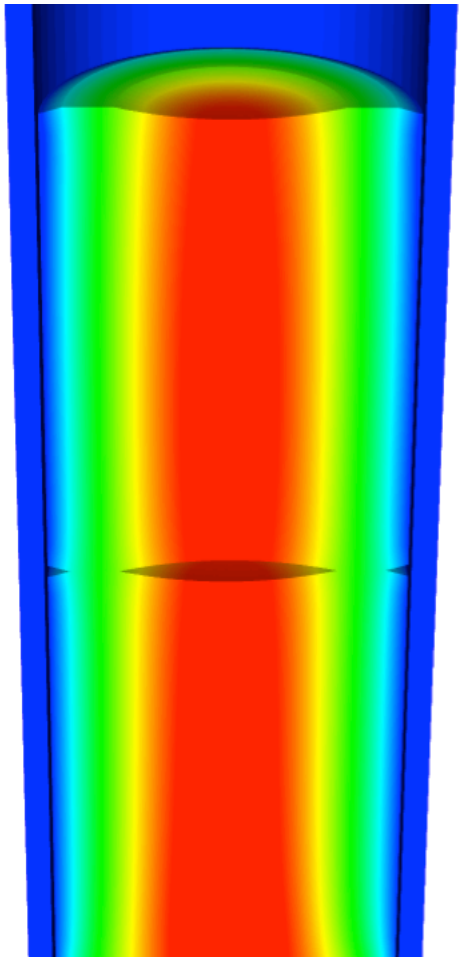
a) How can we consider thermal expansion in the calculation of the pellet T?
  a) Only with numerical methods
  b) With numerical methods or a single analytical calculation
  c) With numerical methods and solving coupled analytical equations for T and for the stress
  d) With a numerical method and an iterative analytical calculation

The temperature and the displacement vector are solved for with the full thermomechanical problem

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + Q \qquad \boldsymbol{\sigma} = \boldsymbol{\mathcal{C}}(\boldsymbol{\epsilon} - \alpha(T - T_{fab})\mathbf{I})$$

$$0 = \nabla \cdot \boldsymbol{\sigma} \qquad \boldsymbol{\epsilon} = \tfrac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)$$

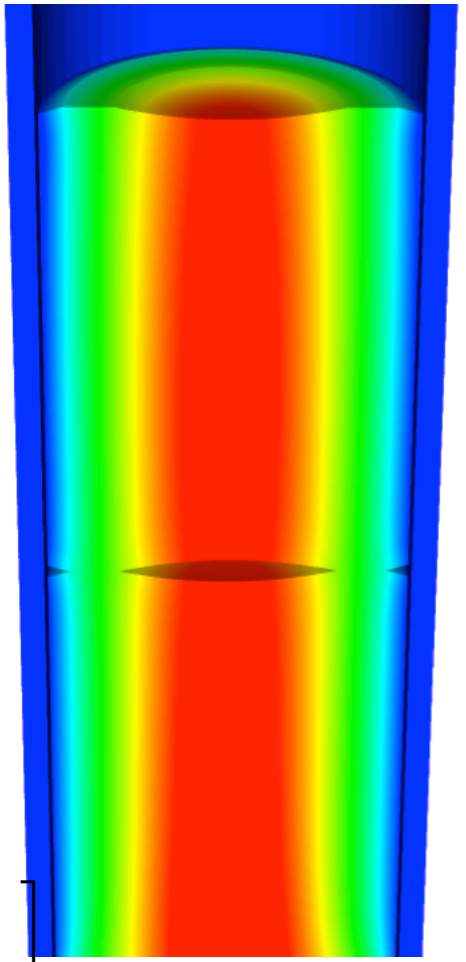
- T impacts the value of **u** through thermal expansion
- **u** impacts the value of T through changes in the thickness of the gap
- The value for T evolves with time
- The value for **u** also evolves with time, even though there is not time in its PDE



The thermomechanical problem becomes 2D when we assume axisymmetry

$$\rho c_p \frac{\partial T}{\partial t} = \nabla \cdot (k \nabla T) + Q \qquad \boldsymbol{\sigma} = \boldsymbol{\mathcal{C}}(\boldsymbol{\epsilon} - \alpha(T - T_{fab})\mathbf{I})$$

$$0 = \nabla \cdot \boldsymbol{\sigma} \qquad \boldsymbol{\epsilon} = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)$$



- Assumption 1: Problem is axisymmetric

$$\rho c_p \frac{\partial T}{\partial t} = \frac{1}{r}\frac{\partial}{\partial r}\left(r k(T)\frac{\partial T}{\partial r}\right) + \frac{\partial}{\partial z}\left(k(T)\frac{\partial T}{\partial z}\right) + Q(r,z)$$

$$\frac{\partial \sigma_{rr}}{\partial r} + \frac{\sigma_{rr} - \sigma_{\theta\theta}}{r} + \frac{\partial \sigma_{rz}}{\partial z} = 0$$

$$\frac{1}{r}\frac{\partial (r\sigma_{rz})}{\partial r} + \frac{\partial \sigma_{zz}}{\partial z} = 0$$

$$\boldsymbol{\sigma} = \boldsymbol{\mathcal{C}}(\boldsymbol{\epsilon} - \alpha(T - T_{fab})\mathbf{I})$$

$$\boldsymbol{\epsilon} = \begin{bmatrix} u_{r,r} & (u_{r,z} + u_{z,r})/2 & 0 \\ (u_{r,z} + u_{z,r})/2 & u_{z,z} & 0 \\ 0 & 0 & u_r/r \end{bmatrix}$$

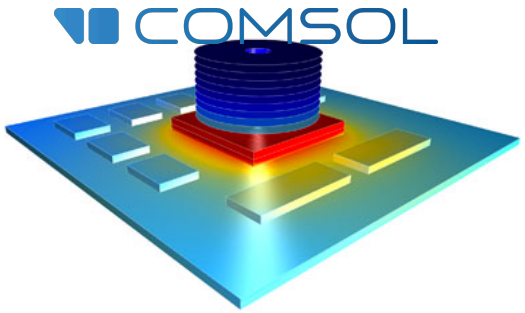
$$\begin{bmatrix} \sigma_{rr} \\ \sigma_{zz} \\ \sigma_{\theta\theta} \\ \sigma_{rz} \end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 \\ \nu & 1-\nu & \nu & 0 \\ \nu & \nu & 1-\nu & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{bmatrix} u_{r,r} \\ u_{z,z} \\ u_r/r \\ (u_{r,z} + u_{z,r})/2 \end{bmatrix}$$

There are various available tools for solving coupled thermomechanical problems with FEM

- Commercial tools
  - ABAQUS
  - ANSYS
  - COMSOL
- Open source
  - MOOSE

MOOSE

COMSOL



ABAQUS

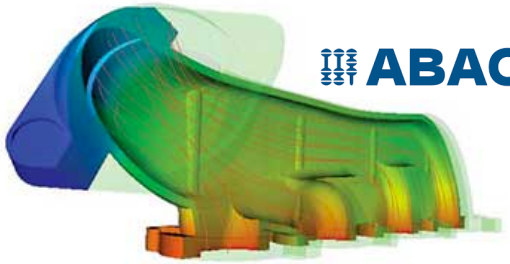
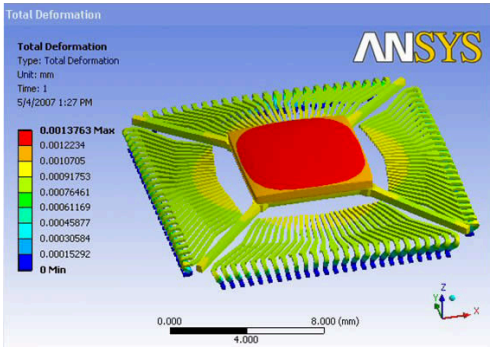


Figure 4. Temperature contour plot of exhaust manifold.

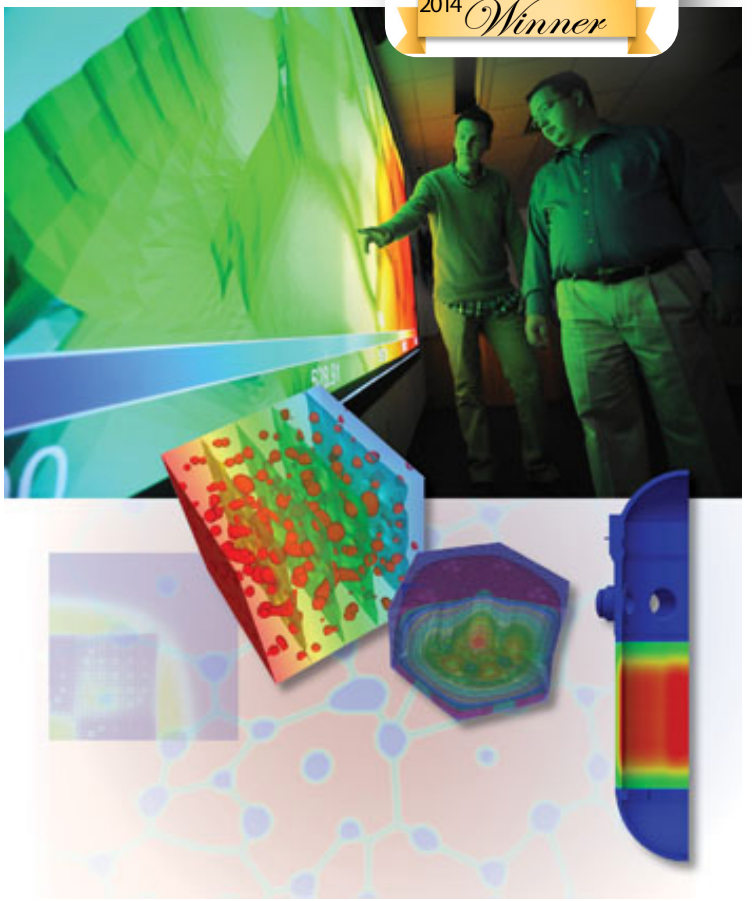


MOOSE

Multiphysics Object Oriented Simulation Environment

- MOOSE is a FEM, multiphysics framework that **simplifies the development** of advanced numerical applications
- It provides a high-level interface to **sophisticated nonlinear solvers** and **massively parallel computational capability**
- It is **open source** and freely available at **mooseframework.org**
- It is maintained and developed by a team of six full time staff

R&D 100 2014 Winner

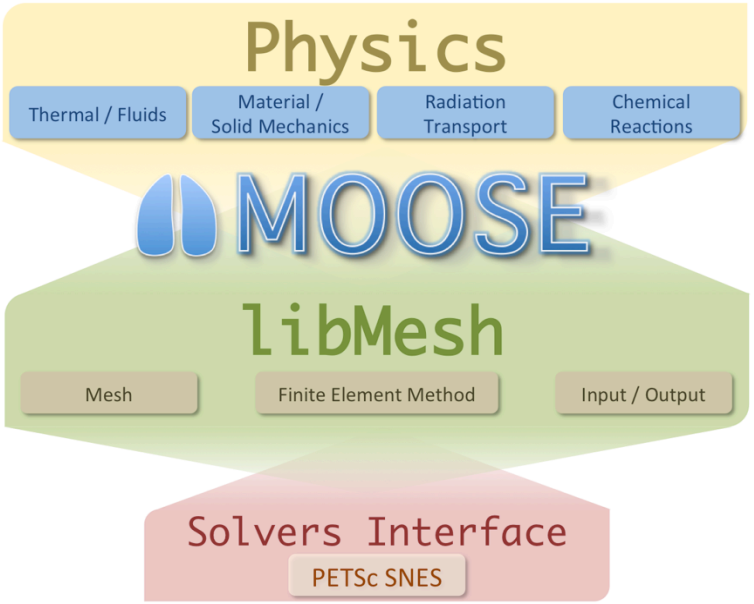


MOOSE is in use across the world to solve:

- Phase field
- Solid mechanics
- Heat conduction
- Neutronics
- Comp. fluid dynamics
- Geomechanics
- Reactive transport
- Corrosion
- Crystal plasticity
- Fracture

MOOSE Capabilities and Structure

- All MOOSE code is dimension agnostic (1D, 2D, or 3D)
- Can be run on a desktop or supercomputer, with no MPI coding
- Full coupled, fully implicit using PETSc SNES
- Mesh and time step adaptivity
- Unstructured mesh
  - All element shapes
  - Various element orders and types
  - Handles mixed element types
  - Reads and write various formats



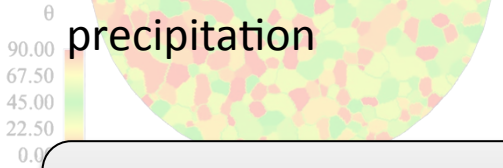
In addition to the numerical framework, MOOSE comes with a number of physics modules

MOOSE

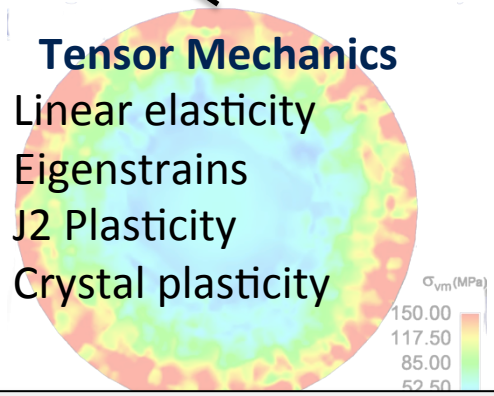
Framework — Tools for solving generic PDEs

Modules

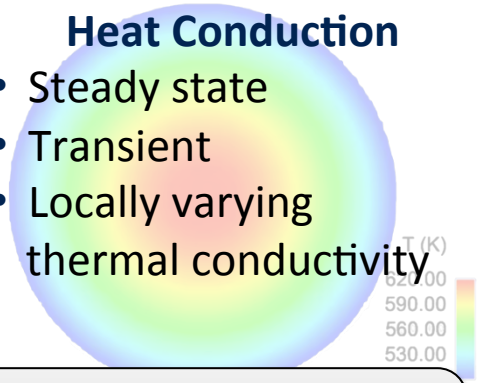
Phase Field
- Grain growth
- Fracture
- Phase change and precipitation



Tensor Mechanics
- Linear elasticity
- Eigenstrains
- J2 Plasticity
- Crystal plasticity

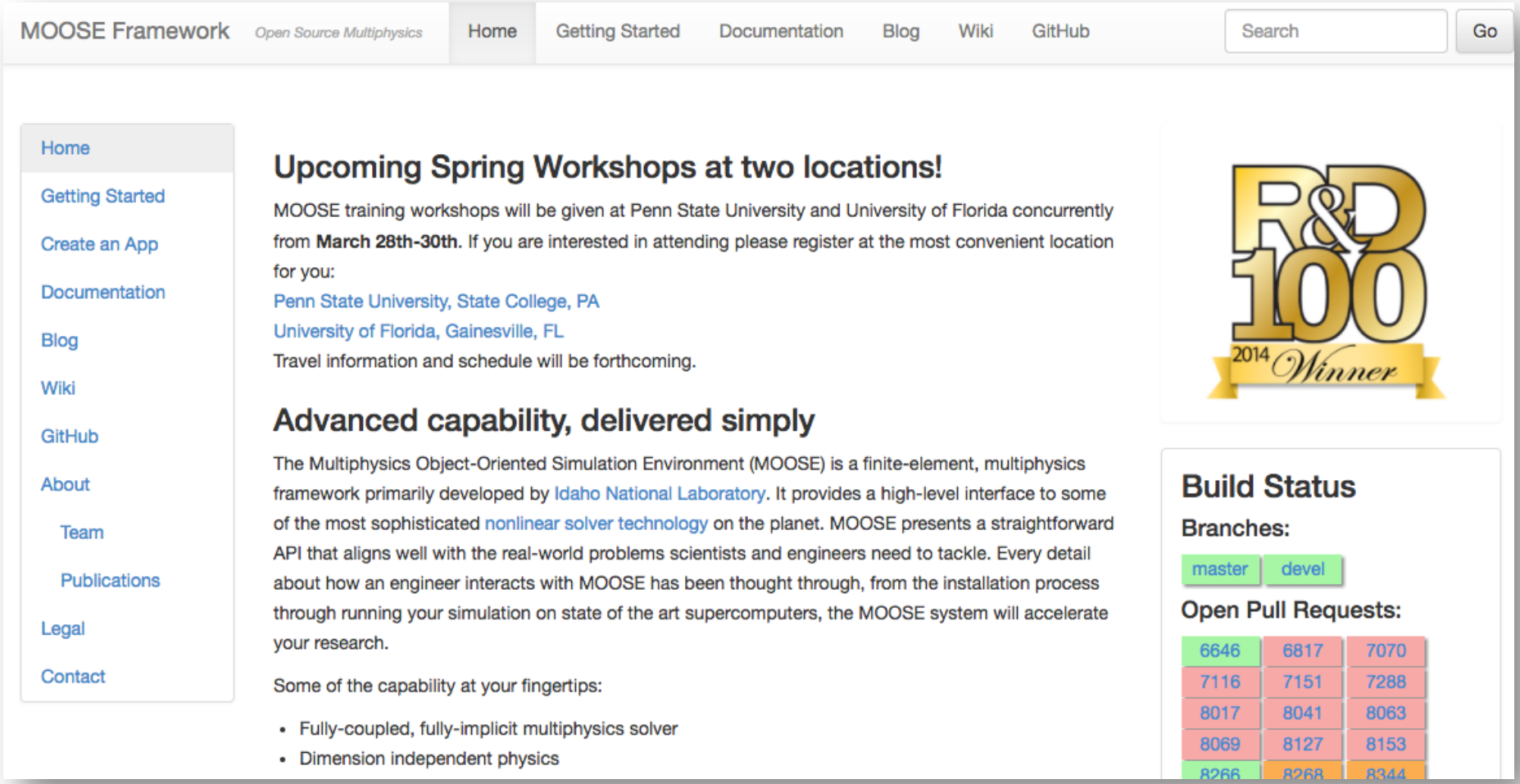


Heat Conduction
- Steady state
- Transient
- Locally varying thermal conductivity

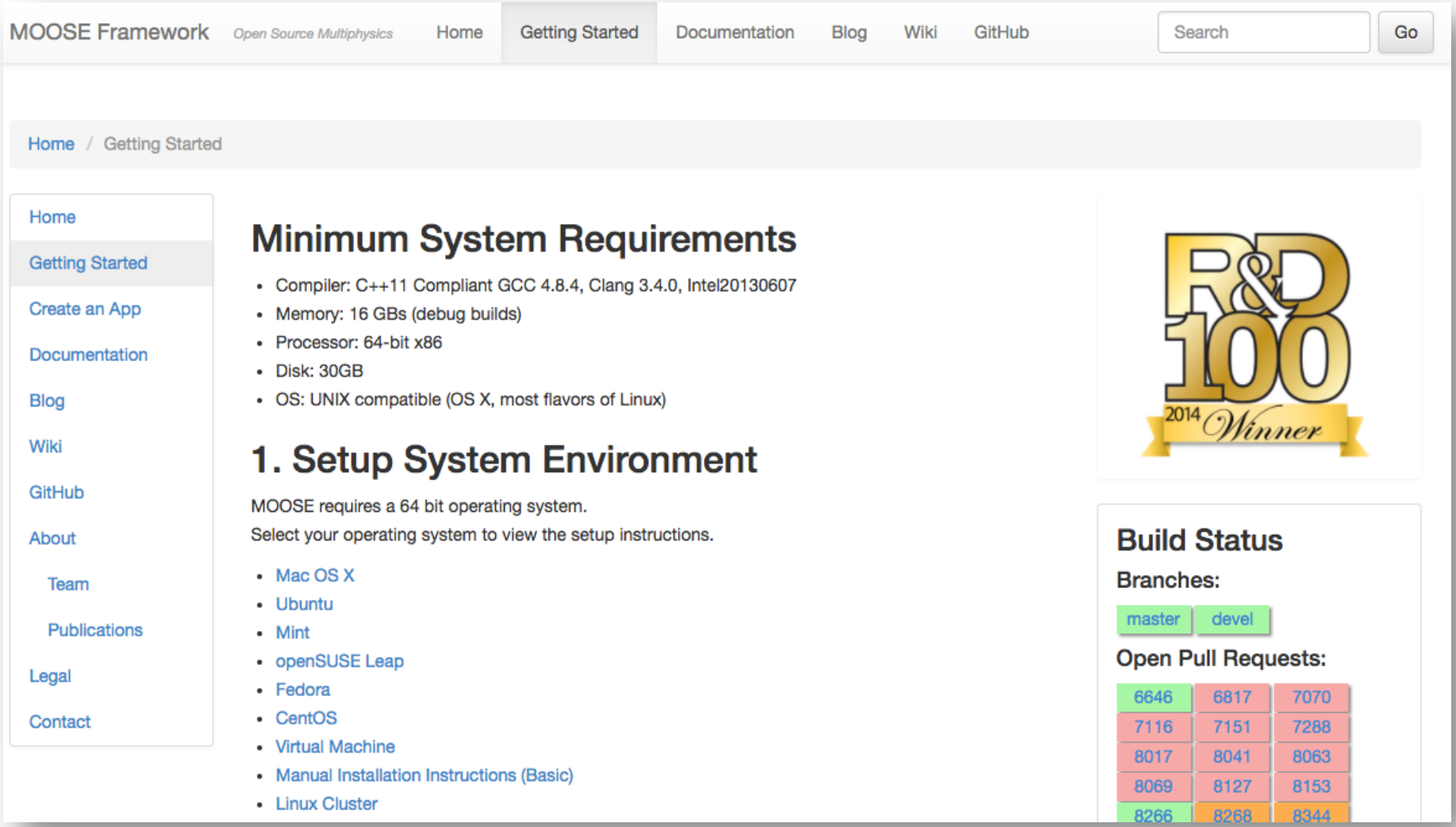


- The various equations are coupled together in one large residual vector
- This vector is used in one Newton or Jacobian-Free Newton Krylov solve
- Partially coupled solves are also available

The primary avenue for learning about and obtaining MOOSE is mooseframework.org



Preparing to use MOOSE

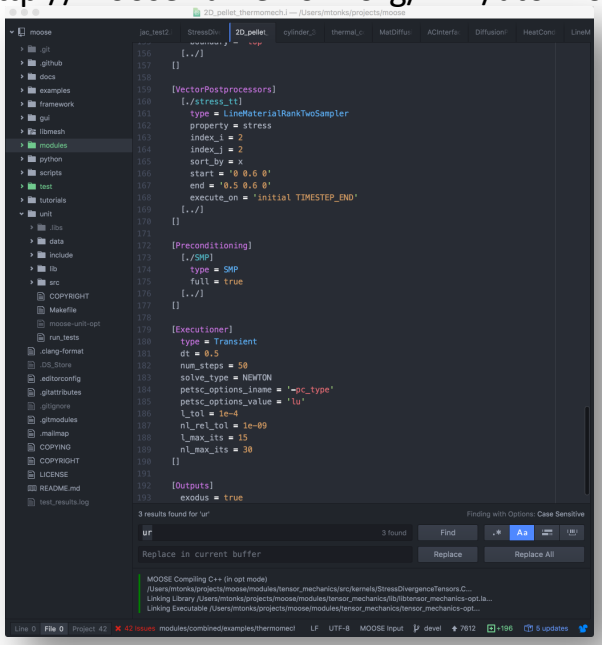


Once you have the code compiled and tested, the steps to run MOOSE are

- Create input file
- Create mesh, if not generated by input file
- Run the code
- Look at the results

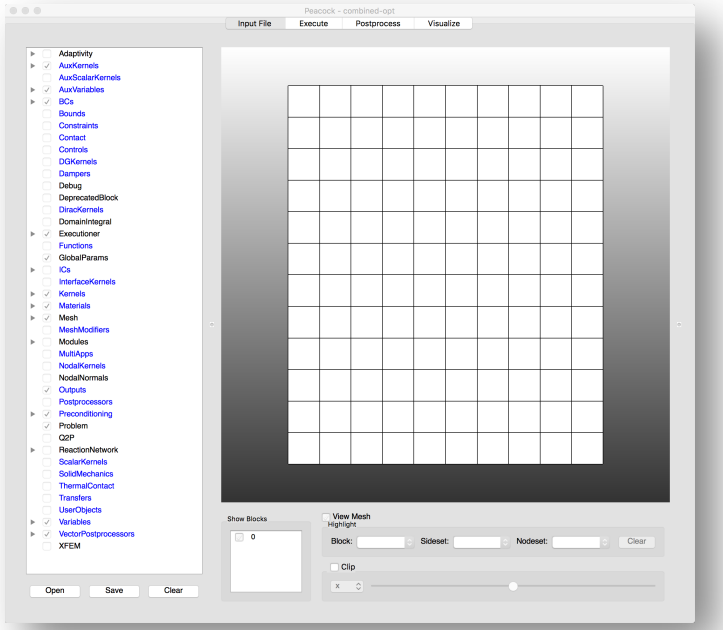
Atom

http://mooseframework.org/wiki/atomio/



PEACOCK (GUI)

http://mooseframework.org/wiki/Peacock/



MOOSE has several blocks in the input file that are required to define the problem to be solved

- **Problem:** Defines basic details about the model, such as the the coordinate system. Often the defaults are fine.
- **Mesh:** Establishes the FEM mesh, whether read from a file or generated.
- **Variables**: Defines the variables that you will solve for
- **Kernels**: Defines the equations are you trying to solve. The equations are often divided into parts
- **Boundary Conditions (BCs):** Establishes the boundary conditions for the problem.
- **Materials:** Computes material properties, stresses, and strains
- **Executioner:** Establishes the type of simulation (transient vs. steady state), the time behavior, and the solution method.
- **Output:** Defines the outputs for the solution defined at the nodes, postprocessors to a .csv file and the output of restart files.

MOOSE also has many other classes that enrich your solution but are not required

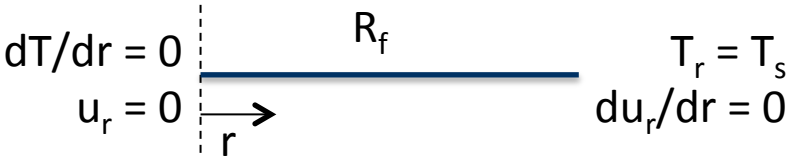
- **Initial Conditions (ics):** The default ic for a variable is a single constant value. More complex ics can be created and assigned for each specific variables.
- **Global Parameters:** When multiple classes in your input file require the same input, they can be defined in the global parameters block.
- **Auxvariables and Auxkernels**: Auxvariables are dependent variables. The are calculated in a corresponding auxkernel. Auxvariables are often used to output the values of material properties (stresses and strains).
- **Postprocessors:** Calculates scalar values over the entire mesh.
- **Preconditioners:** Used when solving multiple coupled variables
- **Functions:** Defines a generic function (can be parsed from the input file) that is then used by something else, like a kernel, BC etc.

Here is an example of an input file for a 2D heat conduction simulation

```
[Problem]
  coord_type = RZ
[]

[Mesh]
  type = GeneratedMesh
  dim = 2
  xmax = 0.5
  ymax = 5
  nx = 10
  ny = 50
[]

[Variables]
  [./T]
  [../]
[]

[Kernels]
  [./htcond]
    type = HeatConduction
    variable = T
    diffusion_coefficient = 0.03
  [../]
  [./Q]
    type = HeatSource
    value = 450
    variable = T
  [../]
[]
```

```
[BCs]
  [./T_right_bottom]
    type = PresetBC
    value = 685
    variable = T
    boundary = 'right bottom'
  [../]
[]

[Executioner]
  type = Steady
  solve_type = NEWTON
  petsc_options_iname = '-pc_type'
  petsc_options_value = 'lu'
[]

[Outputs]
  exodus = true
[]
```

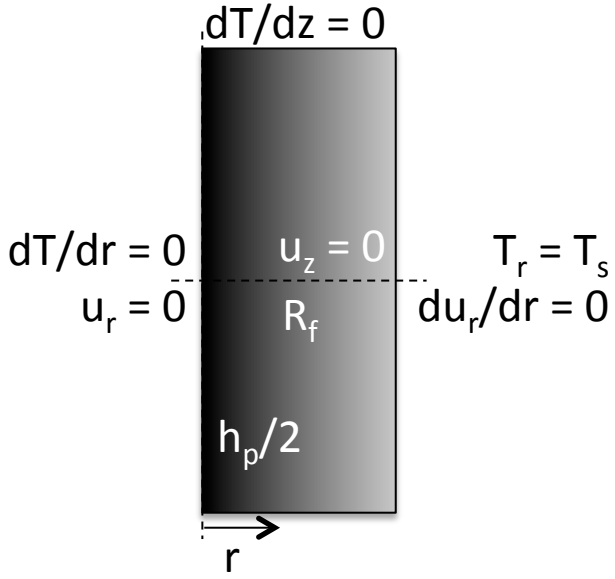
Example of 1D thermomechanics



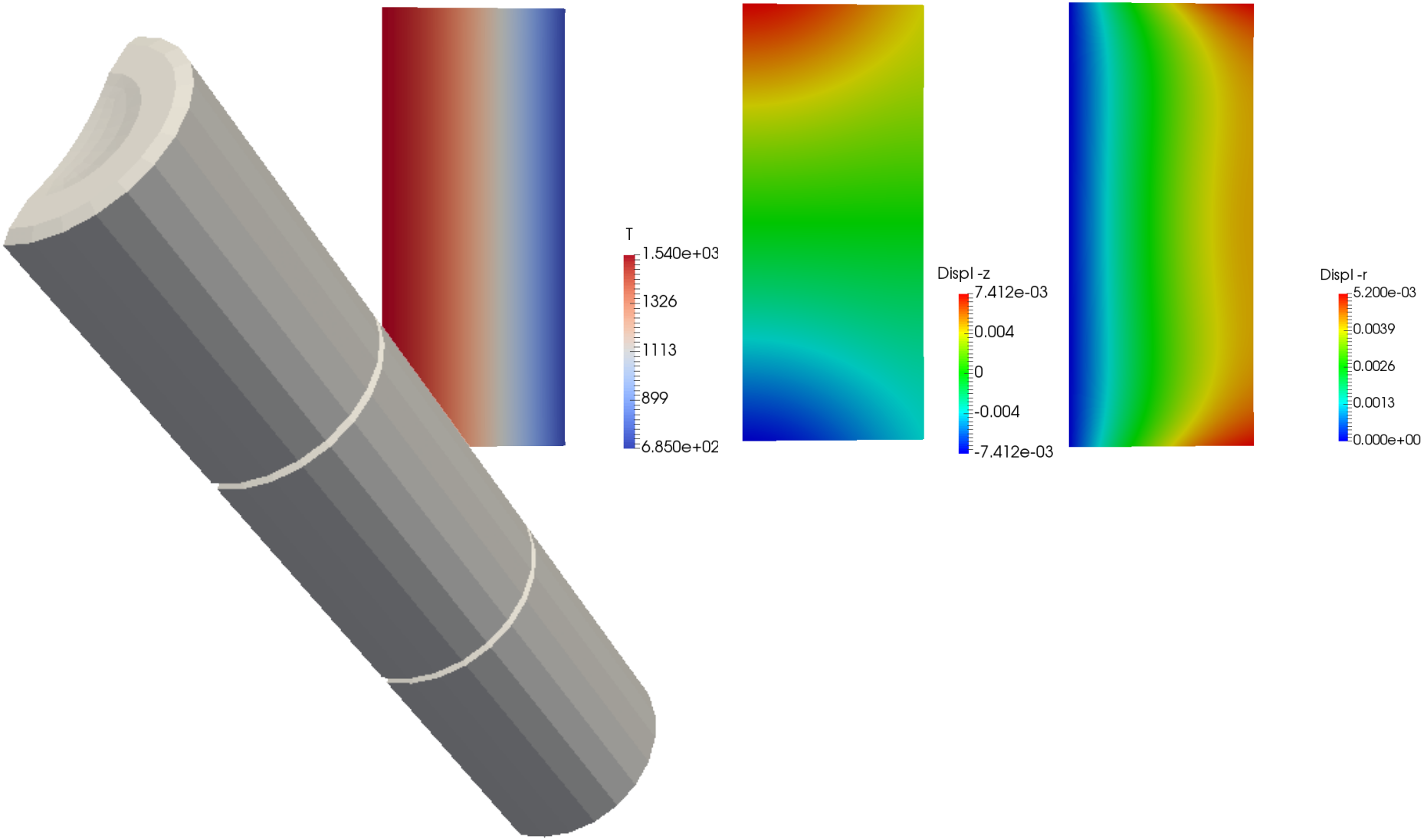
- The initial temperature is set to 273 K
- We will take 50 time steps of 0.5 s
- The full power of Q = 450 begins at time t = 0.
- UO_2 material properties are used for both the thermal and mechanics equations
  - k is a function of temperature
  - Q increases with time

Example of 2D thermomechanics

- The initial temperature is set to 300 K
- We will take 50 time steps of 0.5 s
- The full power of Q = 450 begins at time t = 0.
- UO_2 material properties are used for both the thermal and mechanics equations
  - k is a function of temperature
  - Q increases with time



Why is a fuel pellet dished and chamfered?



Summary

- The coupled thermomechanics equations can be solved using the MOOSE framework
- MOOSE is free and opensource and pretty easy to use
- Pellets are dished and chamfered to reduce interaction between pellets and with the cladding due to thermal expansion