HIGH PERFORMANCE RESEARCH COMPUTING

Technology Lab: FEA with MOOSE

HPRC Short Course
10:00 am
10/18/22

High Performance
Research Computing
DIVISION OF RESEARCH

Overview

- Setting up MOOSE on FASTER
- Building a simple model of heat conduction
- Non-linear heat conduction
- Transient heat conduction
- Anisotropic heat conduction
- Effective thermal conductivity

What is MOOSE

- Multiphysics Object-Oriented Simulation Environment (MOOSE)
- An open-source, parallel finite element framework from Idaho National Laboratory
- MOOSE documentation:
  - https://mooseframework.inl.gov/getting_started/examples_and_tutorials/index.html#examples-and-tutorials
- HPRC documentation on setting up moose:
  - https://hprc.tamu.edu/wiki/SW:moose

MOOSE Physics Modules

- Heat Conduction
- Chemical Reactions
- Tensor Mechanics
- Phase Field
- Navier Stokes
- Fluid-Structure Interactions
- Electromagnetics
- Much More!
- View the MOOSE gallery:
  - https://mooseframework.inl.gov/gallery.html

Accessing FASTER via SSH for TAMU users

- SSH command is required for accessing FASTER:
  - On campus: `ssh userNetID@faster.hprc.tamu.edu`
  - Off campus:
    - Set up and start VPN (Virtual Private Network): u.tamu.edu/VPnetwork
    - Then: `ssh userNetID@faster.hprc.tamu.edu`
  - *Two-Factor Authentication* enabled for CAS, VPN, SSH
- SSH programs for Windows:
  - MobaXTerm (preferred, includes SSH and X11)
  - PuTTY SSH
  - Windows Subsystem for Linux
- https://portal-faster.hprc.tamu.edu/
  - Select the "Clusters" tab and then "_faster Shell Access"
- FASTER has 2 login nodes for TAMU users. Check the bash prompt.

  Login sessions that are idle for **60** minutes will be closed automatically

  Processes run longer than **60** minutes on login nodes will be killed automatically.

  Do not use more than 8 cores on the login nodes!

  Do not use the sudo command.

  hprc.tamu.edu/wiki/HPRC:Access

Accessing FASTER for ACCESS users

- The Advanced Cyberinfrastructure Coordination Ecosystem: Services and Support (ACCESS) is a virtual collaboration funded by the National Science Foundation that facilitates free, customized access to advanced digital resources, consulting, training, and mentorship.
- View the getting started documentation to create an ACCESS account.
- FASTER has 1 login node for ACCESS users
- SSH via Jump Host:

```
ssh -J username@faster-jump.hprc.tamu.edu:8822 username@login.faster.hprc.tamu.edu
```

Hands On Activity

- Please login to FASTER now
- **`cp -r /scratch/training/Techlab-MOOSE/ $SCRATCH/Techlab-MOOSE`**
- **`cd $SCRATCH/Techlab-MOOSE`**

Setting up MOOSE

- After logging in to FASTER enter the following commands

- **`module purge`**
- **`module load MOOSE`**

Setting up MOOSE

- Navigate to your SCRATCH directory and choose a name for your MOOSE folder

- **`cd $SCRATCH`**
- **`stork.sh my_moose`**
- **`cd my_moose`**

Setting up MOOSE

- Open the Makefile and replace *no* with *yes* for the needed modules.
- **`vi Makefile`**
- Change:
  - ALL_MODULES := no to ALL_MODULES := yes

Setting up MOOSE

- Build the MOOSE executable, my_moose-opt
- **`make -j 8`**
- Once this finishes run the tests to confirm it's working
- **`./run_tests`**
- Make sure it says all tests passed!
- The executable my_moose-opt is what you will need to call when you want to run MOOSE.

Job Submission File

- Lets take a look at a SLURM job submission file for MOOSE.
- **`vi submit.job`**

Finite Element Analysis

- A method for numerically approximating the solution to partial differential equations.
- Works by finding a solution function made up of "shape functions" that are multiplied by coefficients and added together.
- FEM is widely applicable for a large range of PDEs and domains

Building a Model of Heat Conduction

Heat Conduction Equation:

$$\rho(t, x)c(t, x)\,\partial t/\partial T = \nabla k(t, x)\nabla T + q$$

where, T is temperature, t is time,
X is the vector of spatial coordinates, ρ is the density, c is the specific heat capacity, k is the thermal conductivity, and q is a heat source.

Steady State Heat Conduction (only conduction term)

$$0 = \nabla k(t, x)\nabla T$$

Hands On Activity Ex01

- Add the missing lines in Ex01.in as we go through the following information.
- You can check the solution in 1D_Linear_SS.i if needed.

Building a Model of Heat Conduction

```
[Mesh]
  type = GeneratedMesh
  dim = 1
  nx = 100
  ny = 0
  nz = 0
  xmin = 0
  xmax = 1
[]
```

- Mesh block will define our system
- Create a 1D linear mesh with GeneratedMesh
- 100 nodes
- X is between 0 and 1
- [MESH] is REQUIRED!

Building a Model of Heat Conduction

```
[Variables]
  [./T] # Temperature
        family = LAGRANGE
        order = FIRST
  [../]
[]
```

- Variables block defines variables that will be solved for in the model.
- The default LAGRANGE variable type is defined at nodes, and interpolated within an element.
- [Variables] is REQUIRED!

Building a Model of Heat Conduction

```
[Kernels]
  [./heat_diffusion]  # evaluate the Laplacian
of temperature
    type = HeatConduction
    variable = T
  [../]
[]
```

- Evaluate the terms that appear in the PDE
- Generally a separate term is needed for every term in the PDE
- [Kernels] is REQUIRED!

Building a Model of Heat Conduction

```
[Materials]
  [./Thermal_Conductivity]
    type = GenericFunctionMaterial
    prop_names = thermal_conductivity
    prop_values = thermal_conductivity_func
  [../]
[]
```

- Defines material properties
- We are assigning thermal conductivity to our defined function in the [Functions] block
- [Materials] is REQUIRED!

Building a Model of Heat Conduction

```
[BCs]
  [./T_left] # T_a
    type = DirichletBC
    variable = 'T'
    boundary = 'left'
    value = 1.0
  [../]
  [./T_right] # T_b
    type = DirichletBC
    variable = 'T'
    boundary = 'right'
    value = 2.0
  [../]
[]
```

- BCs block lets us setup boundary conditions on our system
- DirichletBC assigns a value to T along the left and right boundaries
- Can be a constant or a function
- [Bcs] is REQUIRED!

Building a Model of Heat Conduction

```
[Executioner] # Solver options
type = Steady
solve_type = 'LINEAR'
petsc_options_iname = '-pc_type -sub_pc_type'
#mpi only runs with type asm and subtype lu
petsc_options_value = 'asm lu'
l_max_its = 100
l_tol = 1.0e-6
[]
```

- Contains parameters relevant to solver
- Type = Steady or Transient
- Some petsc options only work in serial!
- You can adjust the linear solver iterations and tolerance depending on your accuracy needs
- [Executioner] is REQUIRED!

Building a Model of Heat Conduction

```
[Outputs]
  file_base = 1D_linear_conduction
  exodus = true
  csv = true
[]
```

- Lets you choose how you would like to create output files
- Exodus files can be read my commonly used visualization software
- [Outputs] is REQUIRED!

Building a Model of Heat Conduction

```
[Functions]
  # Thermal_Conductivity of Material
  [./thermal_conductivity_func]
    type = ParsedFunction
    value = '1.0'
  [../]
[]
```

- Functions block lets you define functions you can use later.
- In this case we are using it to define thermal conductivity (makes it easy to change later)

Building a Model of Heat Conduction

```
[AuxVariables]
  # this is just to visualize the thermal conductivity of material over the domain
  [./mat_therm_cond]
    order = CONSTANT
    family = MONOMIAL
  [../]
  # the temperature gradient
  [./dTdx]
    order = CONSTANT
    family = MONOMIAL
  [../]
  # the heat flux
  [./jx]
    order = CONSTANT
    family = MONOMIAL
  [../]
[]
```

- AuxVariables do not directly appear in the PDE being solved.

```
[AuxKernels] # the kernels that evaluate the values of the auxvariables
  [./mat_therm_cond]
    type = MaterialRealAux
    variable = mat_therm_cond
    property = thermal_conductivity
  [../]
  [./dTdx]
    type = VariableGradientComponent
    variable = dTdx
    gradient_variable = T
    component = x
  [../]
  [./jx]
    type = ParsedAux
    variable = jx
    args = 'mat_therm_cond dTdx'
    function = '-mat_therm_cond*dTdx'
  [../]
[]
```

Building a Model of Heat Conduction

```
[VectorPostprocessors]
  [./x_direction]
    type =  LineValueSampler
     start_point = '0 0 0'
     end_point = '1 0 0'
     variable = 'T jx dTdx'
     num_points = 101  # n_x + 1 (# of nodes)
     sort_by =  id
  [../]
[]
```

- VectorPostprocessors computes a vector of values
- LineValueSampler allows us to sample values along a line
- Make sure you have csv = True in your output block

Hands on Activity Ex01

- Change the job file to read our moose input file and use our previously created executable
- **`vi submit.job`**
- Submit the job and view the output
- **`sbatch submit.job`**

Nonlinear Heat Conduction Calculation

- Change the thermal conductivity values to be nonlinear
- k(T) : k is a function of T
- We will use a different kernel and material type to implement this

$$0 = \nabla k(t, x) \nabla T$$

Steady state heat conduction

$$0 = \nabla \cdot D(c,a,b,\ldots) \nabla u$$

Steady state diffusion

Hands On Activity Ex02

- Add the missing lines in Ex02.in as we go through the following information.
- You can check the solution in 1D_NonLinear_SS.i if needed.

Nonlinear Heat Conduction Calculation

```
[Kernels]
  [./symbolic_diffusion_kernel]
    type = MatDiffusion   # evaluate the divergence of a flux
    variable = T
    diffusivity = k
  [../]
[]
```

Nonlinear Heat Conduction Calculation

```
[Materials]
  [./thermal_conductivity]
    type = DerivativeParsedMaterial
    f_name = k
    function = 'a:=1.0;b:=10.0;n:=10;a+b*T^n'
    args = 'T' # arguments (the variables that k depends on)
    derivative_order = 1
  [../]
[]
```

Hands On Activity Ex02

- Change the job file to read our moose input file and use our previously created executable
- **`vi submit.job`**
- Submit the job and view the output
- **`sbatch submit.job`**

Transient Heat Conduction

$$\rho(t, x)c(t, x)\,\partial t/\partial T = \nabla k(t, x)\nabla T + q$$

where, T is temperature, t is time,
X is the vector of spatial coordinates, ρ is the density, c is the specific heat capacity, k is the thermal conductivity, and q is a heat source.

Hands On Activity Ex03

- Add the missing lines in Ex03.in as we go through the following information.
- You can check the solution in 1D_Linear_Transient.i if needed.

Transient Heat Conduction

```
[Kernels]
  [heat_conduction]
    type = HeatConduction
    variable = T
  []
  [time_derivative]
    type = HeatConductionTimeDerivative
    variable = T
  []
 [heat_source]
    type = HeatSource
    variable = T
    value = 1e4
  []
[]
```

Transient Heat Conduction

```
[BCs]
  [./t_left]
    type = DirichletBC
    variable = 'T'
    boundary = 'left'
    value = 300.0
  []
    [./t_right]
      type = DirichletBC
      variable = 'T'
      boundary = 'right'
      value = '300+5*t'
    []
  []
```

```
[Executioner] # Solver options
  type = Transient
  solve_type = 'LINEAR'
  petsc_options_iname = '-pc_type -sub_pc_types'
  petsc_options_value = 'asm lu'
  l_tol = 1e-9
  num_steps = 20
  dt = 2.0
[]
```

Hands On Activity Ex03

- Change the job file to read our moose input file and use our previously created executable
- **`vi submit.job`**
- Submit the job and view the output
- **`sbatch submit.job`**

Hands On Activity Ex04

- Add the missing lines in Ex04.in as we go through the following information.
- You can check the solution in 2D_Linear_SS.i if needed.

2D Steady State Heat Conduction

- Try changing the 1D mesh to 2D by editing the Mesh block.
- Add a NodalValueSampler to the VectorPostprocessors block.

```
[VectorPostprocessors]
  [./nodal_val]
    type =  NodalValueSampler
     variable = 'T'
     sort_by =  id
  [../]
[]
```

Hands On Activity Ex04

- Change the job file to read our moose input file and use our previously created executable
- **`vi submit.job`**
- Submit the job and view the output
- **`sbatch submit.job`**

Hands On Activity Ex05

- Add the missing lines in Ex05.in as we go through the following information.
- You can check the solution in 2D_Aniso.i if needed.

2D Steady State Anisotropic Heat Conduction

- k will be defined as a rank 2 tensor

$$\begin{bmatrix} k_{xx} & k_{xy} \\ k_{xy} & k_{yy} \end{bmatrix}$$

```
[Materials]
  [./k] # anisotropic conductivity/diffusivity tensor
    type = ConstantAnisotropicMobility
    tensor = '0.5 0.3 0.0
              0.3 0.5 0.0
              0.0 0.0 0.0'
    M_name = k
  [../]
[]
```

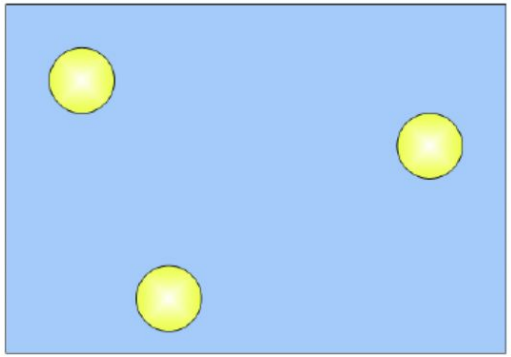
2D Steady State Anisotropic Heat Conduction

```
[Kernels]
  [./heat_diffusion] # with anisotropic conductivity/diffusivity tensor
    type = MatAnisoDiffusion
    diffusivity = k # change to Diffusivity = k (in recent MOOSE update)
    variable = T
  [../]
[]
```

Hands On Activity Ex05

- Change the job file to read our moose input file and use our previously created executable
- **`vi submit.job`**
- Submit the job and view the output
- **`sbatch submit.job`**

Maxwell model for effective thermal conductivity



- Maxwell Model of effective conductivity
- Holds reasonably well under 0.25 particle fraction

Figure 1: Dilute concentration of spherical particles embedded in a continuous matrix. The Maxwell model assumes a lack of thermal interaction between the embedded spheres.

$$\frac{\sigma_e}{\sigma_1} = \frac{(d-1)(1-\phi_2)}{d-\phi_1} \qquad \text{if} \quad \sigma_2/\sigma_1 = 0.$$

Karol Pietrak, Tomasz S. Wisniewski, "A review of models for effective thermal conductivity of composite materials" *Journal of Power Technologies* 95 (1) (2015) 14–24

Modeling 2D Effective Thermal Conductivity

- Apply constant temperatures to the left and right side of the system
- No heat transfer between the top and bottom of the system
- Solve a steady state heat conduction equation to solve for effective thermal conductivity.

$$0 = \nabla (k \nabla T)$$

Hands On Activity Ex06

- Add the missing lines in Ex06.in as we go through the following information.
- You can check the solution in 2D_Effective.i if needed.

Effective Thermal Conductivity

```
[Materials]
  [./Thermal_Conductivity]
      type = ParsedMaterial
      block = 0
      constant_names = 'k_b k_p2'
      constant_expressions = '1.0 0.001'
      function = 'if(phase2=1,k_p2,k_b)'
      # No interface thermal resistance
      f_name = thermal_conductivity
      args = phase2
    [../]
[]
```

Effective Thermal Conductivity

```
[ICs] #Sets the IC for the auxvariable that is used to distinguish between matrix and 2nd phase particle
  [./inclusion]
type = SpecifiedSmoothCircleIC
     variable = phase2
     invalue = 1.0
     outvalue = 0.00
     radii = 144.47 #(~25% area fraction: try different fractions up to the percolation fraction of 0.785)
     int_width = 0.0 # sharp interface
  x_positions = '256'
  y_positions = '256'
  z_positions = '0'
    block = 0
  [../]
[]
```

Effective Thermal Conductivity

- K_eff will be calculated in the postprocessor block.

```
  [./k_eff] #Effective thermal
conductivity
  type = ThermalConductivity
  variable = T
  flux = jx
  length_scale = 1.0
  T_hot = T_hot
  dx = 512
  boundary = right
  [../]
```

Hands on Activity Ex05

- Change the job file to read our moose input file and use our executable
- **`vi submit.job`**
- Submit the job and view the output
- **`sbatch submit.job`**

Effective Thermal Conductivity

```
[ICs]
  [./inclusion]
    type = LatticeSmoothCircleIC
    variable = phase2
    invalue = 1.0
    outvalue = 0.00
    circles_per_side = '5 5' # try different number of particles
    pos_variation = 97.0
    rand_seed = 3000
    radius = 7.65 # try different radii
    int_width = 1.25 # thin interface with a thermal resistance
    radius_variation = 0.7650
    radius_variation_type = 'uniform'
    avoid_bounds = True
  [../]
[]
```

Effective Thermal Conductivity

```
[ICs] #Sets the IC for the second constant phase
  [./inclusion] #Creates circles with smooth interfaces at random locations
    variable = phase2
    type = MultiSmoothCircleIC
    int_width = 0.0
    numbub = 15
    bubspac = 18
    radius = 16
    outvalue = 0
    invalue = 1
    block = 0
  [../]
[]
```

HIGH PERFORMANCE RESEARCH COMPUTING
TEXAS A&M UNIVERSITY

https://hprc.tamu.edu

HPRC Helpdesk:

help@hprc.tamu.edu
Phone: 979-845-0219

Please Help us help you. Please include details in your request for support, such as, Cluster (Faster, Grace, Terra, ViDaL), NetID or ACCESS ID (UserID), Job information (Job ID(s), Location of your jobfile, input/output files, Application, Module(s) loaded, Error messages, etc), and Steps you have taken, so we can reproduce the problem.

Acknowledgements

This course is supported by NSF 2112356: ACES - Accelerating Computing for Emerging Sciences