GeMA – Coupled Thermo-Mechanical example

08/08/2018 – Version 1.0

Tecgraf
PUC-RIO

Example purpose

This example shows:

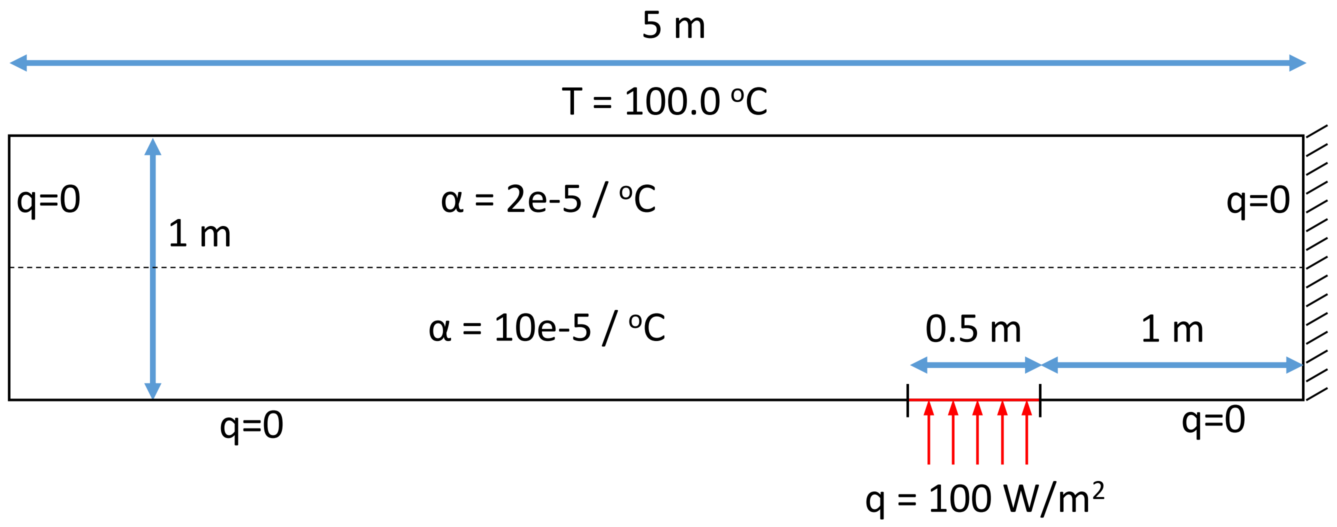
- How to setup a thermo-mechanical analysis by composition of the thermo, mechanical and a coupling physics.

- How to recover principal stress values

Simulation file: BendingBar.lua

The Problem

- This example analyses the heat conduction on a bar, coupled with thermal stress calculation. Bar top is kept at a constant temperature. A small part of the bottom border is heated while the other sides are insulated. The right side of the bar is fixed. The bottom half has a thermal expansion coefficient 5 times greater than top half, making the bar bend upwards.



Model file: Material properties

Needed material properties include required thermo, mechanical and coupling parameters.

```
-- Cell properties
PropertySet
{
  id          = 'MatProp',
  typeName    = 'GemaPropertySet',
  description = 'Material parameters’,

 properties  = {
    {id = 'k',     description = 'Conductivity',      unit = 'W/(m.K)'},
    {id = 'E',     description = 'Elasticity modulus'},
    {id = 'nu',    description = 'Poisson ratio'},
    {id = 'alpha', description = 'Thermal expansion factor',  unit = '1/K', format = '.6f'},
    {id = ‘h',     description = 'Element thickness', unit = 'm'},
  },
  values = {
    {id = 'normal', k = 1.0, h = 0.01, E = 1e6, nu = 0.3, alpha = 2e-5},
    {id = 'high',   k = 1.0, h = 0.01, E = 1e6, nu = 0.3, alpha = 10e-5},
  }
}
```

Thermo property

Elastic mechanical properties

TM coupling property

Two different materials

Model file: Principal stresses

Principal stresses are calculated by user defined functions provided by the StressFunctionsLib and are added to the mesh by two user defined attributes.

```
SharedCodeBegin{}
```

Marks the begining of a code block with functions that might be called from several threads. A good practice when using stressFunctionsLib.

```
dofile('$SCRIPTS/stressFunctionsLib.lua')

-- Declares user cell functions for calculating principal stresses on
-- Gauss points using the stress functions lib
CellFunction(stressFunctionsLib.cellFunctionDef('sigma1'))
CellFunction(stressFunctionsLib.cellFunctionDef('sigma2'))
```

$$\sigma_{1,2} = \frac{\sigma_{xx} + \sigma_{yy}}{2} \pm \sqrt{\frac{(\sigma_{xx} - \sigma_{yy})^2}{4} + \sigma_{xy}^2}$$

```
SharedCodeEnd{}
```

Ends the shared code block initiated above

```
Mesh
{
  ...

  -- Principal Stresses
  gaussAttributes = {
    {id = 's1', description = 'Sigma 1', functions = true, defVal = 'sigma1'},
    {id = 's2', description = 'Sigma 2', functions = true, defVal = 'sigma2'}
  },

  ...
}
```

User attributes for storing principal stresses

Attribute values will be given by the user functions declared above with the help of the Stress Functions lib.

Solution file: Physics

The Thermo-Mechanical coupling in GeMA is solved by a composition of three physics: the thermos physics, the Mechanical physics and the TM coupling physics.

```
-- Physics for temperature calculation
PhysicalMethod {
  id       = 'HeatPhysics',
  typeName = 'ThermoFemPhysics',
  type     = 'fem',

  mesh               = 'mesh',
  boundaryConditions = {'Tbc1', 'Tbc2'},
}

-- Physics for displacement and stress calculation
PhysicalMethod {
  id       = 'StressPhysics',
  typeName = 'MechanicalFemPhysics.PlaneStress',
  type     = 'fem',

  mesh               = 'mesh',
  boundaryConditions = {'ubc'},
  stressMode = 'gauss',
}
```

```
-- Coupling physics
PhysicalMethod {
  id       = 'TempStressPhysics',
  typeName = 'CoupledTMFemPhysics.PlaneStress',
  type     = 'fem',

  mesh       = 'mesh',
  referenceT = 100,       -- Reference temperature
                          -- for the material
  stressMode = 'gauss',
}
```

Stresses should be recovered
on Gauss points only

Solution file: Orchestration script

The orchestration script is very similar to every other example using the linear fem solver. The only noteworthy feature is the physics composition.

```
function ProcessScript()
  -- Solve the problem using the FEM method with a temperature physics, a stress physics
  -- and a coupling physics
  fem.solve({'HeatPhysics', 'StressPhysics', 'TempStressPhysics'}, 'solver')

  print('')
  print('Calculated results:')

  -- Print calculated node data
  io.printMeshNodeData('mesh', {'T', 'u'}, {header_title = true, eval_functions = true})

  -- Print calculated Gauss data
  io.printMeshGaussData('mesh', {'S', 's1', 's2'}, {header_title = true, eval_functions = true})

  -- Save data to a Neutral file
  io.saveMeshFile('mesh', '$SIMULATIONDIR/out/$SIMULATIONNAME.pos', 'nf',
                  {'u', 'T'}, {'S', 's1', 's2'}, {saveDisplacements=true})

end
```

Composition

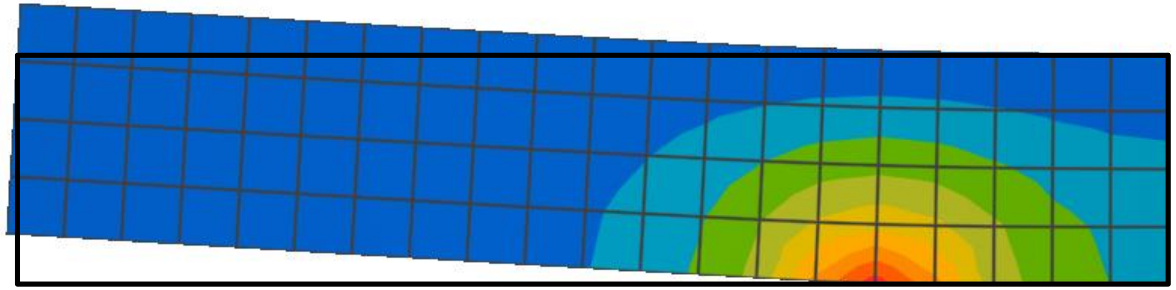
Ensure that principal stress values are calculated for printing

Stress values calculated by the physics set

Principal stress values calculated by user functions

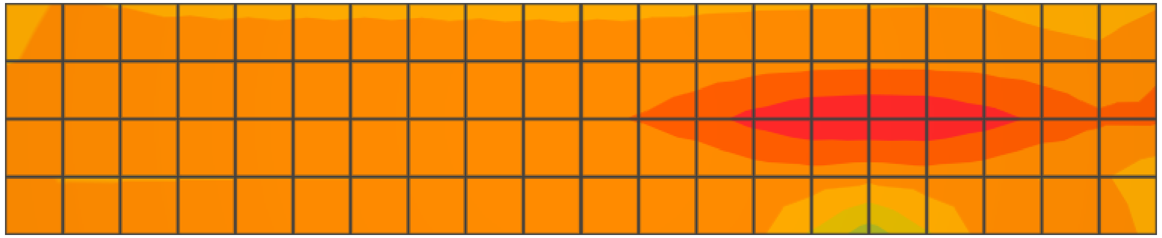
Results

Temperature and deformation (exaggeration = 10x)



Principal Stresses

σ_1



σ_2