Simulation of Nuclear Fuel Using MOOSE Framework

[Project Report]

ABSTRACT

With the current trend in increasing the power generated from the nuclear powers, the safety issue arises as a major concern. Having a safe operation means mainly that the nuclear fuel will accommodate the changes occur inside the reactor during its whole life without failing in a catastrophic way that may lead to the release of radioactive material to the coolant. One way to check and investigate the integrity of the nuclear fuel is done by using simulation to simulate and predict the nuclear fuel behavior under different conditions, i.e., steady-state, transient and accidental. In order to fully characterize the fuel behavior, a multiscale and multiphysics code should be used. One of the promising tools in this context is MOOSE framework which is a parallel finite element framework.

In this work MOOSE has been utilized to study and simulate the thermomechanical behavior of UO_2 nuclear fuel under simple conditions. The differences between the steady-state and the transient heat generation are investigated. Also, the effect of the axial dependence of the linear heat generation (LHR) on the overall thermal behavior is addressed by comparing it to the uniform LHR under steady-state and transient conditions. The combination between the thermal and the mechanical models is achieved by studying the stress induced by the thermal expansion of a UO_2 fuel pellet.

1. INTRODUCTION

The Multiphysics Object-Oriented Simulation Environment (MOOSE) framework is an open-source parallel finite element framework. This framework is an object-oriented C++ finite element that has been developed by Idaho National Laboratory. The main objective of MOOSE is to simplify the development of the advanced numerical applications which provides a high-level interface to sophisticated nonlinear solvers. These advantages of MOOSE made it a wonderful candidate for using in different applications, such as modeling thermomechanics, neutronics, microstructure modeling, etc.

The behavior of the nuclear fuel is very complex because of the interfered physics and the harsh and the unique environment that fuel works in. This complexity should be addressed properly in order to have a reliable simulation and prediction of the fuel behavior. Studying the thermal, mechanical, and thermomechanical behavior is the first step in the way of the fully characterization of the nuclear fuel. In the first part of this work, the temperature profile of a section in the nuclear fuel rod is solved for two cases. The first is for a constant LHR and the second is for transient LHR. In the second part, the axial variation of the LHR has been consider for both cases, i.e. steady-state and transient. In the third part, both thermal and mechanical models have been combined together to calculate the stress induced due to the thermal expansion.

2. TEMPERATURE PROFILE FOR AXISYMMETRIC LHR (Part-I)

2.1. Problem Description

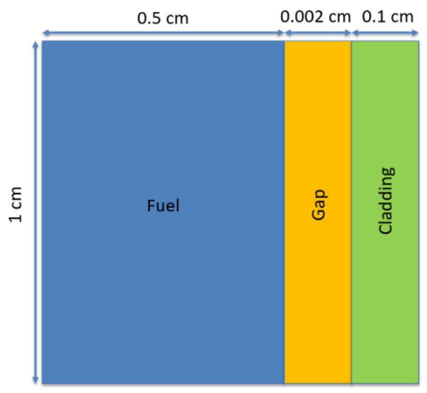

Figure 1 Dimensions of UO_2 fuel for part-I

The case considered in this part is for a UO_2 fuel with gap filled with He gas and Zr-based alloy for cladding. The dimensions of the 2-D problem are shown in figure 1. Because of the symmetry in the axial direction, this is a 1-D problem, however, it's defined as a 2-D in the MOOSE with RZ coordinates. To solve this problem, we need only thermal conductivity as material property for the fuel, gap, and clad. However, and as it will be described below, density and specific heat are defined as well as they be required for next problems. The numerical values of the material properties used in solving this case are listed in table 1. For this problem the thermal conductivity has been assumed to be constant with temperature. However, there is a huge dependence of thermal conductivity on temperature, but it will be ignored in this moment. For this case, we will have 2 boundary conditions. The first is for the temperature value at the outer cladding surface which will be assumed to be constant and equals 500K. The other boundary condition is for the temperature gradient at the center of the fuel pellet (left side) is constant and equals to zero. This BC will provide the problem with the necessary symmetry as we expect the other half of the section to be symmetrical to the one we solve for. For this part, we have 2 cases. The first is with LHR = 150 W/cm, and the second is with LHR given by the following equation:

$$LHR(t) = 150 \times (1 - \exp(-0.05 \times t)) + 150$$

Table 1. Numerical values for materials properties used in Part-I

Material	Density (g/cm^3)	C$_p$ (J/g.K)	K (W/cm.K)	α (K^{-1})
UO_2	10.98	0.33	0.03	1.2×10^{-5}
He	0.178×10^{-3}	5.193	0.0026	1.0×10^{-5}
Zr	6.5	0.35	0.17	

2.2. Input File

In this section we will briefly describe each block in the input file. Most of these blocks will be common in the next problems with slight changes in the numerical values and will not discussed again. For both cases, i.e. steady-state and transient, we need to define our meshing system in the mesh object. For this problem, we used the built-in meshing system in MOOSE as the problem is simple. In the first block we generate meshes over the whole domain and then create 2 subdomains and assign one to the gap and one to the clad. The problem is assumed to be 2D. As we expect the problem to be symmetrical in the axial direction, the number of meshes in the y-direction (ny) is much lower than the x-direction (nx). We define to subdomain (boxes) using two point, i.e. bottom_left and top_right. Each block will have a unique id to refer to. So the fuel is block 0, gap is block 1 and clad is block 2.

```
[Mesh]
  [fuel]
    type = GeneratedMeshGenerator
    dim = 2
    nx = 300
    ny = 50
    xmax = 0.602
    ymax = 1
  []
  [gap]
    input = fuel
    type = SubdomainBoundingBoxGenerator
    bottom_left = '0.5 0 0'
    top_right = '0.602 1 0'
    block_id = 1
  []
  [clad]
    input = gap
    type = SubdomainBoundingBoxGenerator
    bottom_left = '0.502 0 0'
    top_right = '0.602 1 0'
    block_id = 2
  []
[]
```

To define the materials properties required to compute the thermal properties for each block we used the object `ADGenericConstantMaterial` separately for each block.

```
[Materials]
  [propertyf]
    type = ADGenericConstantMaterial
    block = 0
    prop_names = 'fuel_density fuel_specific_heat fuel_thermal_conductivity'
    #prop_names = 'density specific_heat thermal_conductivity'
    prop_values = '10.980 0.33 0.03'
  []
  [propertyg]
    type = ADGenericConstantMaterial
    block = 1
    prop_names = 'gap_density gap_specific_heat gap_thermal_conductivity'
    #prop_names = 'density specific_heat thermal_conductivity'
    prop_values = '0.178e-3 5.193 0.0026'
  []
  [propertyc]
    type = ADGenericConstantMaterial
    block = 2
    prop_names = 'clad_density clad_specific_heat clad_thermal_conductivity'
    #prop_names = 'density specific_heat thermal_conductivity'
    prop_values = '6.5 0.35 0.17'
  []
[]
```

As we aim to solve for the temperature profile, the temperature has been assigned as a variable with initial condition = 400. This value has been used as we expect the surface temperature at steady state to be close to 400K.

```
[Variables]
  [T]
    initial_condition = 300
  []
[]
```

The kernels used in this problem are of two types. The first is the `ADHeatConduction` which is used to solve the heat equation, so we have to define kernel for each block along with the referring to the value of the thermal conductivity defined in the materials block. The second type of kernels is the which is used to assign the volumetric heat generation for the fuel block. The value of the volumetric heat generation is constant and calculated based on the LHR and the fuel radius using the following formula:

$$Q\left(W/cm^3\right)=\frac{LHR\left(W/cm\right)}{\pi R_f^2\left(cm^2\right)}$$

```
[Kernels]
  [conduction_fuel]
    type = ADHeatConduction
    variable = T
    block = 0
    thermal_conductivity = fuel_thermal_conductivity
  []
  [conduction_gap]
    type = ADHeatConduction
    variable = T
    block = 1
    thermal_conductivity = gap_thermal_conductivity
  []
  [conduction_clad]
    type = ADHeatConduction
    variable = T
    block = 2
    thermal_conductivity = clad_thermal_conductivity
  []
  [heat_source]
    type = HeatSource
    variable = T
    value = 190.98
    block = 0
  []
[]
```

The next block is the BCs. As we mentioned we have 2 BCs. The first is for the outer cladding temperature is constant and the second is for the temperature gradient at the center of the fuel rod.

```
[BCs]
  [inlet]
    type = NeumannBC
    variable = T
    boundary = left
    value = 0
  []
  [outlet]
    type = DirichletBC
    variable = T
    boundary = right
    value = 500
  []
[]
```

For this block, we define the solution type and the solver used PJFNK, which is the default solver.

```
[Executioner]
  type = Steady
  solve_type = PJFNK
[]
```

The other blocks some of them are optional and other used to define the problem and the output of the code.

For the **transient** case, we need to define the volumetric heat function, which is time-dependent.

```
[Functions]
  [heat_fn]
    type = ParsedFunction
    value = 150*(1-exp(-0.05*t))+150
  []
[]
```

Also, we need to add another kernel that that implement a time derivative $\rho c_p \frac{\partial T}{\partial t}$ of the heat equation for each block and define the specific heat and the density of each block.

```
  [conduction_derivative_fuel]
    type = ADHeatConductionTimeDerivative
    variable = T
    specific_heat = fuel_specific_heat
    density_name = fuel_density
    block = 0
  []
```

For the heat source, we need to change the kernel from a constant value to a function as described earlier.

```
  [heat_source]
    type = HeatSource
    variable = T
    function = heat_fn
    block = 0
  []
```

The solution type is defined as transient, and the time step is 1 sec and the end time is 100 sec.

```
[Executioner]
  type = Transient
  solve_type = PJFNK
  end_time = 100
  dt = 1
[]
```

2.3. Part-I Results

Figure 2 shows the different blocks and the radial temperature profile. As expected the maximum temperature is at the center of the pellet and it decreases gradually as we moves away from the center. The change in the temperature in the gap and clad is linear and outer temperature is attained at 500K, which is the BC. The temperature profile is constant axially.

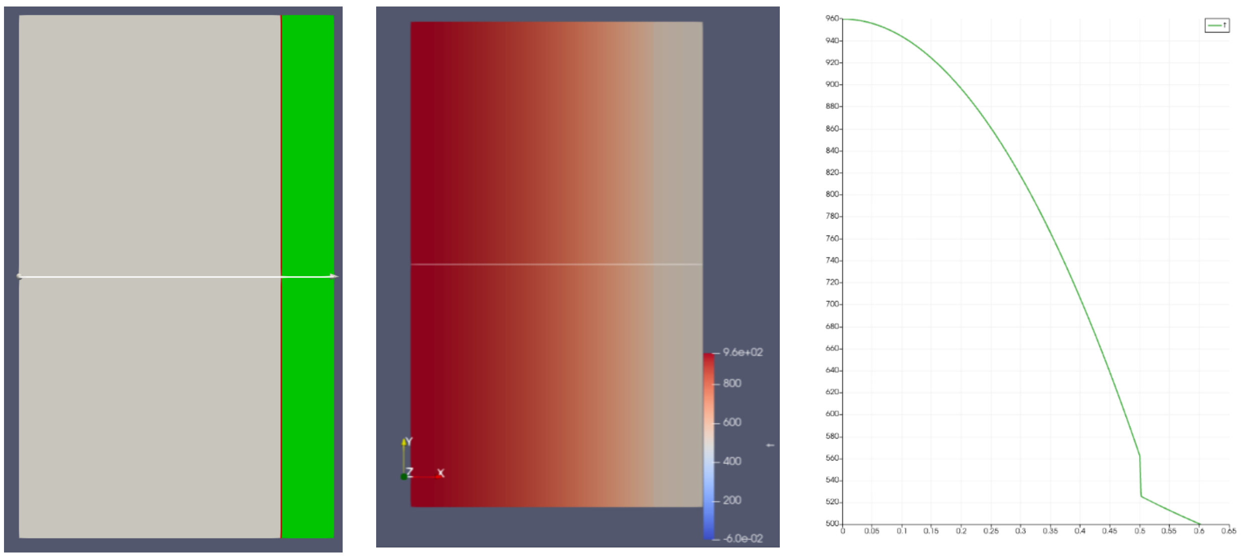

Figure 2 Meshing of the blocks and simulation of the temperature profile at constant LHR.

To compare the solution obtained to the analytical solution, we tried to use some MOOSE objects that can calculate the difference (error) between the obtained solution and the exact one. For this purpose we used `ElementL1Error` along with the exact solution for each block, however the L1 error values obtained were very high.

For the transient condition, two videos will be attached to the report showing the change of the temperature with the time up to 100 sec. Figure 3 shows the final temperature profile at time = 100 sec which has the same observations of the steady-state expect of the higher temperature especially fuel center temperature. Figure 4 shows the variation of the centerline temperature (max temp) with time. This variation with temperature proves that the fuel center temperature will follow the trend of the LHR function and will have a saturation value after enough time.

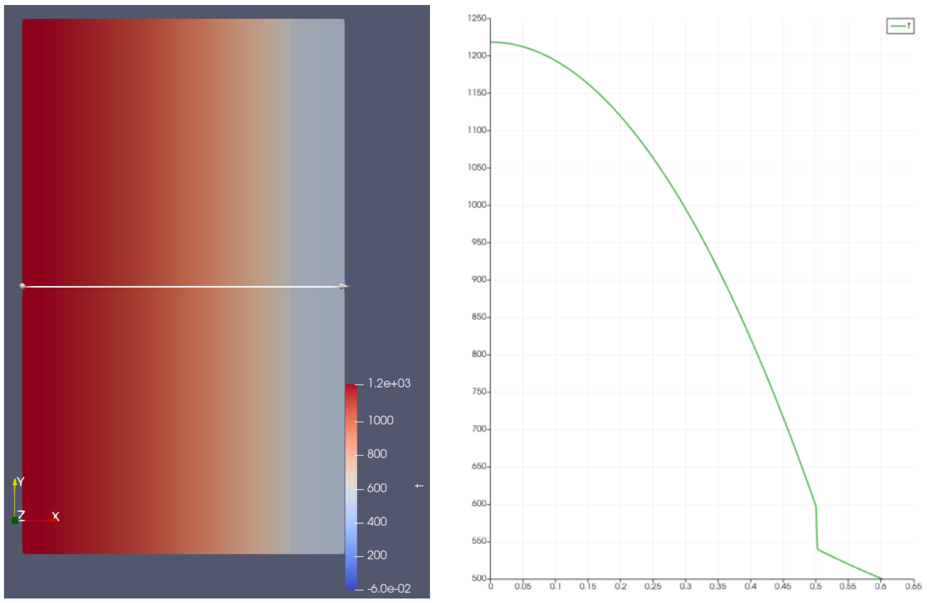

Figure 4 Temperature profile for transient LHR at time = 100 sec.

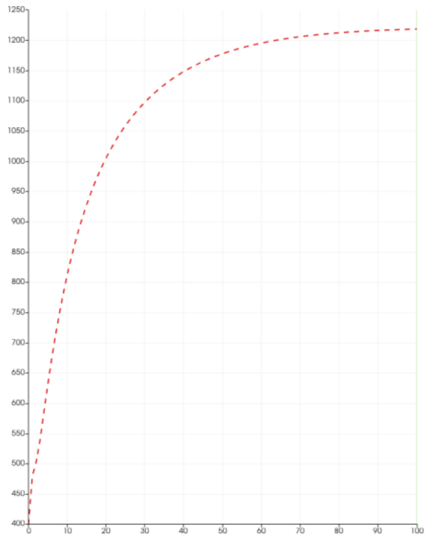

Figure 3 Variation of fuel center temperature with time for transient LHR.

3. TEMPERATURE PROFILE FOR AXIALLY LHR (Part-II)

3.1. Problem Description

For this problem, the LHR is assumed to vary axially. The dimensions of the fuel are shown in figure 5. The axial LHR is given by:

$$LHR\left(\frac{z}{Z_o}\right) = LHR^o \cos\left[\frac{\pi}{2\gamma}\left(\frac{z}{Z_o} - 1\right)\right]$$

As the LHR will vary axially, then the coolant temperature is expected to vary axially as well, which can be given by:

$$T_{cool} = \frac{1}{\gamma}\frac{Z_o \times LHR^o}{\dot{m}C_{PW}} \times \left\{\sin(\gamma) + \sin\left[\gamma\left(\frac{z}{Z_o} - 1\right)\right]\right\} + T_{cool}^{in}$$

The inlet coolant temperature is 400K and the LHR at the midplane = 150 W/cm. The problem is assumed to be 2D and we will solve it for both cases, i.e. steady-state and transient, where variation of the LHR at the midplane with time is given by:

$$LHR^o(t) = LHR^o \times \left(1 - \exp(-0.1 \times t)\right) + 50$$

The materials properties listed in table 1 is used in this case. Additional parameters related to the axial dependence of LHR is required, such as $\gamma = 1.2$, $\dot{m} = 0.25$ and $C_{PW} = 4200$.

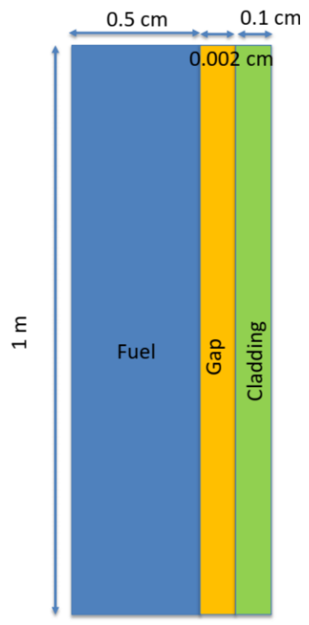

Figure 5 Dimensions of UO2 fuel for part-II

3.2. Input File

The input file for this case under the steady-state is the same as the steady-state of Part-I with some differences described below. To implement the axial variation of LHR and T$_{cool}$, we need to define them as functions, we can refer to them, as we will see.

```
[Functions]
  [vol_heat]
    type = ParsedFunction
    vars = 'rf lhro g zo'
    vals = '0.5 150 1.2 50'
    value = '(1/(pi*(rf^2))) * lhro * cos((pi/(2*g))*((y/zo) - 1))'
  []
  [cool_temp]
    type = ParsedFunction
    vars = 'g zo lhro mo cp tin'
    vals = '1.2 50 150 0.25 4200 400'
    value = '(1/g)*((zo * lhro)/(mo*cp))*(sin(g)+sin(g*(y/zo - 1))) + tin'
  []
[]
```

Instead of having a constant value of heat source we will have a function `vol_heat` defined above.

```
  [heat_source]
    type = HeatSource
    variable = T
    function = vol_heat
    block = 0
  []
```

Also, the BC of the fuel surface temperature will be modified to a function `cool_temp` defined in Functions section.

```
  [outlet]
    type = FunctionDirichletBC
    variable = T
    boundary = right
    function = cool_temp
  []
```

For the transient case, we will modify the `vol_heat` and `cool_temp` by subtituing LHR with the expression given above which describes the time dependence.

```
[Functions]
  [vol_heat]
    type = ParsedFunction
    vars = 'rf g zo'
    vals = '0.5 1.2 50'
    value = '(1/(pi*(rf^2))) * (150*(1-exp(-0.1*t))+50) * cos((pi/(2*g))*((y/zo) - 1))'
  []
  [cool_temp]
    type = ParsedFunction
    vars = 'g zo mo cp tin'
    vals = '1.2 50 0.25 4200 400'
    value = '(1/g)*((zo * ((150*(1-exp(-0.1*t))+50)))/(mo*cp))*(sin(g)+sin(g*(y/zo - 1))) + tin'
  []
[]
```

3.3. Part-II Results

The radial temperature profiles at 4 different heights, i.e. z = 0, 25, 50, and 100cm, are shown in figure 6. At elevation z = 0, which is equivalent to the steady-state case of Part-I, we can notice that the temperature profile has the same distribution. However, and as the elevation increases, the linear variation of temperature at the gap and clad starts to vanish. This can be explained if we plot the axial variation of the outer clad temperature, shown in figure 7. The maximum temperature values are observed at the mid-plane (z = 50 cm).

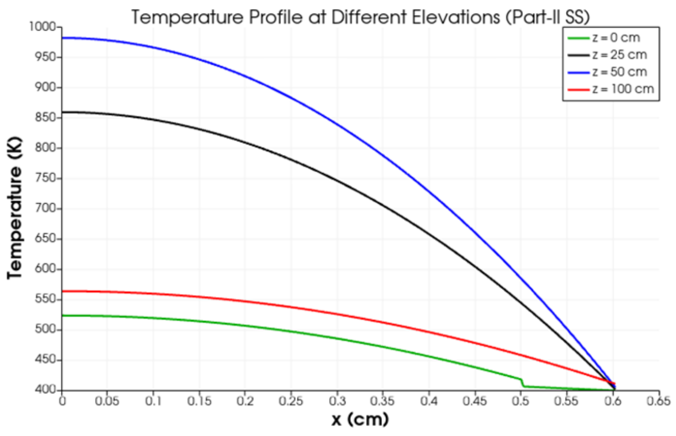

Figure 6. The radial temperature profile at different elevations

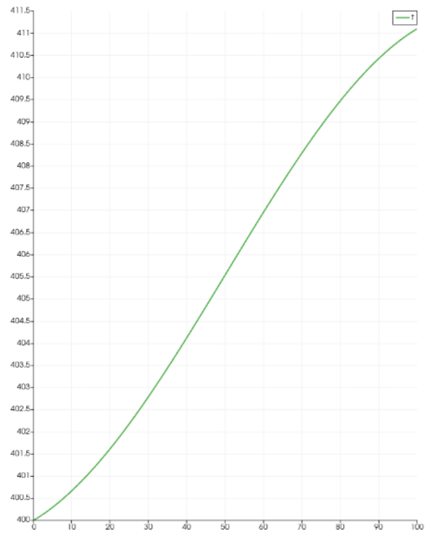

Figure 8 The axial variation of the outer cladding temperature

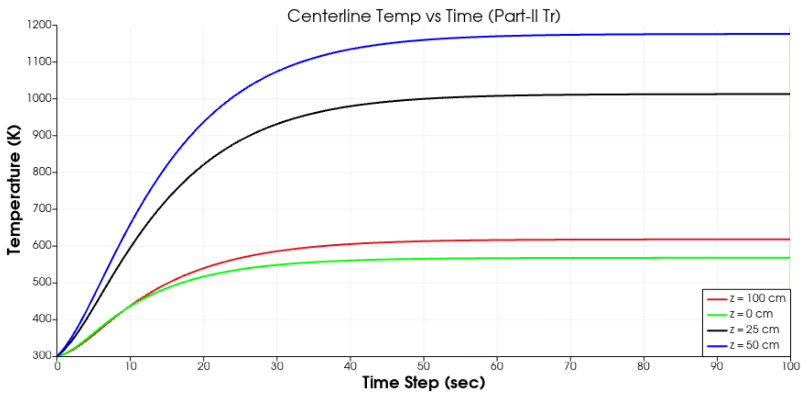

Figure 7 Centerline Temperature vs. Time at different elevations (Part-II Tr)

Figure 8 shows the variation of the centerline (0, z) temperature with time where z has different values, i.e. z = 0, 25, 50, 100 cm. Again, the highest temperatures are observed at the midplane where z = 50 cm. To determine the location of the peak centerline temperature in the steady-state and at t = 100 sec, the axial variation of the centerline temperature for both cases has been plotted in figure 9.

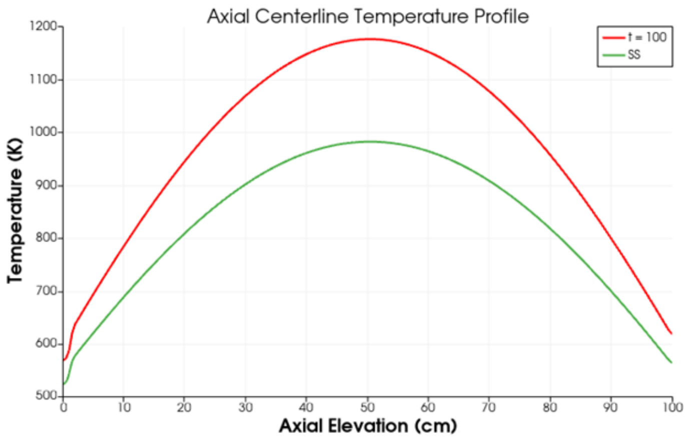

Figure 9 The axial centerline temperature profile

4. THERMAL STRESSES IN UO_2 FUEL PELLET

4.1. Problem Description

In this problem, the thermal and mechanical module are combined and employed to calculate the stresses induced due to thermal expansion for UO_2 fuel pellet. The problem is consider as 1-D, however, it's define as 2-D in the RZ coordinates. We assume that the LHR is constant and uniform and equals = 175 W/cm. For the temperature at the outer fuel surface, we assume that it's constant. We will solve this problem under 2 assumptions, the first is assuming that the thermal conductivity is constant and the second is considering the dependence of thermal conductivity on temperature for UO_2. For the thermal module, we need the same material properties used in the previous parts. However, for the mechanical module more properties are required. These properties are Young's modulus (= 200 GPa) and Poisson's Ratio (=0.345).

4.2.Input File

Most of the blocks used in this input file are already used and described before. However, there is some new blocks to better describe the problem. First we need to identify a pin node that will be used later in the BCs.

```
   [pin1]
    type = ExtraNodesetGenerator
    input = generated
    new_boundary = pin1
    coord = '0 0 0'
  []
```

For this problem, it seems that, we can solve thermomechanical problem on MOOSE at the steady state condition. Hence, we will define the blocks that have been used in the transient condition as before. For the heat source, a new value in the volumetric heat generation is calculated according to the new value of the LHR.

The following block is to automates the process of setting up multiple objects related to the solution.

```
[Modules/TensorMechanics/Master]
  [all]
    add_variables = true
    strain = FINITE
    automatic_eigenstrain_names = true
    generate_output = 'vonmises_stress'
  []
[]
```

In addition to materials related to the heat conduction, we need to identify those related to the thermal expansion and the mechanical properties of fuel. We assumed that the fuel is stress-free at the room temperature (300K).

```
   [elasticity]
    type = ComputeIsotropicElasticityTensor
    youngs_modulus = 200e5
    poissons_ratio = 0.345
  []
  [expansion1]
    type = ComputeThermalExpansionEigenstrain
    temperature = T
    thermal_expansion_coeff = 11e-6
    stress_free_temperature = 300
    eigenstrain_name = thermal_expansion
  []
  [stress]
    type = ComputeFiniteStrainElasticStress
  []
```

The BCs for this problem is a little tricky. First, we need to make sure that there is no translation or rotation for the pellet, which means that we are assuming we have a rigid body. This can be done through 2 BCs. The first is by setting the displacement in the x-direction to zero at the pin node. The second is setting the displacement in the y-direction in the bottom boundary to zero. This means that the pin node is fixed in both direction x and y as it exists in the bottom boundary. For the other BCs, we will assume that the fuel pellet is under plain strain condition, which means that the pellet is not allowed to deform in any direction. This assumption will give us more information about the thermal stresses.

```
  [pin_x]
    type = DirichletBC
    variable = disp_x
    boundary = 'pin1'
    value = 0
  []
  [disp_y]
    type = DirichletBC
    variable = disp_y
    boundary = 'bottom top right left'
    value = 0
  []
  [disp_x]
    type = DirichletBC
    variable = disp_x
    boundary = 'right left bottom top'
    value = 0
  []
```

So far we considered thermal conductivity to be constant. To make it function of temperature, we need to define function that we can refer to it.

```
[Functions]
  [kft]
    type = ParsedFunction
    vars = T
    vals = T
    value = ((6400*exp(-16.35/T))/T^2.5)+(100/(7.5408^(17.629*T) +  3.614 * T^2))
  []
[]
```

Then when we define the material of the HeatConduction we should use `thermal_conductivity_temperature_function`. However, there is something still missing here, and we couldn't do this step.

4.3. Results of Part-III

Figure 10 shows the temperature distribution over the fuel pellet at time t=24 sec. The induced stresses due to this thermal expansion is shown in figure 11.

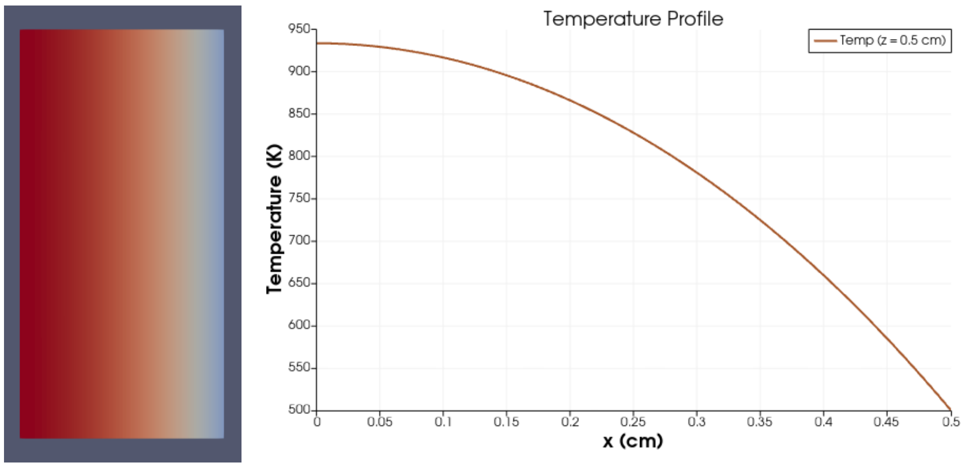

Figure 10 Temperature profile at the midplane of the fuel pellet

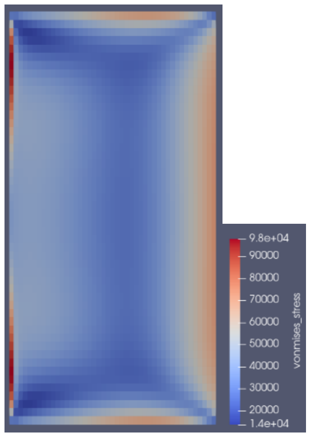

Figure 11 vonMises stress distribution due to thermal expansion

Conclusions

MOOSE framework showed a good capability to simulate the thermomechanical problems and solve them efficiently. The efficiency of the solution proposed by us could be highly improved in terms of the flexibility and the extension to other cases. However, and because of the lack of the arrangement of the MOOSE documentation, this requires more time and effort.