Rashed Almasri **NE533** **04/26/2024**

Project Part 3

Recap of Parts 1 and 2

After finishing Part 2, there was a miscalculation for LHR, as well as a mistake in the temperature of the coolant BC in the code.

The coolant temperature code was adjusted to be:

```
  [coolant_temp]
    type = ParsedFunction ## Tco = Tcool + delta(Tco-Tcool)
    expression = (1/1.2)*(350*50)/(4200*0.1)*(sin(1.2)+sin(1.2*(y/50-
1)))+500+350*cos(1.2*(y/50-1))/(2*3.14159*0.5*2.65)
  []
```

Part 3

In part 3, we don't need to include new variables for the strain and stress, as the new line for the global parameters at the beginning of the code will suffice in this instance.

Additionally, we will include a temperature dependent thermal conductivity for the fuel, as well as mechanical materials properties as follows:

For the fuel, a specific hear of 0.33 J/gK, density of 10.98 g/cm^3, modulus of elasticity of 250 GPa, a Poison's ratio of 0.32, and a linear thermal expansion coefficient of 10.471e-6 /K.

As for the cladding, a specific heat of 0.35 J/gK, density of 6.56 g/cm^3, modulus of elasticity of 99.3 GPa, a Posion's ratio of 0.37, and a linear thermal expansion coefficient of 6e-6 /K.

Also, a Gap Heat Transfer was adjusted to allow for the fuel to expand and allow contact between it and the cladding to occur. Two modules for contact and thermal contact were implemented in our code. As follows:

```
[Contact]
    [gap]
       primary = fuel_with_gap
       secondary = clad_with_gap
       model = frictionless
       formulation = mortar
    []
```

```
[]

[ThermalContact]
   [gap_contact]
     type = GapHeatTransfer
     emissivity_primary = 0.4
     emissivity_secondary = 0.4
     variable = temp
     primary = 'fuel_with_gap'
     secondary = 'clad_with_gap'
     gap_conductivity = 0.0026
     gap_geometry_type = 'CYLINDER'
     quadrature = true
   []
[]
```

As for the BCs, we took the temperature at the fuel surface instead of the cladding surface. And we added two new BCs for the strain at the bottom. Please note that BC for the radial displacement would probably be better if a node in the origin (0, 0, 0) was identified. But due to time restraints, it wasn't implemented here in time.

For the tensor mechanics, the Module/TensorMechanics/Master was written into the code. As follows:

```
[Modules/TensorMechanics/Master]
    [all]
       add_variables = true
       strain = FINITE
       eigenstrain_names = thermal
       generate_output = 'vonmises_stress'
       volumetric_locking_connection = true
       temperature = temp
    []
[]
```

Beside these main points for the thermos-mechanical coupling, additional modules were added for the stress calculations in all directions as AuxVariables and AuxKernels.

Finally, even though the project highlighted that only Steady State for this part, to solve for displacement, we must write it into a transient executioner for the code to observe the change with time.

A look at the generated Mesh can be seen in the below figure.

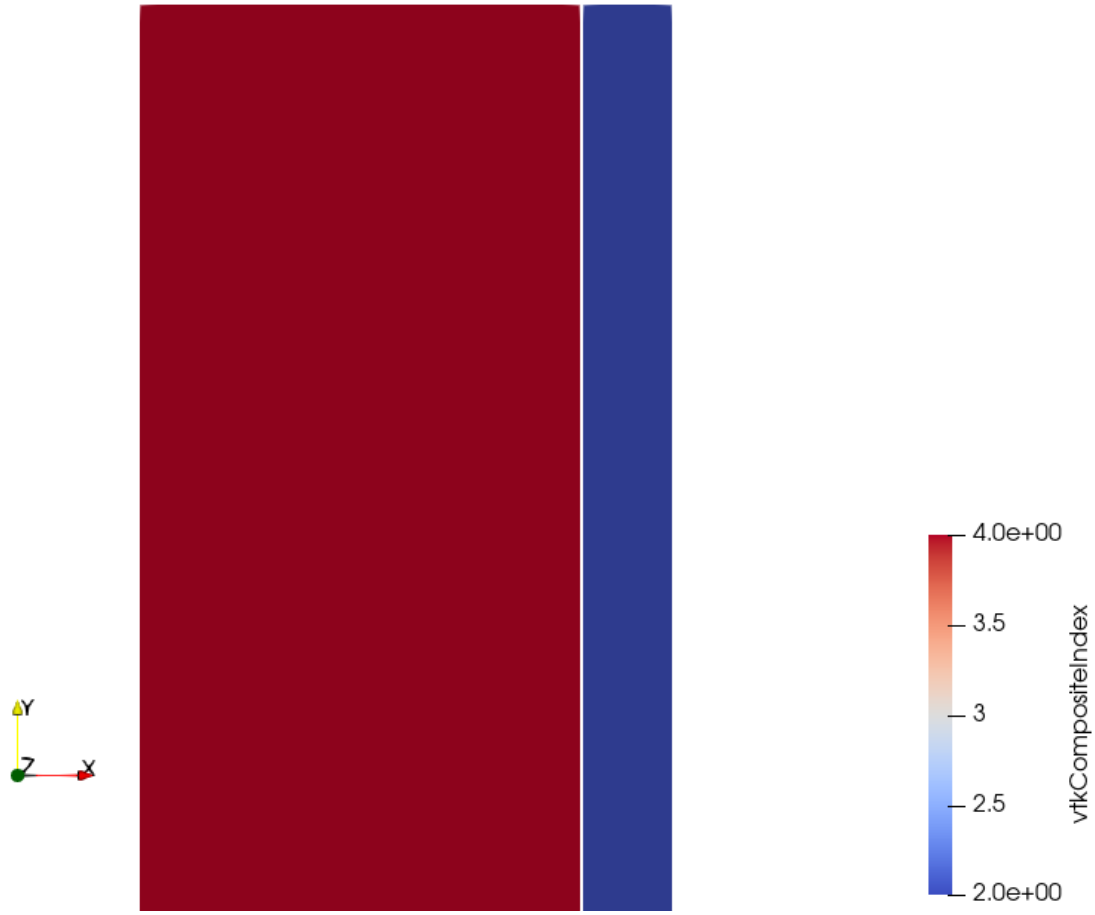

Figure 1: A snippet from the top of the mesh to highlight the gap.

Results

Now, for the results we have. Unfortunately, our code couldn't come to definite answers in time. It either diverged, or gave some errors, and at one iteration gave an output file (attached to this submission) that didn't show a real mechanical response.