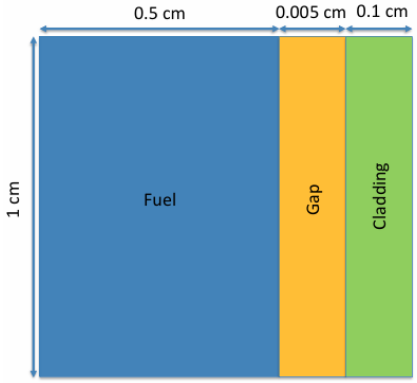
# MOOSE Project, Part 1

In this first part of the project, we were given the dimensions of a fuel system (pellet, gap, cladding) and asked to solve for the temperature profile. A diagram of the system is illustrated below.




There were 5 ways with which to solve this problem:

1. Analytical steady-state solution, constant LHR, constant material properties
2. FEM steady-state solution, constant LHR, constant material properties
3. FEM transient solution, time-dependent LHR, constant material properties
4. FEM steady-state solution, constant LHR, temperature-dependent material properties
5. FEM transient solution, time-dependent LHR, temperature-dependent LHR

All FEM solutions were performed using the MOOSE program. Input and output files for all FEM solutions are attached in this submission.

Guidance for setting up the problem was as follows:

- Assume reasonable values for material properties
- Outer cladding constant temperature: 550°C
- Constant Linear Heat Rate (LHR): 350 W/cm
- Time-dependent LHR: 350*EXP(-(t-20)^2/2) + 350 W/cm
- Determine transient solution for up to t = 100

Constant material properties were found through various resources and are given below.

| Material | Thermal Conductivity, k (W/cm*K) | Specific Heat, $C_p$ (J/g*K) | Density (g/cm$^3$) |
|---|---|---|---|
| $UO_2$ (Fuel) | 0.03 | 0.33 | 10.98 |
| He gas (Gap) | 0.00152 | 5.193 | 0.1785 |
| Zr (cladding) | 0.23 | 0.35 | 6.511 |

Properties for the fuel and cladding were found in NE 533 Lecture 3. Properties for the gap were found at Helium - Thermal Conductivity.

The only equations used to solve for temperature-dependent material properties are those given for the thermal conductivity of the fuel pellet, and the thermal conductivity of the gap (assuming the gap remains pure He gas) and are given in NE 533 Lecture 8 and Lecture 3, respectively. Those equations are presented for both below in W/cm*K.

$$k_f = \frac{100}{7.5408 + 17.629t + 3.6142t^2} + \frac{6400}{t^{\frac{5}{2}}} \exp\left[-\frac{16.35}{t}\right]$$

Where t = T / 1000, and

$$k_g = 16 \times 10^{-6} T^{0.79}$$

## Solution #1- Analytical

The following equations were used to solve the steady-state temperature profile of the fuel system from the outside of the cladding to the fuel centerline. Subscripts denote the type of material (fuel, gap, cladding), and the radius/thickness of the materials are given in the diagram. The temperatures at each boundary were calculated, and the profile from the edge to the centerline of the pellet was modeled.

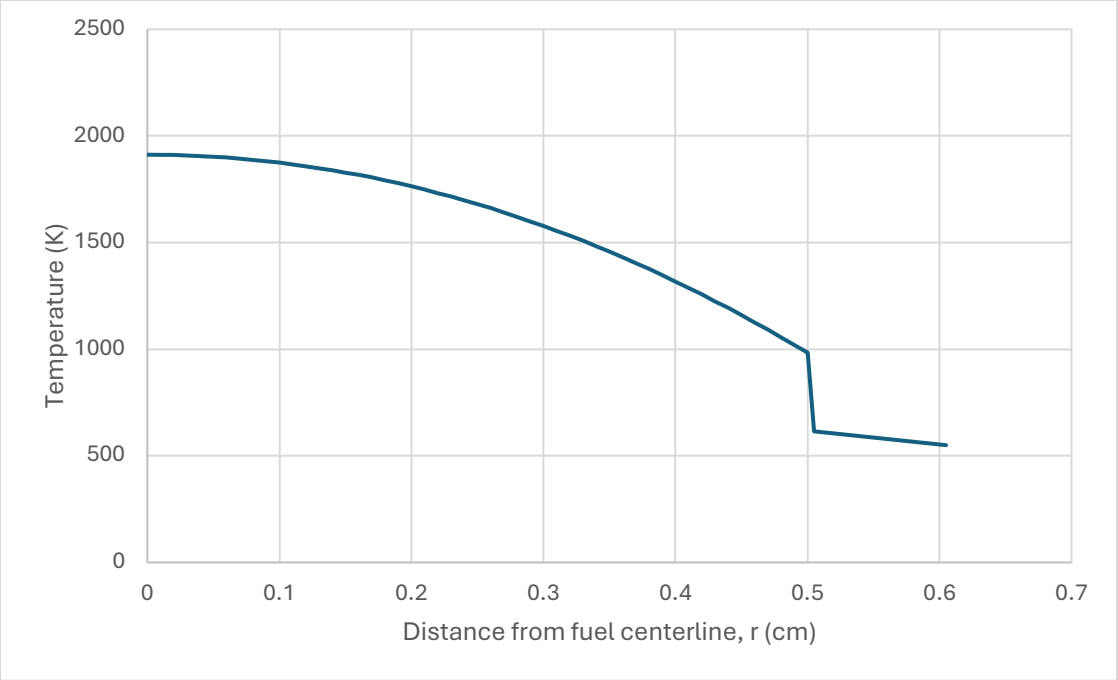
$$T_{IC} = \frac{LHR}{2\pi R_f} \times \frac{t_c}{k_c} + T_{OC}$$

$$T_s = \frac{LHR}{2\pi R_f} \times \frac{t_g}{k_g} + T_{IC}$$

$$T_o = \frac{LHR}{4\pi k_f} + T_s$$

$$T(r) = \frac{LHR}{4\pi k_f}\left(1 - \frac{r^2}{R_f^2}\right) + T_s$$

| Boundary | Location (cm) | Temperature (K) |
|---|---|---|
| Outer Cladding | 0.605 | 550 |
| Inner Cladding | 0.505 | 615.5 |
| Surface of Fuel | 0.5 | 983.7 |
| Centerline of Fuel | 0 | 1912.1 |




## Solution #2- FEM, steady-state, constant properties

There are six parts to each MOOSE input file:

- Mesh
- Variables
- Kernels
- BCs
- Executioner
- Outputs

All six of these parts will be described in this section. Since some of these parts will be repeated for the input files of other solutions, it is not necessary to cover them multiple times. This solution most closely correlates to the analytical solution, with a peak centerline temperature of about 1900K.

### Mesh

A single mesh was generated using *GeneratedMeshGenerator* that had the dimensions of the entire system outlined. Individual blocks within that mesh were defined using *SubdomainBoundingBoxGenerator*. The coordinate type was changed from the default x-y system to the axisymmetric r-z system, with symmetry around the default y-axis. Very small elements were used in creating this mesh to generate a smoother solution.

```
[Mesh]
    [gen]
        type = GeneratedMeshGenerator
        dim = 2
        nx = 1210
        ny = 100
        xmax = 0.605
        ymax = 1
    []
    [block1]
        type = SubdomainBoundingBoxGenerator
        input = gen
        block_id = 1
        block_name = 'pellet'
        bottom_left = '0 0 0'
        top_right = '0.5 1 0'
    []
    [block2]
        type = SubdomainBoundingBoxGenerator
        input = block1
        block_id = 2
        block_name = 'gap'
        bottom_left = '0.5 0 0'
        top_right = '0.505 1 0'
    []
    [block3]
        type = SubdomainBoundingBoxGenerator
        input = block2
        block_id = 3
        block_name = 'clad'
        bottom_left = '0.505 0 0'
        top_right = '0.605 1 0'
    []
    coord_type = RZ                 # Axisymmetric RZ
    rz_coord_axis = Y               # Which axis the symmetry is around
[]
```

### Variables

The variable was set as temperature, and an initial condition of 300K was set because it made sense.

```
[Variables]
    [temperature]
        initial_condition = 300 #(K)
    []
[]
```

### Kernels

The function *ADHeatConduction* was used to model the heat conduction of this closed system using automatic differentiation. The function *HeatSource* was required to add the volumetric heat generation rate, given to us in the problem as LHR. The LHR given was divided by the cross-sectional area of the pellet to give the value seeked.

```
[Kernels]
    [heat_conduction]
        type = ADHeatConduction
        variable = temperature
    []
    [heat_source]
        type = HeatSource
        variable = temperature
        #value = 445.6 #(W/cm3)
        function = '(350 / (pi*(0.5^2)))' #converting LHR=350 (W/cm) to volumetric heat rate (w/cm3)
        block = 'pellet'
    []
[]
```

### BCs

There were two boundary conditions to be used in this problem. The centerline temperature of the fuel is at a peak, thus the derivative of temperature here is 0. This is reflected in the function *NeumannBC*. The outside of the cladding is held at 550K, and is modeled using *DirichletBC*.

```
[BCs]
    [left]
        type = NeumannBC
        variable = temperature
        boundary = left
        value = 0 #(K/cm)
    []
    [right]
        type = DirichletBC
        variable = temperature
        boundary = right
        value = 550 #(K)
    []
[]
```

### Materials

Three subsets of materials were chosen and their properties (i.e. thermal conductivity) were defined using *ADGenericConstantMaterial* as is done with most materials with constant properties.

```
[Materials]
    [pellet]
        type = ADGenericConstantMaterial
        prop_names = thermal_conductivity
        prop_values = 0.03 #(W/cm*K)
        block = 'pellet'
    []
    [gap]
        type = ADGenericConstantMaterial
        prop_names = thermal_conductivity
        prop_values = 0.00152 #(W/cm*K)
        block = 'gap'
    []
    [clad]
        type = ADGenericConstantMaterial
        prop_names = thermal_conductivity
        prop_values = 0.23 #(W/cm*K)
        block = 'clad'
    []
[]
```

### Executioner

Since this was a steady-state solution, the function for the executioner was chosen as *Steady* with a Newton solve type as is common with other steady-state solutions in MOOSE. The ‘petsc’ lines were artifacts of other steady-state solutions observed online.

```
[Executioner]
    type = Steady       # Steady state problem
    solve_type = NEWTON # Perform a Newton solve, uses AD to compute Jacobian terms
    petsc_options_iname = '-pc_type -pc_hypre_type' # PETSc option pairs with values below
    petsc_options_value = 'hypre boomeramg'
[]
```

### Outputs

The output to this input file was simply an exodus file that could simulate the solution in a program like Paraview.

```
[Outputs]
    exodus = true # Output Exodus format
[]
```

## Solution #3- FEM, transient, constant properties

The parts of this solution identical to Solution #2 are: Variables, BCs, and Outputs. The max peak centerline temperature is a bit higher, about 2100K.

### Mesh

The mesh was changed to be much rougher to make the computations easier (nx=121, ny=1). Without this change, the solution does not converge to t=100.

### Kernels

A kernel was added to include the time-dependence of heat conduction, *ADHeatConductionTimeDerivative*. The time-dependent formula for LHR in *HeatSource* was written out for the pellet.

```
[Kernels]
    [heat_conduction]
        type = ADHeatConduction
        variable = temperature
    []
    [time_derivative]
        type = ADHeatConductionTimeDerivative
        variable = temperature
    []
    [heat_source]
        type = HeatSource
        variable = temperature
        #value = 445.6 #(W/cm3)
        function = '((350*exp(-((t-20)^2)/2)+350)/(pi*(0.5^2)))'
        block = 'pellet'
    []
[]
```

### Materials

Materials were still defined using *ADGenericConstantMaterial*, but additional properties (specific heat, density) were required to run the time derivative.

```
[Materials]
    [pellet]
        type = ADGenericConstantMaterial
        prop_names = 'thermal_conductivity specific_heat density'
        prop_values = '0.03 0.33 10.98' #(W/cm*K) (J/g*K) (g/cm^3)
        #prop_values = 0.03 #(W/cm*K)
        block = 'pellet'
        #all properties are from Lecture 3 for UO2
    []
    [gap]
        type = ADGenericConstantMaterial
        prop_names = 'thermal_conductivity specific_heat density'
        prop_values = '0.00152 5.193 0.1785' #(W/cm*K) (J/g*K) (g/cm^3) [use 5.193 next time]
        #prop_values = 0.00152 #(W/cm*K)
        block = 'gap'
        #all properties from periodic-table.org for He
    []
    [clad]
        type = ADGenericConstantMaterial
        prop_names = 'thermal_conductivity specific_heat density'
        prop_values = '0.23 0.35 6.511' #(W/cm*K) (J/g*K) (g/cm^3)
        #prop_values = 0.17 #(W/cm*K)
        block = 'clad'
        #all properties are from Lecture 3 for Zr
    []
[]
```

### Executioner

The type for this problem was changed from Steady to Transient. The solve type was kept at Newton since it worked before. The time step dt was chosen as 5 since the solution would not converge with a smaller time step. Thus, although the solution goes to t=100, there are only 20 time steps.

```
[Executioner]
    type = Transient
    solve_type = NEWTON
    start_time = 0.0
    end_time = 100
    #num_steps = 100
    dt = 5 #DO NOT MESS WITH THIS
    petsc_options_iname = '-pc_type -pc_hypre_type' # PETSc option pairs with values below
    petsc_options_value = 'hypre boomeramg'
[]
```

## Solution #4- FEM, steady-state, changing properties

The Mesh, Variables, BCs, Executioner, and Outputs are identical to Solution #2. However, the max peak centerline temperature for this solution was only around 810K.

### Kernels

The only change is that the heat conduction function was changed from *ADHeatConduction* to just *HeatConduction*, since no AD materials were used.

### Materials

The materials were changed from *ADConstantGenericMaterial to ParsedMaterial* for the fuel and gap, and *GenericConstantMaterial* for the cladding. *ParsedMaterial* was used for materials whose thermal conductivity changed with temperature, and the cladding was changed to reflect a non-AD heat conduction kernel.

```
[Materials]
    [pellet]
        type = ParsedMaterial
        property_name = thermal_conductivity
        coupled_variables = T
        expression = '(100 / (7.5408 + (17.629*(T/1000)) + (3.6142*((T/1000)^2)))) + (((6400 / ((T/1000)^(5/2)))*exp(-16.35 / (T/1000)))' #W/cm*K
        block = 'pellet'
    []
    [gap]
        type = ParsedMaterial
        property_name = thermal_conductivity
        coupled_variables = T
        expression = '(16*10^(-6))*(T^0.79)' #W/cm*K
        block = 'gap'
    []
    [clad]
        type = GenericConstantMaterial
        prop_names = thermal_conductivity
        prop_values = 0.17 #(W/cm*K)
        block = 'clad'
    []
[]
```

## Solution #5- FEM, transient, changing properties

The Mesh is rough like the other transient solution. The Variables, BCs, and Outputs are similar to Solution #2. This solution ultimately did not converge at t=100, but there is data up to t=33 with a max peak centerline temperature of 920K, similar to Solution #4.

### Kernels

The same functions were used as in Solution #3 except they were converted from AD-functions to their non-AD form.

### Materials

*ParsedMaterial* and *GenericConstantMaterial* were used as in Solution #4. However, both functions would have to be used for the fuel and the gap since they had both constant properties and changing properties that were required for the time derivative function.

```
[Materials]
    [pellet_k]
        type = ParsedMaterial
        property_name = thermal_conductivity
        coupled_variables = T
        expression = '(100 / (7.5408 + (17.629*(T/1000)) + (3.6142*((T/1000)^2)))) + (((6400 / ((T/1000)^(5/2)))*exp(-16.35 / (T/1000)))' #W/cm*K
        #prop_names = 'thermal_conductivity specific_heat density'
        #prop_values = '0.03 0.33 10.98' #(W/cm*K) (J/g*K) (g/cm^3)
        #prop_values = 0.03 #(W/cm*K)
        block = 'pellet'
        #all properties are from Lecture 3 for UO2
    []
    [pellet_prop]
        type = GenericConstantMaterial
        prop_names = 'specific_heat density'
        prop_values = '0.33 10.98' #(J/g*K) (g/cm^3)
        block = 'pellet'
    []
    [gap_prop]
        type = GenericConstantMaterial
        prop_names = 'specific_heat density'
        prop_values = '5.193 0.1785' #(J/g*K) (g/cm^3)
        block = 'gap'
    []
    [gap_k]
        type = ParsedMaterial
        property_name = thermal_conductivity
        coupled_variables = T
        expression = '(16*10^(-6))*(T^0.79)' #W/cm*K
        #prop_names = 'thermal_conductivity specific_heat density'
        #prop_values = '0.00152 5.193 0.1785' #(W/cm*K) (J/g*K) (g/cm^3)
        #prop_values = 0.00152 #(W/cm*K)
        block = 'gap'
        #all properties from periodic-table.org for He
    []
```

**Executioner**

This was set up similar to Solution #3 except a time step of dt=1 was used. The solution would not converge no matter what the time step size, it always stopped at t=33. Therefore, it was ultimately decided to reduce the time step to its minimum (dt=1) and utilize whatever output file was generated. The non-convergence of this solution is likely due to the material properties chosen, with the temperature-dependence adding another layer of complex calculations to the transient problem. The constant material properties of helium gas may also be a factor, as using lower numbers solved the non-convergence issue on some occasions. However, these changes were artificial and did not reflect the nature of the system. The PJFNK solve type did not mitigate these issues.

```
[Executioner]
    type = Transient
    solve_type = NEWTON
    start_time = 0.0
    end_time = 100
    dt = 1 #DO NOT MESS WITH THIS
    petsc_options_iname = '-pc_type -pc_hypre_type' # PETSc option pairs with values below
    petsc_options_value = 'hypre boomeramg'
[]
```