# Computer Project - PWR and BWR Modeling

Arnav Goyal

10 December 2025

# Contents

# List of Figures

## List of Tables

## 1 Introduction

Rather than neutronics being the limiting factor for the amount of power we can produce, heat transfer limits due to material properties cap the amount of power that can be removed safely from nuclear reactors. Fluid selection, therefore, is a critical part of light water nuclear systems, both for its role in moderating neutrons to ensure the continuation of the fission chain reaction and for its role as a coolant and heat transfer fluid. In this report, we aim to model fluid flow in two channels from separate nuclear reactors using numerical analysis. This method is inherently finite so it will lack complete spatial clarity. The goal is to derive and solve the heat transfer equations and conservation equations to solve for an accurate model of the fluid in the two channels and within the fuel rods.

## 2 Problem Statement

Table 1: Starting conditions for both PWR and BWR scenarios [1]

| Property | PWR | BWR |
|---|---|---|
| H | 4 m | 4.1 m |
| $H_e$ | 4.3 m | 4.4 m |
| $D_{rod}$ | 0.95 cm | 1.227 cm |
| Pitch | 1.26 cm | 1.62 cm |
| $D_{fuel}$ | 0.82 cm | 1.04 cm |
| Gap Thickness | 0.006 cm | 0.010 cm |
| $k_{gap}$ | $0.25\ \frac{W}{m°C}$ | |
| $k_{fuel}$ | $3.6\ \frac{W}{m°C}$ | |
| $k_{cladding}$ | $21.5\ \frac{W}{m°C}$ | |
| G | $4000\ \frac{kg}{m^2 s}$ | $2350\ \frac{kg}{m^2 s}$ |
| $q_0'$ | $380\ \frac{W}{cm}$ | $605\ \frac{W}{cm}$ |
| $P(z=-H/2)$ | 15 MPa | 7.5 MPa |
| $T_f(z=-H/2)$ | $277°C$ | $272°C$ |

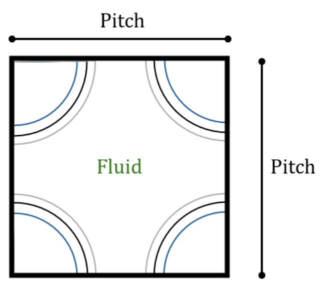

Fig. 1: Channel geometry

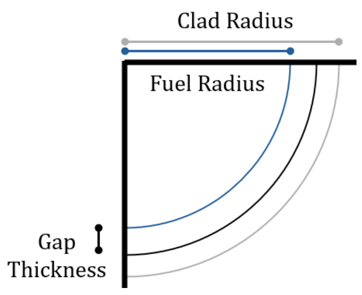

Fig. 2: Rod geometry

The generic setup for both nuclear systems is the same; water is flowing upward through a vertical channel with fuel rods at each corner, heating the fluid. Each modeled channel is given different starting conditions, and different parts have variable sizes. The starting conditions and the generic shape of the channel are given in Table 1, and Figs. 1 and 2. The linear heat generation for the channel is given by Equation 1.

$$q'(z) = q_0' \cos(\pi z/H_e), -H/2 \leq z \leq H/2 \tag{1}$$

This can be converted to other measured fluxes for the rod surfaces ($q''$), or heat generation within the fuel pellet ($q'''$) using the relations in Equation 2.

$$q[W] = q'[\frac{W}{m}] \cdot \Delta z[m] = q''[\frac{W}{m^2}] \cdot (2\pi r_{fuel} \cdot \Delta z)[m^2] = q'''[\frac{W}{m^3}] \cdot (\pi r_{fuel}^2 \cdot \Delta z)[m^3] \tag{2}$$

For both setups, some crucial assumptions have been made. These include steady-state operation, no internal heat generation in the fluid, negligible axial conduction, constant single-phase properties, and negligible changes in fuel and clad properties with temperature. The models developed by solving for the properties of the fluid and the temperature within the rod will help us understand more about how the reactor behaves during operation.

## 3 Methods

### 3.1 Fluid Flow Modeling

The Homogeneous Equivalent Model (HEM) assumes that the speed of the gas ($v_g$) and fluid ($v_f$) in a mixed flow system are equal, $P_f = P_g = P_{sat}$, and $T_f = T_g = T_{sat}$. This allows us to treat the entire fluid as a mixed fluid instead of having to solve for different percentages and properties when in $2\phi$ flow. When there is single-phase ($1\phi$) flow, the equations we will derive can be applied as usual, though the pressure and temperature will not be restricted to just the saturated values.

There are three conservation equations that are derived as part of the HEM and can be applied to calculate the necessary values of the fluid.

Equation 3 is the mass conservation equation for the fluid; given our assumption of steady-state behavior, it is clear that G, the momentum of the fluid or $\rho_m v$, is constant.

$$\cancel{\frac{d\rho_m}{dt}} + \frac{d\rho_m v}{dz} = 0 \tag{3}$$

$\rho_m$ is the mixed density of the fluid and can be calculated with the quality and the specific volume as in Equation 4, or with the void fraction as in Equation 5.

$$\rho_m = (v_g \chi + (1-\chi)v_f)^{-1} \tag{4}$$

$$\rho_m = \rho_g \alpha + (1-\alpha)\rho_f \tag{5}$$

The next conservation equation is for momentum, and it is used to calculate the pressure drop along the channel. It is given in Equation 6.

$$\frac{d\rho_m v}{dt} + \frac{d\rho_m vv}{dz} = -\frac{dP}{dz} - \frac{\tau_f \xi_w}{A_f} - \rho g \sin(\theta) \tag{6}$$

Since this is a vertical channel with water flowing upward, $\theta = 90°$, this simplification and using our definition of G and prior assumptions, the new conservation equation is given in Equation 7. The definition of $\tau_f$ is also used, where $f$ is the friction factor.

$$-\frac{dP}{dz} = G^2 \frac{d}{dz}\frac{1}{\rho_m} + \frac{\xi_w}{A_f}\frac{1}{2} f \frac{G^2}{\rho_m} + \rho_m g \tag{7}$$

While the equilibrium quality, $\chi_e$, is less than 0, the fluid is $1\phi$ flow, so $\rho_m$ is constant, and $f = f_{1\phi} = 0.316 \cdot Re^{-0.25}$. These corrections simplify Equation 7 into Equation 8

$$-\frac{dP}{dz} = 0.316 \cdot Re^{-0.25} \frac{\xi_w}{A_f}\frac{G^2}{2\rho_i} + \rho_i g \tag{8}$$

Once the fluid is operating in a $2\phi$ flow, the temperature of the fluid has reached $T_{sat}$, the pressure is no longer constant, and the friction factor is given by Equation 9. The acceleration term of the momentum conservation equation can be simplified to Equation 12. The expanded pressure drop equation can then be given by Equation 13

$$f_{2\phi} = 0.046 \cdot Re^{-0.2} \cdot (\frac{\mu_f}{\mu_m})^{-0.2} \text{ where } \mu_m = (\frac{\chi_e}{\mu_g} + \frac{1-\chi_e}{\mu_f})^{-1} \tag{9}$$

$$G^2 \frac{d}{dz}\frac{1}{\rho_m} = G^2 \frac{d}{dz}(v_{fg}\chi + v_f) \tag{10}$$

$$G^2(\frac{d}{dz}v_{fg}\chi + \frac{dv_f}{dz}) = G^2(v_{fg}\frac{d\chi}{dz} + \chi\cancel{\frac{dv_{fg}}{dz}} + \cancel{\frac{dv_f}{dz}}) \tag{11}$$

$$G^2 \frac{d}{dz}\frac{1}{\rho_m} = G^2 v_{fg}\frac{d\chi}{dz} \tag{12}$$

$$-\frac{dP}{dz} = G^2 v_{fg}\frac{d\chi}{dz} + \frac{1}{2} f_{2\phi} \cdot G^2 \cdot (v_g\chi + (1-\chi)v_f)\frac{\xi_w}{A_f} + \frac{g}{v_{fg}\chi + v_f} \tag{13}$$

The final step for the fluid is the energy conservation equation, which is given in Equation 14.

$$\frac{d}{dt}\rho_m h + \frac{d}{dz}\rho_m v h = \frac{q''\xi_w}{A_f} + \frac{dP}{dt} + q''' \tag{14}$$

Again, the assumptions, steady state and no internal heat generation in the fluid, and our definition of G can be applied to yield a simplified Equation 15

$$G\frac{dh}{dz} = \frac{q''\xi_w}{A_f} \tag{15}$$

Given that $h = \chi_e h_{fg} + h_f$, Equation 17 can be plugged into Equation 15 to give Equation 19.

$$\frac{dh}{dz} = \frac{d}{dz}(\chi_e h_{fg} + h_f) \tag{16}$$

$$\frac{dh}{dz} = \chi_e\frac{dh_{fg}}{dz} + h_{fg}\frac{d\chi_e}{dz} + \frac{dh_f}{dz} \tag{17}$$

$$h_{fg}\frac{d\chi_e}{dz} + \frac{dh_f}{dz} + \chi_e\frac{dh_{fg}}{dz} = \frac{q''\xi_w}{A_f G} \tag{18}$$

$$\frac{d\chi_e}{dz} = \frac{1}{h_{fg}}(\frac{q''\xi_w}{A_f G} - \frac{dh_f}{dz} - \chi_e\frac{dh_{fg}}{dz}) \tag{19}$$

With a new value for $\chi_e$ calculated, the new enthalpy, $h$, and the corresponding temperature can also be solved for.

## 3.2 Heat transfer within the rod

Region 1 is for $r < r_{fuel}$ corresponding to the fuel pellet where heat is being generated. Region 2 corresponds to the gap between the pellet and clad and Region 3 corresponds to the cladding. The dimensions for these different regions are specified in the code, but the geometry follows Fig. 2. Within the first region, the fuel, there is heat generation. The constant properties and cylindrical geometry allows for a clearer equation to be solved for, which is given in Equation 25.

$$\nabla k \nabla T_1 = -q''' \tag{20}$$

$$\frac{1}{r}\frac{d}{dr}r\frac{dT_1}{dr} = -\frac{q'(z)}{\pi r_{fuel}^2 \cdot k_{fuel}} \tag{21}$$

$$\frac{d}{dr}r\frac{dT_1}{dr} = -\frac{q'(z)}{\pi r_{fuel}^2 \cdot k_{fuel}}r \tag{22}$$

$$r\frac{dT_1}{dr} = -\frac{q'(z)}{\pi r_{fuel}^2 \cdot k_{fuel}}\frac{r^2}{2} + C_1 \tag{23}$$

$$\frac{dT_1}{dr} = -\frac{q'(z)}{\pi r_{fuel}^2 \cdot k_{fuel}}\frac{r}{2} + \frac{C_1}{r} \tag{24}$$

$$T_1(r) = -\frac{q'(z)}{\pi r_{fuel}^2 \cdot k_{fuel}}\frac{r^2}{4} + C_1 \cdot \ln(r) + C_2 \tag{25}$$

The heat flux from the fuel is then transferred into the gap, and the clad, Regions 2 and 3, both of which have the same generic form once solved, as is clear in Equations 29 and 33.

$$\nabla k \nabla T_2 = 0 \tag{26}$$

$$\frac{1}{r}\frac{d}{dr} r \frac{dT_2}{dr} = 0 \tag{27}$$

$$\frac{dT_2}{dr} = \frac{C_3}{r} \tag{28}$$

$$T_2(r) = C_3 \ln(r) + C_4 \tag{29}$$

$$\nabla k \nabla T_3 = 0 \tag{30}$$

$$\frac{1}{r}\frac{d}{dr} r \frac{dT_3}{dr} = 0 \tag{31}$$

$$\frac{dT_3}{dr} = \frac{C_5}{r} \tag{32}$$

$$T_3(r) = C_5 \ln(r) + C_6 \tag{33}$$

These three general solutions, Equations 25, 29, and 33, can then be solved through the use of boundary conditions. The first boundary condition is symmetry, given in Equation 34.

$$\frac{dT_1}{dr}|_{r=0} = 0 \text{ So, } C_1 = 0 \tag{34}$$

The next two boundary conditions are for heat flux continuity at the two boundaries within the cladding, these are given by Equations 35, and 38.

$$-k_{fuel}\frac{dT_1}{dr}|_{r=r_{fuel}} = -k_{gap}\frac{dT_2}{dr}|_{r=r_{fuel}} \tag{35}$$

$$-k_{fuel}(-\frac{q'(z)}{\pi r_{fuel}^2 \cdot k_{fuel}}\frac{r_{fuel}}{2}) = -k_{gap}\frac{C_3}{r_{fuel}} \tag{36}$$

$$C_3 = (\frac{-q'(z)}{2\pi \cdot k_{gap}}) \tag{37}$$

$$-k_{gap}\frac{dT_2}{dr}|_{r=r_{c,i}} = -k_{clad}\frac{dT_3}{dr}|_{r=r_{c,i}} \tag{38}$$

$$-k_{gap}\frac{C_3}{r_{c,i}} = -k_{clad}\frac{C_5}{r_{c,i}} \tag{39}$$

$$C_5 = \frac{k_{gap}C_3}{k_{clad}} \tag{40}$$

The fourth boundary condition is for the heat transfer to the fluid. This process can be either single phase or two-phase heat transfer, the switch between them occurs when the temperature of wall crosses the saturation temperature of the fluid. When the temperature is lower than the saturation temperature, the heat transfer is one phase can be given by Equation 44 which utilizes the Dittus-Boelter correlation [2]. The fourth boundary condition can then be calculated as given in Equation 41. When

$$-k_{clad}\frac{dT_3}{dr}|_{r=r_{c,o}} = h_{1\phi}(T_3(r_{c,o}) - T_f) \tag{41}$$

$$-k_{clad}\frac{C_5}{r_{c,0}} = h_{1\phi}(C_5\ln(r_{c,o}) + C_6 - T_f) \tag{42}$$

$$C_6 = C_5(\frac{-k_{clad}}{h_{1\phi}\cdot r_{c,0}} - \ln(r_{c,o})) + T_f \tag{43}$$

$$h_{1\phi} = 0.023\cdot Re^{0.8}Pr^{0.4}\frac{k}{D_{equiv}} \tag{44}$$

If the wall temperature is larger than the saturation temperature, the Liu-Winterton correlation needs to be used instead. It is given as Equation 45 with all of the definitions for its necessary parameters [2].

$$(q'')^2 = (Fh_{1\phi}(T_w - T_f))^2 + (Sh_{NB}(T_w - T_{sat}))^2 \tag{45}$$

$$F = (1 + \chi\cdot \mathrm{Pr}\cdot(\frac{\rho_f}{\rho_g} - 1))^{0.35} \tag{46}$$

$$S = (1 + 0.055F^{0.1}Re_L^{0.16})^{-1} \tag{47}$$

$$h_{NB} = 55(\frac{P}{P_c})^{0.12}(q'')^{2/3}\cdot(-\log_{10}(\frac{P}{P_c}))^{-0.55}\cdot MM_w^{-0.5} \tag{48}$$

This correlation can then be used to solve for the fourth boundary condition as is clear in Equation 49.

$$-k_{clad}\frac{dT_3}{dr}|_{r=r_{c,o}} = q'' = \sqrt{(F \cdot h_{1\phi} \cdot (T_3(r_{c,o}) - T_f))^2 + (S \cdot h_{NB} \cdot (T_3(r_{c,o}) - T_{sat}))^2} \tag{49}$$

$$k_{clad}^2 \frac{C_5^2}{r_{c,0}^2} = F^2 \cdot h_{1\phi}^2 \cdot (T_3(r_{c,o}) - T_f)^2 + S^2 \cdot h_{NB}^2 \cdot (T_3(r_{c,o}) - T_{sat})^2 \tag{50}$$

$$A_1 = F^2 \cdot h_{1\phi}^2, A_2 = S^2 \cdot h_{NB}^2 \tag{51}$$

$$k_{clad}^2 \frac{C_5^2}{r_{c,0}^2} = A_1 \cdot (T_3(r_{c,o})^2 - 2T_f T_3(r_{c,o}) + T_f^2) + A_2 \cdot (T_3(r_{c,o})^2 - 2T_{sat} T_3(r_{c,o}) + T_{sat}^2) \tag{52}$$

$$0 = (A_1 + A_2) \cdot T_3(r_{c,o})^2 - 2(A_1 \cdot T_f + A_2 \cdot T_{sat}) T_3(r_{c,o}) + A_1 T_f^2 + A_2 T_{sat}^2 - k_{clad}^2 \frac{C_5^2}{r_{c,0}^2} \tag{53}$$

$$L_1 = (A_1 + A_2), L_2 = -2(A_1 \cdot T_f + A_2 \cdot T_{sat}), L_3 = A_1 T_f^2 + A_2 T_{sat}^2 - k_{clad}^2 \frac{C_5^2}{r_{c,0}^2} \tag{54}$$

$$L_1 \cdot T_3(r_{c,0})^2 + L_2 T_3(r_{c,0}) + L_3 = 0 \tag{55}$$

$$T_3(r_{c,0}) = \frac{-L_2 \pm \sqrt{L_2^2 - 4(L_1)(L_3)}}{2L_1} \tag{56}$$

$$C_6 = T_3(r_{c,0}) - C_5 \ln r_{c,0} \tag{57}$$

The final two boundary conditions are temperature boundary conditions and are given in Equations 58, and 61.

$$T_2(r_{c,i}) = T_3(r_{c,i}) \tag{58}$$

$$C_3 \ln(r_{c,i}) + C_4 = C_5 \ln(r_{c,i}) + C_6 \tag{59}$$

$$C_3 \ln(r_{c,i})(\frac{k_{gap}}{k_{clad}} - 1) + C_6 = C_4 \tag{60}$$

$$T_1(r_{fuel}) = T_2(r_{fuel}) \tag{61}$$

$$-\frac{q'(z)}{\pi r_{fuel}^2 \cdot k_{fuel}} \frac{r_{fuel}^2}{4} + C_2 = C_3 \ln(r_{fuel}) + C_4 \tag{62}$$

$$C_2 = C_3 \ln(r_{fuel}) + C_4 + \frac{q'(z)}{4\pi \cdot k_{fuel}} \tag{63}$$

These 6 boundary conditions also help us solve for the 6 unknowns allowing us to calculate the temperature within the rod at any radial value. The difference between single and two phase heat transfer is clear in the computational weight and must always be checked for, but we also need to check and make sure we haven't reached the critical heat flux. This is a value associated with either Departure from Nucleate Boiling (DNB) or dryout, two separate phenomenon where the lack of adequate heat transfer to the fluid leads to a spike in the temperature at the wall of the channel. DNB occurs when a thin film of vapor forms on the walls of the channel as the bubbles generated are not replaced by fluid fast enough and occurs when the heat flux is much higher. Dryout occurs when the heat flux has been applied long enough to the fluid that all of the fluid has been converted to a vapor and now there is no effective heat transfer. Regardless of the method, the important point is that when designing these channels it is important to not reach the critical heat flux. This critical heat flux correlation can be calculated using the W-3 correlation, which is given in Equation 64 [2].

$$\begin{aligned} q''_{crit} = {} & [(2.022 - 0.06238\frac{P}{10^6}) + (0.1722 - 0.01427\frac{P}{10^6}) \cdot \exp{(18.177 - 0.5987\frac{P}{10^6})\chi_e}] \\ & \cdot[(0.1484 - 1.596\chi_e + 0.1729\chi_e|\chi_e|)2.326G + 3271] \\ & \cdot[1.157 - 0.869\chi_e] \cdot [(0.2664 + 0.8357\exp(-124.1D_h))(0.8258 + 0.0003414(\frac{h_f}{1000} - \frac{h_{in}}{1000}))] \end{aligned} \tag{64}$$

The departure from nucleate boiling ratio can then be used to understand how close one is to the critical heat flux with the current heat flux based on Equation 65.

$$DNBR = \frac{q''_{crit}}{q''(z)} \tag{65}$$

A measurement in a similar vein is the critical quality, $\chi_{crit}$. It can be solved for using Equation 66, with the necessary values solved for below it. $L_{cr}$ is the distance traveled from the inlet in meters. This is the CISE-4 correlation and exclusively applies to BWRs [2].

$$\chi_{crit} = (a\frac{L_{cr}}{L_{cr} + b}) \tag{66}$$

$$G^* = 3375(1 - \frac{P}{P_c})^3 \tag{67}$$

$$a = \begin{cases} \frac{1 - \frac{P}{P_c}}{(G/1000)^{1/3}} & \text{if } G \geq G^* \\ (1 + 1.481 \cdot 10^{-4} \cdot G(1 - \frac{P}{P_c})^{-3})^{-1} & \text{if } G < G^* \end{cases} \tag{68}$$

$$b = 0.199(\frac{P_c}{P} - 1)^{0.4} \cdot GD^{1.4} \tag{69}$$

## 3.3 Solving it numerically

The combination of all of these equations yields a complicated transcendental equation that cannot be solved to give an analytic solution. As such, the models for this code are based on the use of finite difference approximations to calculate the change in fluid properties and temperatures along the channel height. Equation 70 can be used with Equations 8 and 13 to solve for the pressure along the channel numerically, while Equation 71 can be used to solve for the temperatures and other fluid properties along the channel.

$$\frac{dP}{dz} = \frac{P^{i+1} - P^{i}}{\Delta z} \tag{70}$$

$$\chi_e^i = \frac{1}{h_{fg}^{i-1}}\left(\frac{q''\xi_w \Delta z}{A_f G} - (h_f^i - h_f^{i-1}) - \chi_e^{i-1}(h_{fg}^i - h_{fg}^{i-1})\right) + \chi_e^{i-1} \tag{71}$$

# 4 Results and Analysis

## 4.1 Pressurized Water Reactor (PWR)

The geometric information and the initial properties for the fluid are all given in Table 1. The size of the numerical steps to accurately model the entire channel was selected through a mesh convergence, as can be seen in Fig. 3.

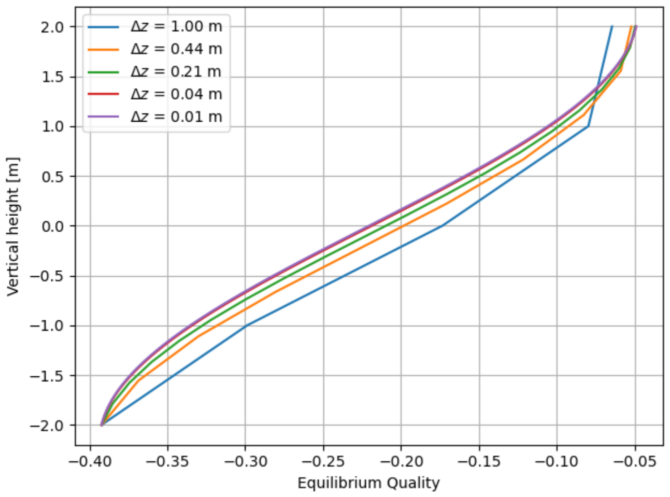

Fig. 3: $\chi_e$ convergence based on $\Delta z$ where smaller step sizes are tested to find a good balance between computational intensity and accuracy

Once the step size was selected, all of the necessary properties could be plotted with the height of the channel, $z$ on the y-axis. Fig. 4 demonstrates the behavior of the fluid, coolant, temperature and the wall temperature in relation to the saturation temperature. Fig. 5 demonstrates the behavior of the fuel centerline, fuel surface, clad inner surface and clad outer surface temperatures along the channel. Fig. 6 gives the pressure along the channel, while Fig. 7 gives the departure from nucleate boiling ratio.

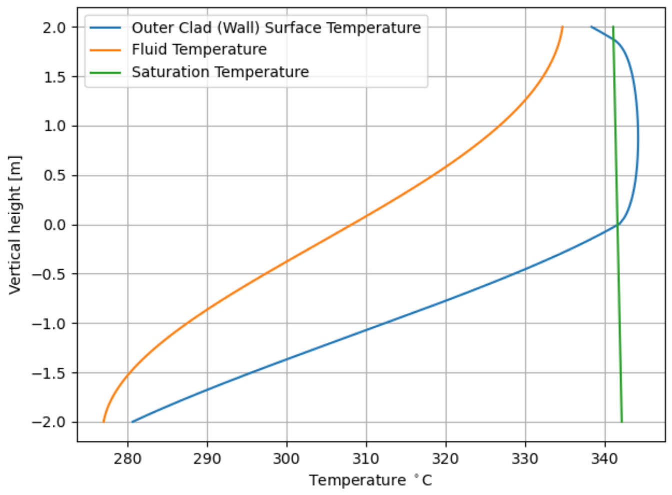

Fig. 4: Temperatures of fluid and wall with saturation temperature for comparison

The temperature for the rods is the highest at the centerline, and as the radius increases, the temperature measured decreases as well. The maximum temperature for the fuel is about 1580 °C and happens halfway through the channel. The peak fluid temperature occurs at its exit, where it is around 333 °C. The clad outer surface peaks at 345 °C close to the exit, while the inner surface peaks closer to the center at around 360°C. The radial temperature profile at $z = H/2, H/4$, and $z_{max,CL}$ can be given by Fig. 8, 9, and 10.

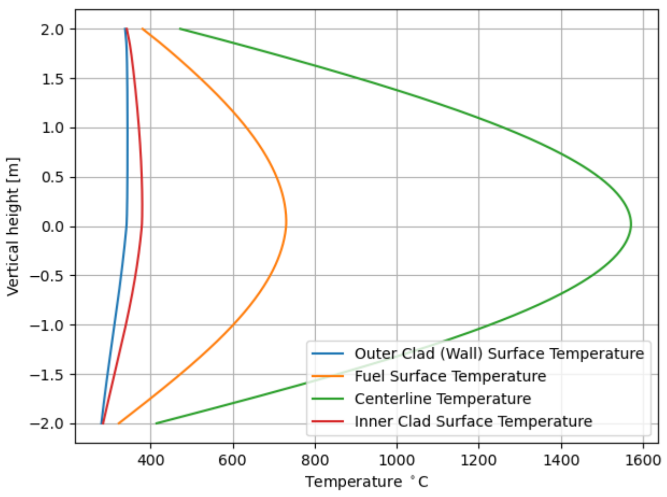

Fig. 5: Temperature at fuel centerline, fuel surface, clad inner surface and clad outer surface temperature

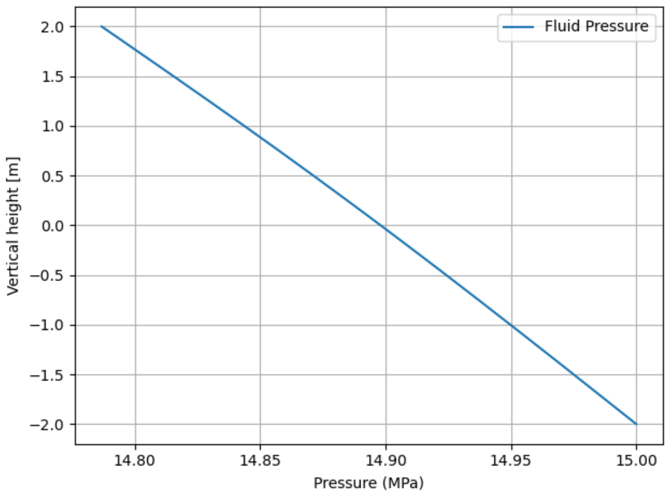

Fig. 6: Pressure of the water along the channel

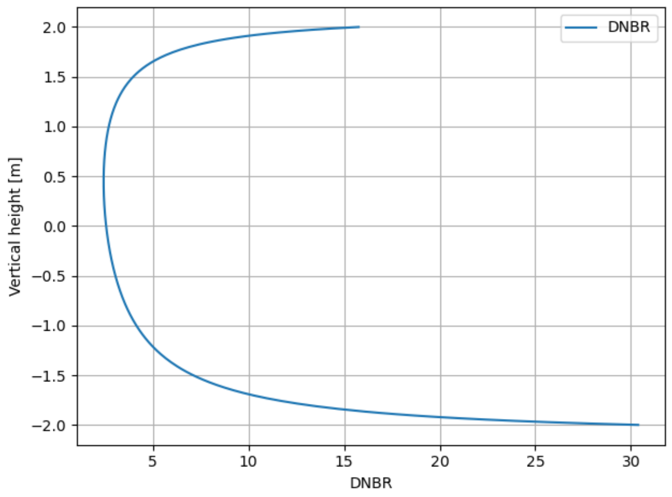

Fig. 7: Departure from nucleate boiling ratio, calculated by comparing the critical heat flux and actual heat flux

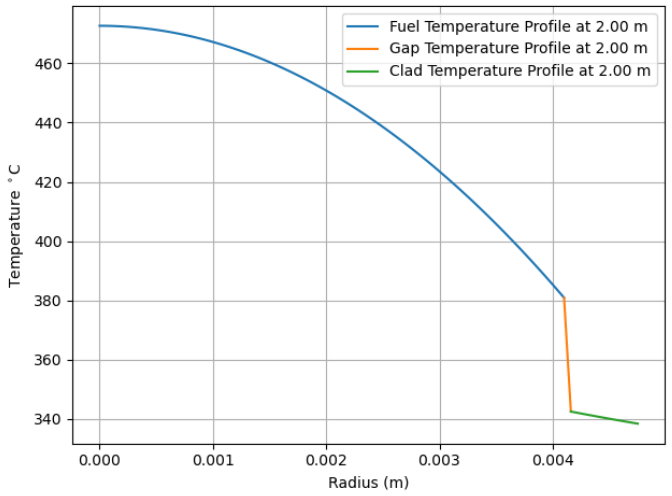

Fig. 8: Radial temperature profile for PWR at axial location of z = $H/2$

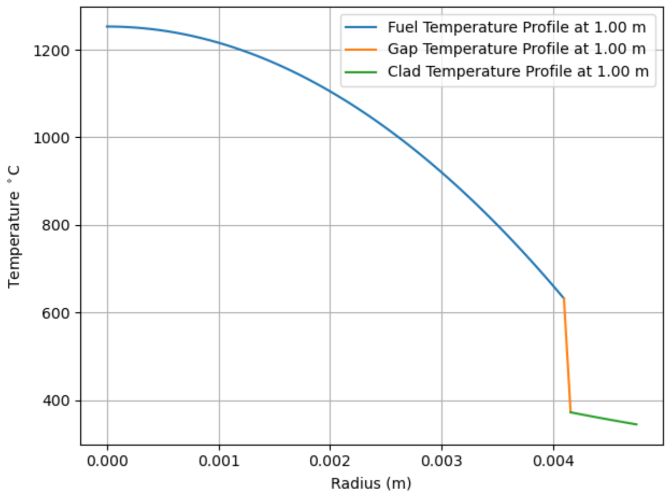

Fig. 9: Radial temperature profile for PWR at axial location of z = $H/4$

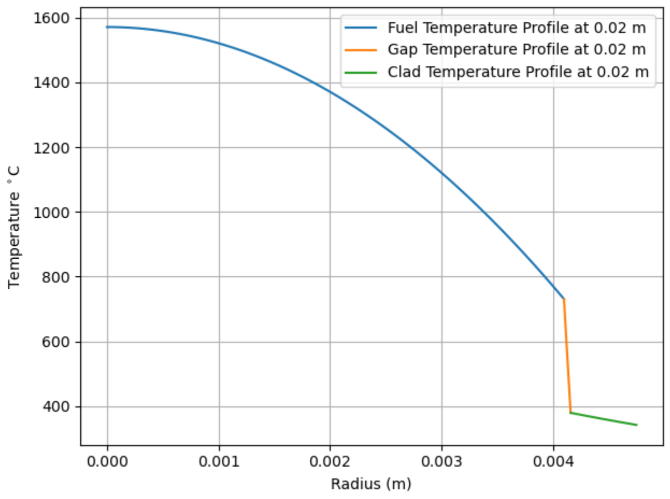

Fig. 10: Radial temperature profile for PWR at axial location of z = 0.02 m, which is where the max centerline temperature occurs

The calculated minimum departure from nucleate boiling ratio (MDNBR) was found to be 2.41745, which is drastically larger than the 1.3 requirement for PWRs, which means that this PWR does meet the thermal design criteria.

The maximum centerline temperature can easily change with changes in the boundary conditions. A higher linear heat generation rate over the entire channel would lead to much larger centerline temperatures, while decreasing the heat generation rate in the fuel would lead to lower centerline temperatures. Increasing or decreasing the flow rate shouldn't change the centerline temperature, though the fluid would heat up faster with a lower flow rate and cool down if the flow rate were increased. Changing the inlet properties of the fluid, like the temperature and pressure, would change the fluid's properties along the rest of the channel and would change the wall temperature of the system, but wouldn't affect the centerline temperature of the fuel.

## 4.2 Boiling Water Reactor

The geometric information and the initial properties for the fluid are all given in Table 1. The size of the numerical steps to accurately model the entire channel was selected through a mesh convergence, as can be seen in Fig. 11. Once the step size was selected, all of the necessary properties could be plotted with the height of the channel, $z$, on the y-axis. Fig. 12 demonstrates the behavior of the fluid, coolant, temperature, and the wall temperature in relation to the saturation temperature. Fig. 13 demonstrates the behavior of the fuel centerline, fuel surface, clad inner surface, and clad outer surface temperatures along the channel. Fig. 14 gives the pressure along the channel, Fig. 15 gives the density, Fig. 16 gives the void fraction, Fig. 17 gives both the equilibrium quality and critical quality, and Fig. 18 gives the departure from nucleate boiling ratio.

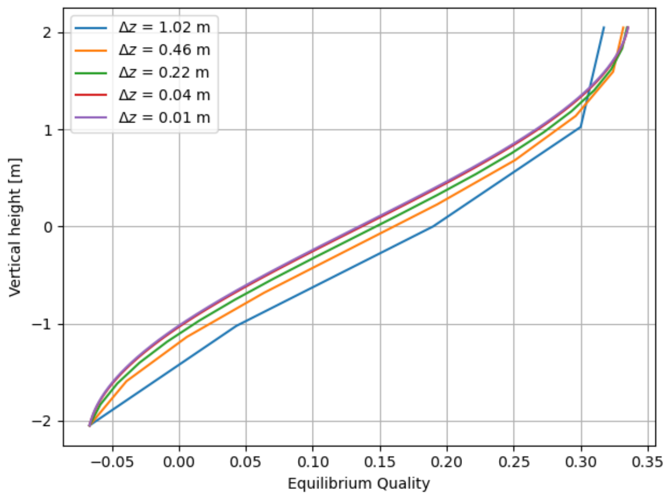

Fig. 11: $\chi_e$ convergence based on $\Delta z$ where smaller step sizes are tested to find a good balance between computational intensity and accuracy

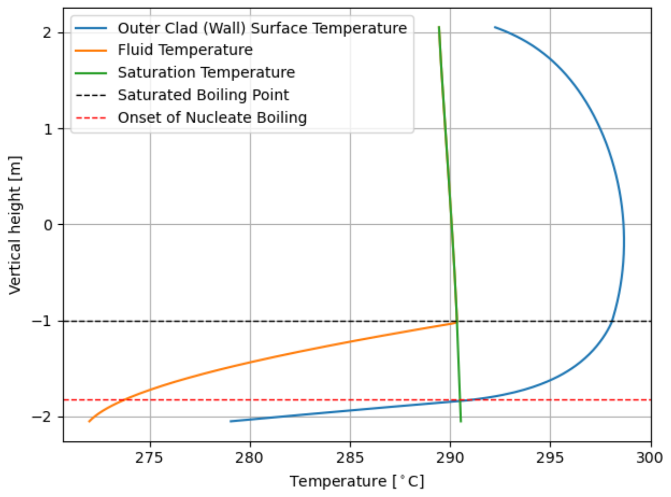

Fig. 12: Temperatures of fluid and wall with saturation temperature for comparison. The locations for the onset of nucleate boiling (ONB) and saturated boiling are also marked.

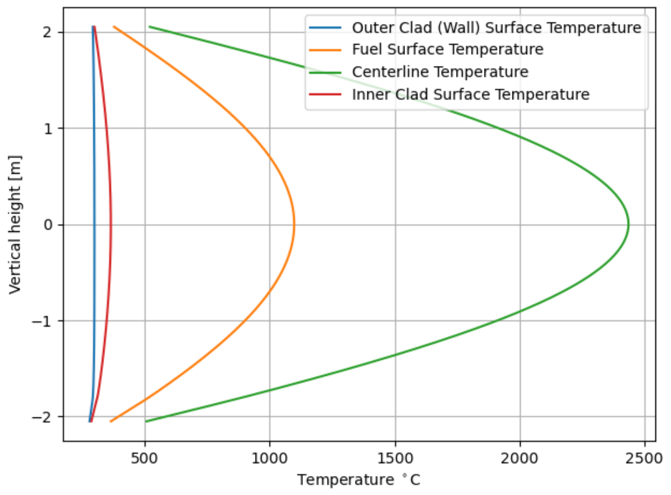

Fig. 13: Temperature at fuel centerline, fuel surface, clad inner surface and clad outer surface temperature

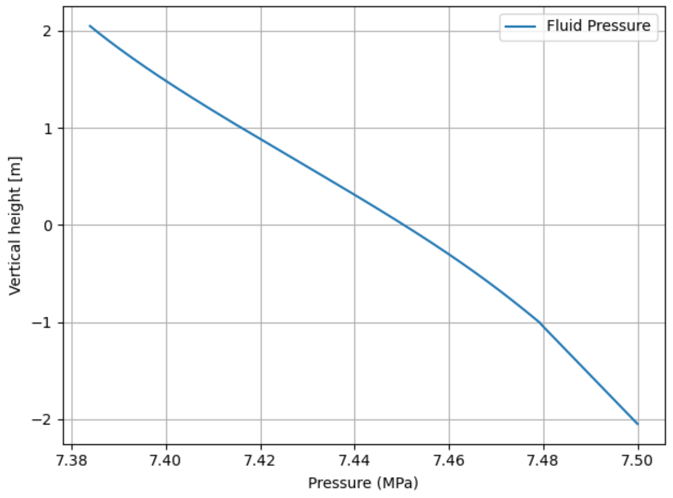

Fig. 14: Pressure of the water along the channel

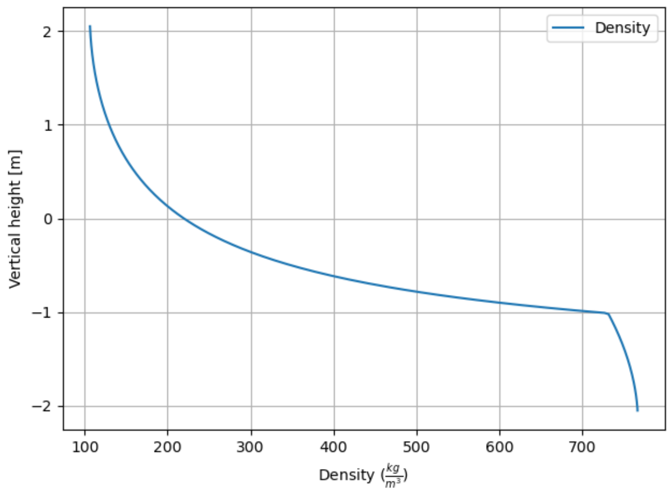

Fig. 15: Density of the fluid along the channel

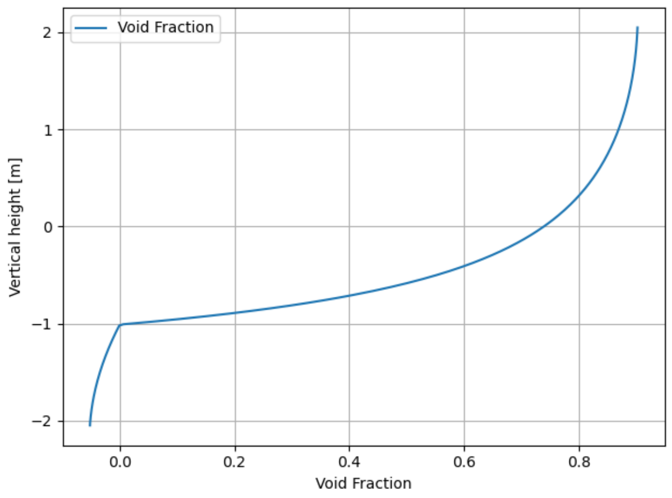

Fig. 16: Void Fraction of the fluid along the channel

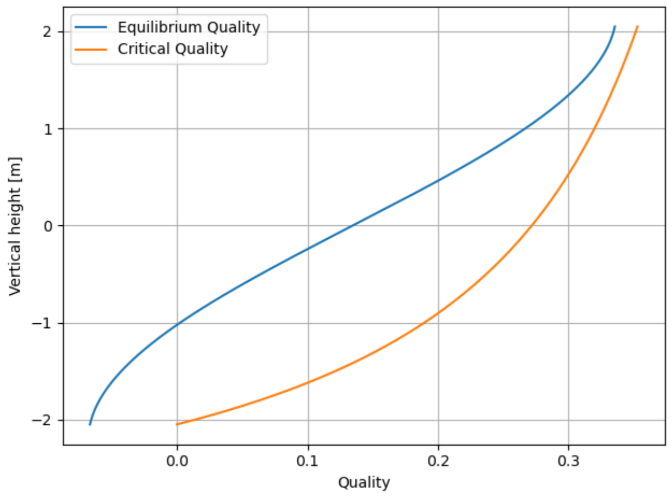

Fig. 17: Quality and Critical Quality of the fluid along the channel

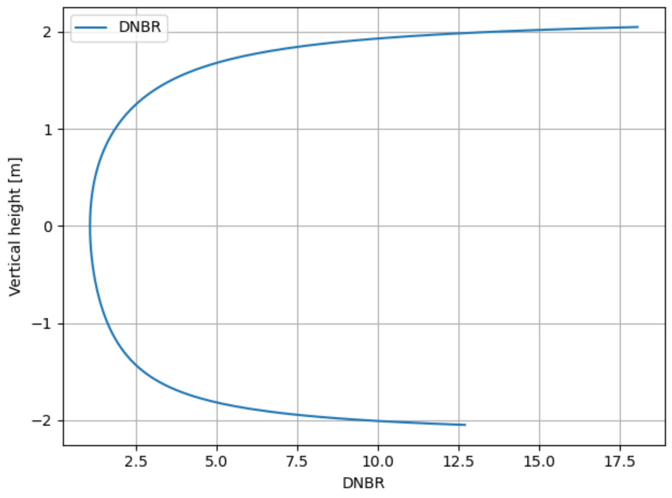

Fig. 18: Departure from nucleate boiling ratio, calculated by comparing the critical heat flux and actual heat flux

The maximum temperature for the fuel centerline is 2400 °C for the BWR case, and the fluid reaches the saturation temperature of 290°C. The clad outer surface reaches its maximum temperature of 295°C around the same axial location as the fuel centerline, as does the clad inner surface temperature. The radial temperature profiles at $z = H/2, H/4$, and $z_{max,CL}$ can be given by Fig. 19, 20, and 21 respectively.

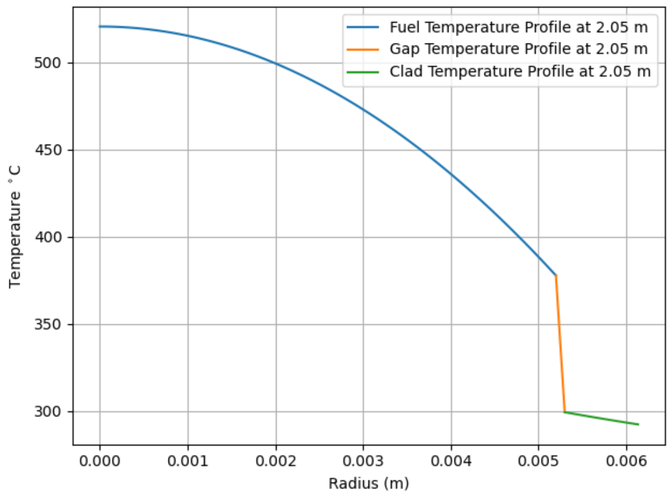

Fig. 19: Radial temperature profile for BWR at axial location of z = $H/2$

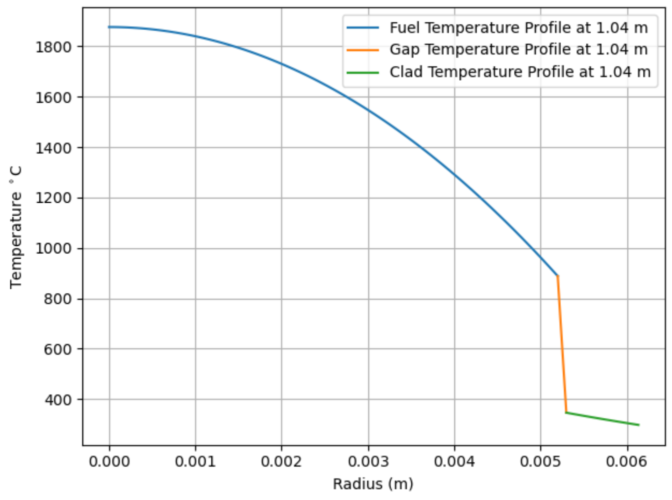

Fig. 20: Radial temperature profile for BWR at axial location of z = $H/4$

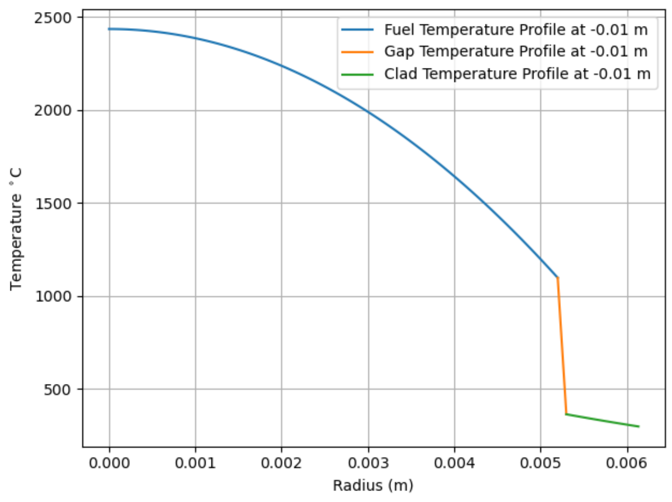

Fig. 21: Radial temperature profile for BWR at axial location of z = $-0.01$ m, which is where the max centerline temperature occurs

The minimum departure from nucleate boiling ratio (MDNBR) for the BWR is calculated to be 1.062813. The critical power ratio can be tested by multiplying the linear heat generation ratio by the CPR to get it to reach the critical quality. This value was calculated to be 1.042 as can be seen in Fig. 22.

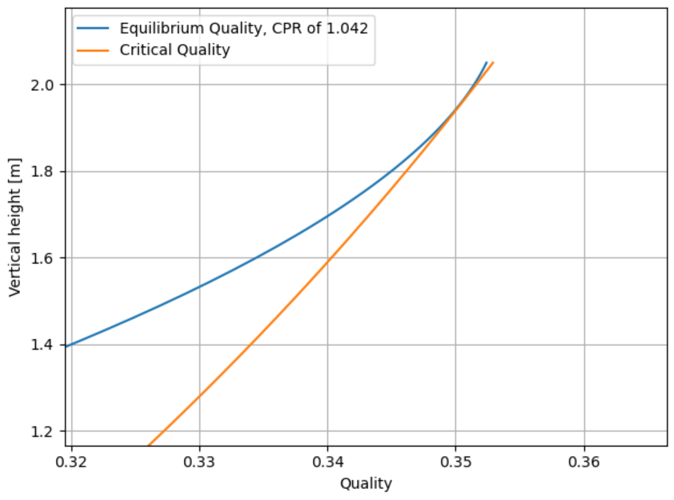

Fig. 22: Critical power ratio calculated by getting the quality to reach the critical quality

Both the MDNBR of 1.063 and CPR of 1.042 are below there necessary thermal design criteria which means this BWR does not meet the thermal design criteria.

The maximum centerline temperature can easily change with changes in the boundary conditions. A higher linear heat generation rate over the entire channel would lead to much larger centerline temperatures, while decreasing the heat generation rate in the fuel would lead to lower centerline temperatures. Increasing or decreasing the flow rate shouldn't change the centerline temperature, though the fluid would heat up faster with a lower flow rate and cool down if the flow rate were increased. Changing the inlet properties of the fluid, like the temperature and pressure, would change the fluid's properties along the rest of the channel and would change the wall temperature of the system, but wouldn't affect the centerline temperature of the fuel.

## 5 Conclusions

In this work, we explored the use of a numerical code to investigate the properties of channels simulated to be in PWR and BWR conditions separately. Using information from the models, the safety of each setup was judged, and it was found that though the PWR met its design criteria, the BWR didn't meet its design criteria. Other key insights were also acquired about each setup. The maximum temperatures for the fuel centerline in the PWR was less than 1600 while the temperature in the BWR reached 2400. The MDNBR for the PWR was 2.41745 which was much larger than the necessary 1.3, while the BWR's was only 1.063 giving rise to alarm. The BWR's density and void fraction are almost mirror images of each other which makes sense, the fluid is transitioning to a vapor as it travels through the channel which will lead to much larger void fractions and correspondingly smaller densities. The BWR equilibrium quality doesn't change as drastically but it can still be compared to critical quality to understand the thermal design limits built into the system. Since the CPR is only 1.042, this thermal design limit is not nearly enough and the modeled BWR would not be safe.

Though the 1 dimensional model makes a lot of assumptions to simplify the problem, it covers many keep phenomenon and if the modeled BWR can't pass a simplified code's analysis of its safety, it won't be able to pass anything more complicated. Though more resources and effort could be thrown into developing a more complete model, this would be moot if the design can't pass a simpler model which is why this project has its merits.

## 6 References

[1] C. Brooks S. Malik. *NPRE 449 Computer Project Fall 2025.* Urbana-Champaign, IL, USA.

[2] Neil E. Todreas and Mujid S. Kazimi. *Thermal-Hydraulic Fundamentals*, volume 1 of *Nuclear Systems*. MIT Press, Cambridge, MA, 1990.

## 7 Appendix (Code)

Also available at Arnav Goyal's GitHub.

```
import numpy as np
import matplotlib.pyplot as plt
from pyfluids import Fluid, FluidsList, Input
import scipy.integrate as spi


MM_water = 18.02 #g/mol

water = Fluid(FluidsList.Water).unspecify_phase()
critP = water.critical_pressure


def presQualProp(pres, qual, prop):
    """
    Parameters
    -----------
    pres: int
        pressure in pascals
    qual: int
        Number between 0 and 100 for flow quality
    prop: str
        one of the properties in the following list["vol", "intEnrg", "dynVisc", "enth", "rh
    """
    water_state = water.with_state(Input.pressure(pres), Input.quality(qual))

    if prop == "vol":
        return water_state.specific_volume #m3/kg
    elif prop == "intEnrg":
        return water_state.internal_energy #J/kg
    elif prop == "dynVisc":
        return water_state.dynamic_viscosity #Pa*s
    elif prop == "enth":
        return water_state.enthalpy # J/kg
    elif prop =="rho":
        return water_state.density #kg / m3
    elif prop == "pr":
        return water_state.prandtl
    elif prop == "k":
        return water_state.conductivity

def tempPresProp(temp, pres, prop):
    """
    Parameters
    -----------
    pres: int
        pressure in pascals
    temp: int
        Degrees celsius
    prop: str
        one of the properties in the following list["vol", "intEnrg", "dynVisc", "enth", "rh
    """
    water_state = water.with_state(Input.temperature(temp), Input.pressure(pres))

    if prop == "vol":
        return water_state.specific_volume #m3/kg
    elif prop == "intEnrg":
        return water_state.internal_energy #J/kg
    elif prop == "dynVisc":
        return water_state.dynamic_viscosity #Pa*s
    elif prop == "enth":
        return water_state.enthalpy # J/kg
    elif prop =="rho":
        return water_state.density #kg / m3
    elif prop == "pr":
        return water_state.prandtl
    elif prop == "k":
        return water_state.conductivity

def calchfg(pres):
    return presQualProp(pres, 100, "enth") - presQualProp(pres, 0, "enth") #J/kg

def calcXe(pres, temp):
    h = tempPresProp(pres=pres, temp=temp, prop="enth")
    hfg = calchfg(pres = pres)
    hf = presQualProp(pres=pres, qual=0, prop="enth")
    return (h-hf)/hfg

def calcNewTemp(Xe, pressure):
    hf = presQualProp(pres=pressure, qual=0, prop="enth")
    hfg = calchfg(pres=pressure)
    #print(Xe)
    #print(Xe*hfg)
    h = Xe*hfg + hf
    #print(Xe, h, hf, hfg)
    corresTemp = water.with_state(Input.enthalpy(h), Input.pressure(pressure)).temperature

    return corresTemp, h # degrees Celsius

def calcvfg(pres):
    return presQualProp(pres, 100, "vol") - presQualProp(pres, 0, "vol") #J/kg

def qCritUniform(pressure, xe, G, hf, hin, Dh):
    #W-3 Correlation
    pressure = pressure/(10**6) #convert to MPa
    hf = hf/(10**3) #convert to kJ/kg
    hin = hin/(10**3) #convert to kJ/kg
    term1 = ((2.022 -0.06238*pressure)+(0.1722-0.01427*pressure)*(np.e**((18.177-0.5987*pres
    term2 = ((0.1484-1.596*xe + 0.1729*xe*np.abs(xe))*2.326*G + 3271)
    term3 = (0.2664 + 0.8357*(np.e**(-124.1*Dh)))*(0.8258+0.0003414*(hf-hin))
    return term1*term2*term3*1000 #W/m2

def chiCrit(pressure, Lcr, G, D_equiv):
    #CISE-4 Correlation
    Gstar  = 3375* (1-(pressure/critP))**3
    if G < Gstar:
        a = (1+(1.481*(10**-4)*((1-(pressure/critP))**(-3))*G))**-1
    else:
        a = (1-(pressure/critP))/((G/1000)**(1/3))

    b = 0.199 * (((critP/pressure)-1)**(0.4)) * G * D_equiv**1.4

    return a*(Lcr/(Lcr+ b))

def voidFraction(rho, pres):
    rhog = presQualProp(pres, 100, "rho")
    rhof = presQualProp(pres, 0, "rho")
    return (rho-rhof)/(rhog - rhof)

option = 2 # 1 : PWR, 2: BWR
plotted = ["Critical Quality", "Equilibrium Quality", "Quality"]

# Options for plotted:
# "Linear Heat Generation", "Pressure", "Wall Temperature", "Fluid Temperature",
# "Centerline Temperature", "Saturation Temperature", "Clad Inner Temperature",
# "Equilibrium Quality", "Density", "Viscosity",  "Reynolds Number", "Friction Factor",
# "Heat Flux", "Fuel Surface Temperature", "Critical Heat Flux", "DNBR", "Void Fraction",
# "Axial Plot"

versions = [300]#[5, 10, 20, 100, 400] #number of points to calculate at over entire height
```

```
fig,ax = plt.subplots()


if option == 1 :

    ###variables###

    H = 4 # m
    He = 4.3 # m - extrapolated height
    D_rod = 0.95 * 0.01 #m diameter
    pitch = 1.26 * 0.01 #m distance between fuel pellets
    D_fuel = 0.82 * 0.01#m diameter
    gap_thick = 0.006 * 0.01 #m space between fuel and clad
    k_gap = 0.25 # W / m degC
    k_fuel = 3.6 # W / m degC
    k_clad = 21.5 # W / m degC

    r_fuel = D_fuel/2
    r_clad_i = r_fuel + gap_thick
    r_clad_o = D_rod/2

    G = 4000 #kg/ m2 s or density times velocity
    q0 = 380*100 # W/m linear initial heat generation
    P_low = 15*(10**6) #MPa at z=-H/2
    T_f_low = 277 #degC at z=-H/2

elif option == 2:
    ###variables###

    H = 4.1 # m
    He = 4.4 # m - extrapolated height
    D_rod = 1.227 * 0.01 #m diameter
    pitch = 1.62 * 0.01 #m distance between fuel pellets
    D_fuel = 1.04 * 0.01#m diameter
    gap_thick = 0.010 * 0.01 #m space between fuel and clad
    k_gap = 0.25 # W / m degC
    k_fuel = 3.6 # W / m degC
    k_clad = 21.5 # W / m degC

    r_fuel = D_fuel/2
    r_clad_i = r_fuel + gap_thick
    r_clad_o = D_rod/2

    G = 2350 #kg/ m2 s or density times velocity
    q0 = 605*100 # W/m linear initial heat generation
    P_low = 7.5*(10**6) #MPa at z=-H/2
    T_f_low = 272 #degC at z=-H/2


g = 9.8 #m/s2 acceleration due to gravity

wet_perim = np.pi * D_rod #m
area = pitch**2 - (np.pi*((D_rod/2)**2))
D_equiv = 4*area / wet_perim





def reynolds(visc):
    return (G*D_equiv/visc) # Unitless

def frictionfactor1phase(re):
    return 0.316 * re**(-0.25)

def frictionfactor2phase(re, pressure, chi):
    mug = presQualProp(pressure, 100, "dynVisc")
    muf = presQualProp(pressure, 0, "dynVisc")
    mux = ((muf*chi + mug - chi*mug)/(muf*mug))**-1
    #mux = chi*(mug-muf) + muf
    #print(re, muf/mux)
    return (0.046 * (re**(-0.2)) * (muf/mux)**(-0.2)), mux

def dittusBoelter(k, re, pr):
    return 0.023 * re**(0.8) * pr**0.4 * (k/D_equiv)

def liuWinterton(S, F, htc, hnb, tf, tsat, tw):
    temporary = (F*htc*(tw-tf))**2 + (S*hnb*(tw-tsat))**2
    return np.sqrt(temporary)



for numOfPoints in versions:


    Ps = [P_low]
    T_f_s = [T_f_low]
    Xes = [calcXe(pres=Ps[0], temp=T_f_s[0])]
    chis = [0]
    mus = [tempPresProp(T_f_low, P_low, 'dynVisc')]
    rhos = [tempPresProp(T_f_low, P_low, 'rho')]
```

```
alphas = [voidFraction(rhos[0], Ps[0])]

Res = [reynolds(mus[0])]
frics = [frictionfactor1phase(Res[0])]

hs = [tempPresProp(T_f_low, P_low, "enth")]
hfs = [presQualProp(P_low, 0, "enth")]
hfgs = [calchfg(pres=P_low)]
Prs = [tempPresProp(T_f_low, P_low, "pr")]
ks = [tempPresProp(T_f_low, P_low, "k")]
htcs = [dittusBoelter(ks[0], Res[0], Prs[0])]

Tsats = [water.dew_point_at_pressure(P_low).temperature]

############################################

zs, deltaz = np.linspace(-H/2, H/2, numOfPoints, retstep=True)

qlins = q0*np.cos(np.pi*(zs/He)) # W / m
qdoubles = qlins/(np.pi*D_fuel)

############################################

c_3 = (-qlins[0])/(2*np.pi*k_gap)
c_5 = k_gap * c_3 / k_clad
c_6 = (c_5*((-k_clad/(htcs[0]*r_clad_o)) - np.log(r_clad_o))) + T_f_s[0]
c_4 = (c_3 *np.log(r_clad_i)*((k_gap/k_clad)-1))+c_6
c_2 = (c_3*np.log(r_fuel)) + c_4 + (qlins[0]/(4*np.pi*k_fuel))

T_f_c = [c_2]
T_f_r = [c_3*np.log(r_fuel) + c_4]
T_c_in = [c_3*np.log(r_clad_i) + c_4]
T_ws = [c_5*np.log(r_clad_o) + c_6]

############################################
jump = 0 #Saturated boiling
jump2 = 0 #ONB
for i in range(1, len(zs)):
    print(f"{i}, {Ps[-1]/(10**6):0.2e}MPa, {Tsats[-1]:0.2f}C, {T_ws[-1]:0.2f}C, {T_f_s[


    if Xes[-1] < 0:
        f1phase = frics[0]
        frics.append(f1phase)
        rhos.append(tempPresProp(T_f_s[-1], Ps[0], 'rho'))
        deltaPcurrent = -1*((2*f1phase*(G**2)/(rhos[-1]*D_equiv)) + (rhos[-1]*g))
```

```
else:

    if jump==0 :
        jump = i

    Res.append(reynolds(mus[0]))
    f2phase, mux = frictionfactor2phase(Res[-1], Ps[0], chis[-1])
    mus.append(mux)
    frics.append(f2phase)
    vfg = calcvfg(pres=Ps[0])
    vf  = presQualProp(Ps[0], 0, "vol")
    rhos.append(1/(vfg*chis[-1] + vf))
    deltaPcurrent = -1*(((G**2)*vfg*(chis[-1]-chis[-2])/deltaz) + (f2phase*(G**2)*(v

P_new = deltaPcurrent*deltaz + Ps[-1]
Ps.append(P_new)

#print(P_new)
Tsats.append(water.dew_point_at_pressure(Ps[-1]).temperature)

hfgs.append(calchfg(pres=Ps[-1]))
hfs.append(presQualProp(Ps[-1], 0, "enth"))
newXe = (((qdoubles[i]*4*deltaz/(D_equiv*G))-(hfs[-1]-hfs[-2])-(Xes[-1]*(hfgs[-2]-hf
Xes.append(newXe)
newT_f, newh = calcNewTemp(Xes[-1], pressure=Ps[-1])


if newT_f < Tsats[-1]:
    T_f_s.append(newT_f)
else:
    T_f_s.append(Tsats[-1])
hs.append(newh)
htcs.append(dittusBoelter(ks[-1], Res[-1], Prs[-1]))
c_3 = (-qlins[i])/(2*np.pi*k_gap)
c_5 = k_gap * c_3 / k_clad
if Xes[-1]< 0 and T_ws[i-1] <= Tsats[i-1]:
    c_6 = (c_5*((-k_clad/(htcs[i]*r_clad_o)) - np.log(r_clad_o))) + T_f_s[i]
else:
    if jump2 == 0:
        jump2 = i
    curP = Ps[i]
    F = (1+(chis[-1]*Prs[-1]*((presQualProp(curP, 0, "rho")/presQualProp(curP, 100,
    S = (1+(0.055*(F**0.1)*(Res[0]**0.16)))**(-1)
    hnb = 55*((curP/critP)**0.12)*(qdoubles[i]**(2/3))*((-1*np.log10(curP/critP))**-
    A1 = (F**2) * (htcs[i]**2)
```

```
            A2 = (S**2) * (hnb**2) #If F and S are flipped there is a discontinuity in the g
            L1 = A1+A2
            L2 = -2*(A1*T_f_s[i] + A2*Tsats[i])
            L3 = A1*(T_f_s[i]**2) + A2*(Tsats[i]**2) - ((k_clad*c_5/r_clad_o)**2)
            #print(f"F = {F}, S = {S:0.2f}, hnb={hnb:0.2f}, A1 = {A1:0.2f}, A2 = {A2:0.2f},
            #T_w_min = ((-L2-((L2**2 - (4*L1*L3))**0.5))/(2*L1))
            T_w_plus = ((-L2 +((L2**2 - (4*L1*L3))**0.5))/(2*L1))
            #print("Wall temp", T_w_plus)
            c_6 = T_w_plus - (c_5*np.log(r_clad_o))

        c_4 = (c_3 *np.log(r_clad_i)*((k_gap/k_clad)-1))+c_6
        c_2 = (c_3*np.log(r_fuel)) + c_4 + (qlins[i]/(4*np.pi*k_fuel))

        T_f_c.append(c_2)
        T_f_r.append(c_3*np.log(r_fuel) + c_4)
        T_c_in.append(c_3*np.log(r_clad_i) + c_4)
        T_ws.append(c_5*np.log(r_clad_o) + c_6)

        if Xes[-1] < 0:
            chis.append(0)
        elif Xes[-1] > 1:
            chis.append(1)
        else:
            chis.append(Xes[-1])

        alphas.append(voidFraction(rhos[-1], Ps[-1]))

    qcrits = []
    xcrits = []
    dnbrs = []
    intercept = None
    for i in range(len(zs)):
        qCrit = (qCritUniform(Ps[i], Xes[i], G, hfs[i], hs[0], D_equiv))
        F = 1
        qcrits.append(qCrit/F)
        dnbrs.append(qCrit/(qdoubles[i]*F))
        xcrit = chiCrit(Ps[i], zs[i]+H/2, G, D_equiv)
        xcrits.append(xcrit)

    #print(np.min(dnbrs))# - 1.0628126932088091 for BWR, 2.417453195453736 for PWR

    if "Axial Plot" in plotted:
        heightLoc = np.where(np.array(zs)>=H/2)[0][0]

        c_3 = (-qlins[heightLoc])/(2*np.pi*k_gap)
        c_5 = k_gap * c_3 / k_clad
        if Xes[-1]< 0 and T_ws[heightLoc] <= Tsats[heightLoc]:
            c_6 = (c_5*((-k_clad/(htcs[heightLoc]*r_clad_o)) - np.log(r_clad_o))) + T_f_s[he
        else:
            curP = Ps[heightLoc]
            F = (1+(chis[-1]*Prs[-1]*((presQualProp(curP, 0, "rho")/presQualProp(curP, 100,
            S = (1+(0.055*(F**0.1)*(Res[0]**0.16)))**(-1)
            hnb = 55*((curP/critP)**0.12)*(qdoubles[heightLoc]**(2/3))*((-1*np.log10(curP/cr
            A1 = (F**2) * (htcs[heightLoc]**2)
            A2 = (S**2) * (hnb**2) #If F and S are flipped there is a discontinuity in the g
            L1 = A1+A2
            L2 = -2*(A1*T_f_s[heightLoc] + A2*Tsats[heightLoc])
            L3 = A1*(T_f_s[heightLoc]**2) + A2*(Tsats[heightLoc]**2) - ((k_clad*c_5/r_clad_o
            T_w_plus = ((-L2 +((L2**2 - (4*L1*L3))**0.5))/(2*L1))
            #print("Wall temp", T_w_plus)
            c_6 = T_w_plus - (c_5*np.log(r_clad_o))

        c_4 = (c_3 *np.log(r_clad_i)*((k_gap/k_clad)-1))+c_6
        c_2 = (c_3*np.log(r_fuel)) + c_4 + (qlins[heightLoc]/(4*np.pi*k_fuel))

        rs1 = np.linspace(0, r_fuel, 100)
        T_1s = -qlins[heightLoc]*(rs1**2)/(4*np.pi*k_fuel*(r_fuel**2)) + c_2
        rs2 = np.linspace(r_fuel, r_clad_i, 100)
        T_2s = c_3*np.log(rs2) + c_4
        rs3 = np.linspace(r_clad_i, r_clad_o, 100)
        T_3s = c_5*np.log(rs3) + c_6
        ax.plot(rs1, T_1s, label=f"Fuel Temperature Profile at {zs[heightLoc]:0.2f} m")
        ax.plot(rs2, T_2s, label=f"Gap Temperature Profile at {zs[heightLoc]:0.2f} m")
        ax.plot(rs3, T_3s, label=f"Clad Temperature Profile at {zs[heightLoc]:0.2f} m")

    if "Linear Heat Generation" in plotted:
        ax.plot(qlins, zs,  label="q\'")
    if "Heat Flux" in plotted:
        ax.plot(qdoubles, zs,  label="q\'\'")
    if "Pressure" in plotted:
        ax.plot(np.array(Ps)/(10**6), zs, label="Fluid Pressure")
    if "Wall Temperature" in plotted:
        ax.plot(T_ws, zs, label="Outer Clad (Wall) Surface Temperature")
    if "Fluid Temperature" in plotted:
        ax.plot(T_f_s, zs, label="Fluid Temperature")
    if "Fuel Surface Temperature" in plotted:
        ax.plot(T_f_r, zs, label="Fuel Surface Temperature")
    if "Centerline Temperature" in plotted:
        ax.plot(T_f_c, zs, label="Centerline Temperature")
    if "Saturation Temperature" in plotted:
        ax.plot(Tsats, zs, label="Saturation Temperature")
    if "Clad Inner Temperature" in plotted:
        ax.plot(T_c_in, zs, label="Inner Clad Surface Temperature")
    if "Equilibrium Quality" in plotted:
        ax.plot(Xes, zs, label=f"Equilibrium Quality")
    if "Critical Quality" in plotted:
        ax.plot(xcrits, zs, label="Critical Quality")

    if "Density" in plotted:
        ax.plot(rhos, zs, label="Density")
    if "Viscosity" in plotted:
        ax.plot(mus, zs, label="Dynamic Viscosity")
    if "Reynolds Number" in plotted:
        ax.plot(Res, zs, label="Reynolds Number")
    if "Friction Factor" in plotted:
        ax.plot(frics, zs,  label = "Friction Factor")
    if "Critical Heat Flux" in plotted:
        ax.plot(qcrits, zs,  label = "Critical Heat Flux")
    if "DNBR" in plotted:
        ax.plot(dnbrs, zs,  label = "DNBR")
    if "Void Fraction" in plotted:
        ax.plot(alphas, zs, label="Void Fraction")


#ax.set_xscale('log')
#ax.axhline(zs[jump], 0, 1, color='k', linestyle = '--', linewidth= 1, label="Saturated Boi
#ax.axhline(zs[jump2], 0, 1, color='r', linestyle = '--', linewidth= 1, label="Onset of Nuc
ax.set_ylabel(f"Vertical height [m]")
if "Axial Plot" in plotted:
    ax.set_ylabel(f"Temperature $^\circ$C")
ax.set_xlabel(f"{plotted[-1]}")
ax.grid()
ax.legend()
plt.tight_layout()

plt.show()
```