TALLINN UNIVERSITY OF TECHNOLOGY
School of Science
Department of Cybernetics

Kaspar Selke 212094YAFM

SIMULATING PRESSURISED WATER REACTOR USING MODELICA

Master's Thesis

Supervisor: Marti Jeltsov
PhD

Tallinn 2024

TALLINNA TEHNIKAÜLIKOOL
Loodusteaduskond
Küberneetika instituut

Kaspar Selke 212094YAFM

SURVEVEEREAKTORI MODELLEERIMINE MODELICAGA

Magistritöö

Juhendaja: Marti Jeltsov
PhD

Tallinn 2024

Author’s Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Kaspar Selke

01.01.2024

Supervisor's Approval

The thesis complies with the requirements for Master's theses.

Supervisor: Marti Jeltsov

01.01.2024

Abstract

In this thesis we created a preliminary pressurised water reactor model. The model was developed using Modelica language with the Dymola interface. The main library used for the reactor components was the TRANSFORM library. The final system reaches steady state on its own and three different transients tests were performed and analyzed. The reactor model was a success and the goal of the thesis was achieved although further validation and verification is needed. Multiple steps have to be taken before real world decisions can be made based on the model.

Annotatsioon

Surveveereaktori modelleerimine Modelicaga

Antud töös lõime esialgse surveveerreaktori mudeli. Mudel valmistati kasutades Modelica programeerimis keelt koos Dymola liidesega. Pealmine teek mida kasutati reaktori komponentideks oli TRANSFORM. Lõplik süsteem saavutab stabiilse seisundi enda pea ja kolm erinevat katse olukorda testiti ja analüüsiti. Reaktori mudel oli edukas ja töö eesmärk saavutati, kuigi järgnevat validatsiooni ja tõestamist on tarvis. Mitu sammu peab veel võtma enne kui mudel on valmis päris maailmas kasutamiseks ja otsuste võtmiseks.

Table of Contents

1. Introduction

Computer modelling is a powerful process to forecast and understand the events that haven't yet taken place in the physical world. It allows us to learn about and design systems without having to spend the materials and facing the difficulties of building a physical prototype. As nuclear systems are large and complex, it is challenging to perform physical, real world experiments and computer modelling has become the main design and analysis tool. However, computer modelling isn't perfect and comes with its' share of issues. The most well-known being its dependence on human knowledge of physics, human error and lack of validation data. Most of these are inherent to computer modelling. Within this field the things we can optimize are accuracy with better physical models, computational time with faster code and for large models the large overhead can be reduced by using more efficient systems or languages. These three are what Modelica is trying to achieve. To reduce the cost of building and maintaining models while also making them reasonably fast a library of compatible components with well defined physical assumptions is provided. This thesis aims to test Modelica and its development environment Dymola by developing a Pressurized Water Reactor (PWR) model using the TRANSFORM library. This research aligns with Estonia's considerations on deployment of nuclear power to meet its power system and climate goals and supports the plans by developing reactor technology and modelling related local expertise. The reason for selecting a pressurized water reactor over a boiling one is that the PWR core that consists of single-phase, all liquid water is somewhat simpler to model compared to very non-linear turbulent two-phase in case of a BWR. Since the purpose of the thesis is to gain the knowledge of how to work with Modelica and develop a full reactor model, the PWR type was chosen. Since currently there is no prevailing reactor modelling framework or tools in Estonia, options like Modelica might have use in the future while also giving students the possibility to practise and learn nuclear computer modelling.

2. Background

In this section a brief history of Dymola and Modelica is presented and their main working principles described. Also the main collection of components used in the model is introduced.

Modelica is a freely available modelling language developed and maintained by the Modelica Association [1]. It was first released in 1999 alongside Dymola which was the first commercial tool that supported the language. Dymola is a component based tool for modelling and simulation [2]. The original idea behind Dymola dates back to 1970's and were developed by Dr Hilding Elmqvist as a PhD thesis. Modelica has a free editing tool as well called OpenModelica, but due to differences in how they compile the models the library required for building a nuclear reactor was only available using Dymola. Later Dymola became a part of Dassault Systèmes. It solves the problems of overhead and computational efficiency in many ways. First of them being a component-oriented approach allowing for components of different domains such as electrical, hydraulic, thermal etc. to be combined and done so with efficiency since all components are checked beforehand. The second is an equation based approach meaning all models are described mathematically using differential, discrete, and algebraic equations. Third and possibly the most useful feature is Symbolic Manipulation. Most modelling and programming languages use the form $variable = expression$ and code is read procedurally. This means that first variables are given values and then a computation can be made based on an expression with those variables and the value given to a new variable. This requires that the model developer manually rearranges the equations to suit the problem for every possible way of calculating a given necessary value. For example, having acceleration as input but also sometime requiring that instead of acceleration, speed or position can be given. This usually causes one model to have multiple versions causing large overheads due to maintenance. The work required increases with rising complexity. Dymola uses $expression = expression$ form so that the program itself can arrange the equations as needed to suit a particular problem. For example we might have an line that computes $x = y * z$. Traditionally if we had x, z given and wanted the value of y instead we would need to write a separate line $y = \frac{x}{z}$. With more complex equations this adds up to multiple lines of code and can even create compounding amounts of work required. Dymola on the other hand can do this automatically if at any point any of the values in an expression need to be calculated it can rearrange the equation as required. In practice this Symbolic Manipulation also gives us an increase in speed due to less equations required. Another reason why it was decided to use Modelica and more specifically Dymola in this thesis work is due to the TRANSFORM library. It is a library of components designed for building nuclear reactors. Although most of the focus has been on molten salt reactors, it has the necessary components for more common reactor types such as pressurized water reactors. The library is developed by Scott Greenwood [3]. The library was developed with the idea that the user learns through examples that have been included. This means that it has minimal documentation and the learning curve for new users is extremely steep, especially those who have no previous experience with nuclear modelling or modelling in general. Although active development of TRANSFORM has stopped it still receives bug fixes semi regularly. An additional reason for choosing Modelica is it's successful and wide use in variety of fields. This gives the students better opportunities in the future while also giving them the skills to focus further into other nuclear modelling systems.

3. Theory

In the thesis we are assuming that the reader has basic knowledge of modelling and nuclear physics ideally up to the level of cross-sections. If the reader wants to model their own reactor, then basic knowledge of Modelica language is also required. In this section we will go over the basic nuclear reactor physics required to understand how a reactor core operates, the main value used to describe the activity of the reactor and the nuclear kinetics model this thesis uses.

3.1 Basic Nuclear and Reactor Physics

These sections and related subsections about reactivity and multiplication factor were written based on U.S Department of Energy nuclear energy handbook volume 1 [4] and volume 2 [5].

Reactors work by using nuclear reactions with fissile material to be in a constant critical state. The critical state is a state of self-sustaining reactions where all excess neutrons either escape or are captured by something other than the fuel. Typical fission reactions have two fission products, some number of prompt neutrons, prompt gamma and then released energy. The probability of this reaction happening is dependent on the energy of the neutron as seen in figure 1. Here a larger cross-section can be viewed as a higher probability of the event happening.

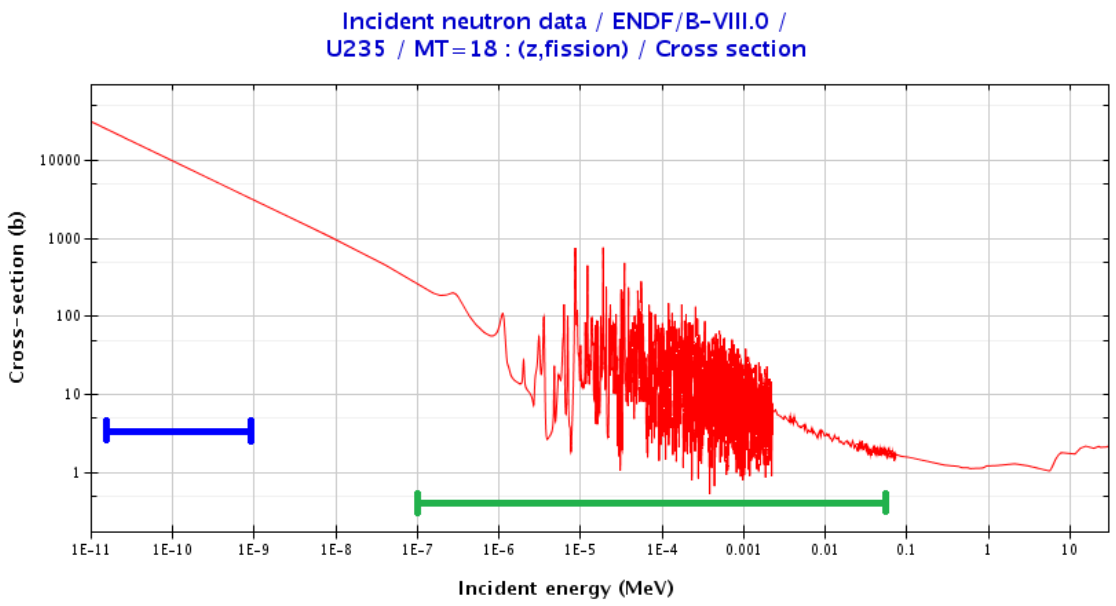

Figure 1. *U-235 fission cross-section dependence on neutron energy. The blue indicates energies of thermal neutrons and the green is the resonance region. Source: OECD NEA Janis database [6].*

From figure 1 we can also see why we want to reduce the energy of the neutrons. It is because neutrons of lower energy have a higher probability to produce fission reactions leading to more useful energy and we also want to avoid the resonant part which can be see in the graph. It is also important to know that the delayed fission reactions can produce neutrons upon decay and these are important in keeping the reactor stable and have to therefore be considered in modelling. Furthermore the heat produced by the decay of these materials continues on for a long time after the main fission reaction has stopped. The number of neutrons in a reactor in time can either decrease, increase or stay the same. This "trend" is represented in nuclear reactor physics by the effective multiplication factor k_{eff} which is relevant for finite reactors.

$$k < 1 \rightarrow \text{subcritical}$$
$$k = 1 \rightarrow \text{critical}$$
$$k > 1 \rightarrow \text{supercritical}$$

For reactors which are very well known and defined we can use the six-factor formula to calculate k_{eff}. In our case the calculation for k_{eff} is more complex and will be explained later into the chapter. It is still useful to explain six-factor formula as it gives us a good overview how the reactor stays at a critical state and other factors that help maintain it.

To understand k_{eff} we must first understand k_{∞}, which is known as the infinite multiplication factor. Both tell us how many neutrons were produced from fission in one neutron generation compared to how many neutrons were lost in the previous generation. This means that if $k_\infty = 1$ then the same amount of neutrons were produced as were lost, meaning there is no change in the neutron population and the reaction is self-sustaining. The formula for calculating k_∞ is called the four-factor formula. One has to consider an infinitely large system which has no neutron leakage due to being infinite. In this system four parameters define the multiplication ability of the system. These four parameters are the fast fission factor, resonance escape probability, thermal utilization factor and reproduction factor.

3.1.1 Fast Fission Factor

The fast fission process ϵ is the first process that a new generation of neutrons takes part in. It is the fission of neutrons at much higher than thermal energies. In figure 1 the energy of fast neutrons is around 1 MeV. This process has a lot lower probability of happening compared to fission at thermal energies as can be seen in figure 1 so it only creates a small amount of fission but this also depends on the concentrations of the fuel and the arrangement of the core.

3.1.2 Resonance Escape Probability

The way for neutrons to reduce in energy is by colliding with a moderator like water. Each time they collide with moderator molecules they loose some amount of energy. Ideally we would want them to end up at thermal energies which is noted in blue in figure 1. Sometimes they lose a certain amount of energy and end up colliding with fuel while in the resonance region seen in the same figure as green. The resonance escape probability p is the probability of neutrons reaching thermal energy without being captured in this resonance zone and is defined as the ratio of neutrons that reach thermal energies compared to the number of fast neutrons that are slowed down. Elements in the core such as ^{238}U and other isotopes have wider resonance bands and there for contribute strongly to the absorption of neutrons in this region. The value of resonance escape probability thus strongly depends on the fuel composition and also the geometry of the core. If neutrons manage to escape the fuel matrix they have much higher chance of slowing down without reacting to ^{238}U or other isotope atoms resulting in a higher resonance escape probability. Resonance escape probability is also one of the main safety features and contributes to reactor stability.

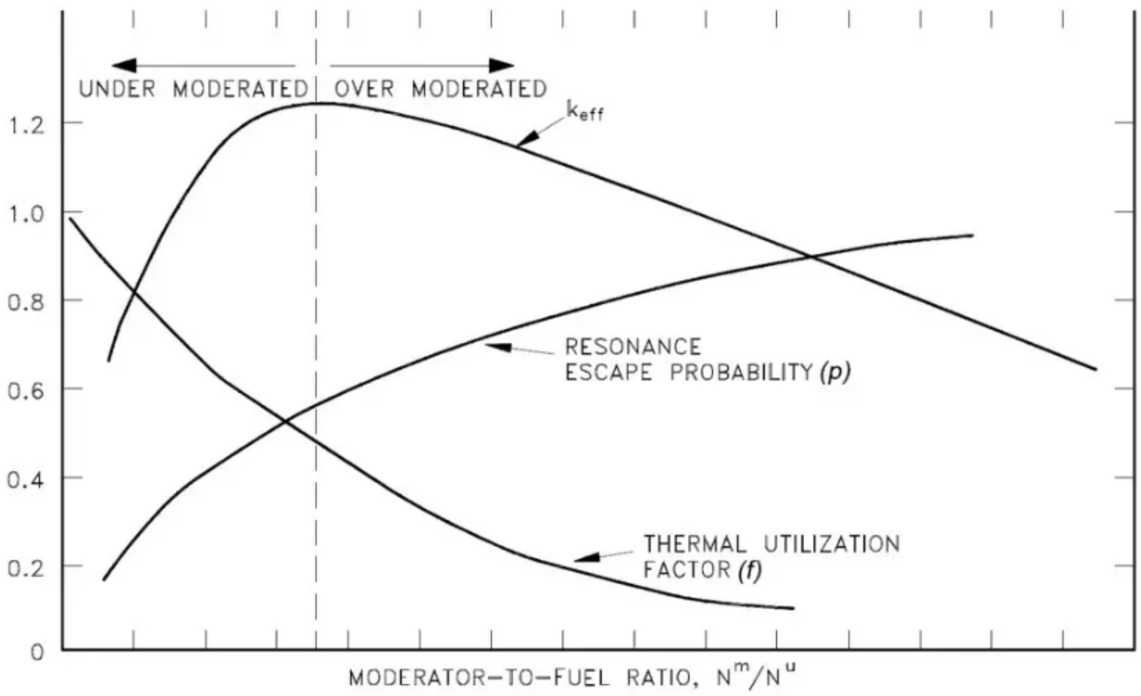

Figure 2. *Effective multiplication factor (k_{eff}), resonance escape probability (p), and thermal utilization factor (f) dependence on moderator to fuel ratio. The y-axis is the neutron generation multiplication factor. [7]*

These days most light water reactors are designed in the under moderated region seen in figure 2. This means that the reactors aren't actually designed at the optimal point. As the moderator temperature increases the density decreases causing shift in figure 2 to the left meaning that more neutrons are absorbed in the resonance region, that is if the reactor is operating in the under moderated zone. This decreases the amount of neutrons taking part in fission reactions and in turn lowers the power output, which will under normal circumstances reduce the temperature of the moderator and the core will slowly stabilize. This is also known as the moderator temperature coefficient (MTC). The second way it helps to keep everything stable is Doppler broadening. This is the fastest feedback we have since it takes place within the fuel and is considered almost instant. The aforementioned MTC takes time since the water has to warm up. The neutron production of the fuel changes with temperature and in most reactors this change coefficient is negative meaning higher temperatures will reduce the neutron production rate. The simplified theory behind the Doppler effect is that when the fuel nuclei lattice, that is the atomic structure of the fuel, starts to vibrate the relative velocity between neutron and nucleus changes. The effect of this change in relative velocity is the broadening of the resonant capture area and a shortening of its cross-section. Meaning neutrons can now be captured at higher energies by the resonance region, thus less neutrons are left for fission processes.

3.1.3 Thermal Utilization Factor

Next come the neutrons that didn't get absorbed into the resonance region. If they remain in the core they will reach thermal energies, but before fission they also have the probability of being absorbed by the various other materials inside the core such as cladding, control rods, moderator and so on. The thermal utilization factor f there for is the ratio of neutrons that get absorbed into the nuclear fuel divided by the number of neutrons that are absorbed into all the materials in the core. This process of non-fuel materials absorbing thermal neutrons is also referred to as parasitic absorption.

3.1.4 Reproduction Factor

The last step the neutrons go through is reproduction. This is where finally the neutrons reach the fuel and induce a fission reaction. To consider the reproduction factor in fresh uranium fuel only three uranium isotopes have to be considered but as the fuel burns up, different isotopes are created which then also have to be taken into consideration when calculating the reproduction factor. There is also a chance that even if the neutrons get absorbed they don't produce a fission reaction but rather result in radiative capture, creating a new compound nucleus and not producing any neutrons or a different amount. The reproduction factor η is there for the ratio of fast neutrons produced by thermal fission divided by the number of thermal neutrons absorbed into fuel.

These four together form the four-factor formula also called the infinite multiplication factor k_∞:

$$k_\infty = \epsilon\, p\, f\, \eta \tag{3.1}$$

From this we can get k_{eff} by adding two more factors: the fast non-leakage probability and the thermal non-leakage probability. When before we considered an infinite reactor now we are considering a finite one. Due to this we have to add the two factors.

3.1.5 Fast Non-leakage Probability

When we were assuming an infinite reactor leakage could be neglected but now moving to a finite reactor we can't anymore. The fast non-leakage probability P_f characterizes the amount of fast neutrons that do not leak out of the reactor. It is a ratio of the number of fast neutrons that do not leak over the number of fast neutrons that were produced.

3.1.6 Thermal Non-Leakage Probability

Very similar to fast non-leakage probability, only this time its the leakage of thermal neutrons P_T and is defined as the ratio between the number of thermal neutrons that do not leak from the reactor core and the number of neutrons that reach thermal energies. Both of these probabilities are influenced by the moderator temperature. An increase in moderator temperature results in an increase of leakage. This is one of the main ways the moderator temperature coefficient is kept negative in PWRs. These probabilities can sometimes be combined into one called the total non-leakage probability.

Having added the two factors we have now reached an adequate level of understanding of how the core stays stable. The six-factor formula there for is:

$$k_{eff} = \epsilon \, p \, f \, \eta \, P_f \, P_T \tag{3.2}$$

k_{eff} is one of the most important parameters and usually tells us a lot about the status of the reactor. Instead of k_{eff} however a different parameter called reactivity ρ is more commonly used which is a measure of a reactors relative departure from criticality.

$$\rho = \frac{k_{eff} - 1}{k_{eff}} \tag{3.3}$$

When changing to rho the critical state is then determined as:

$$\begin{aligned} &\rho < 0 \rightarrow \text{subcritical} \\ &\rho = 0 \rightarrow \text{critical} \\ &\rho > 0 \rightarrow \text{supercritical} \end{aligned}$$

As was discussed earlier the computation for k_{eff} in a less well defined/studied reactor is a lot more complicated. It usually requires the solving of an eigenvalue problem where k_{eff} is the eigenvalue of that system. The explanation of how this works is outside the scope of this thesis.

3.1.7 Point Kinetics

This subsection was written based on the book "Dynamics and Control of Nuclear Reactors"[8].

To know how much power our reactor is making we need a set of ordinary differential equations that describe the time evolution of neutron density. Dynamic equations can be formulated in the following logic:

Rate of change of quantity = rate of production – rate of losses (3.4)

Using this logic the equations for a point kinetics model can be presented as follows:

$$\frac{\partial n}{\partial t} = F_p\, P_T + P_d - L_{a+L} \tag{3.5}$$

$$\frac{\partial C_i}{\partial t} = F_d\, P_T - L_{decay}, \qquad i = 1, 2, \ldots, 6 \tag{3.6}$$

Here n is the neutron density per unit volume, F_p is the fraction of neutrons released promptly, P_T is the total rate of neutrons produced by fission, P_d is the rate of release for delayed neutrons, L_{a+L} is the rate at which neutrons are lost by absorption and leakage, F_d is the fraction of neutrons that are released by delayed neutron precursors, C_i is the concentration of i-th delayed neutron precursor, and L_{decay} is the rate of decay of the delayed precursors.

From here we can define β which is defined as the total fraction of neutrons produced that are delayed and β_i will be the fraction of fission reactions that result in delayed neutron precursors C_i being produced. We get the loss of precursors C_i by using a radioactive decay constant λ_i as $\lambda_i C_i$. Typically precursors are grouped into six groups. This is mainly done because if we do it by isotope we won't know if we have missed a specific precursor. It has been shown to work well and determining all delayed neutron precursors is a difficult challenge. Also there is the added benefit of speeding up the model computation. With these additions we can write the previous equations as:

$$\frac{\partial n}{\partial t} = (1-\beta)P_T - L_{a+L} + \sum_{i=1}^{6} \lambda_i C_i \tag{3.7}$$

$$\frac{\partial C_i}{\partial t} = \beta_i P_T - \lambda_i C_i \qquad i = 1, 2, \ldots, 6 \tag{3.8}$$

We can write eq 3.7 as:

$$\frac{\partial n}{\partial t} = P_T[(1-\beta) - \frac{L_{a+L}}{P_T}] + \sum_{i=1}^{6} \lambda_i C_i \tag{3.9}$$

We can now see that $\frac{L_{a+L}}{P_T}$ is actually k_{eff} since the fraction is the ration of total neutrons produced by fission divided by the total amount of neutrons lost. Using this we can turn the equation into the following:

$$\frac{\partial n}{\partial t} = P_T[(1-\beta) - \frac{1}{k_{eff}}] + \sum_{i=1}^{6} \lambda_i C_i \tag{3.10}$$

From earlier when we defined reactivity we can use equation 3.3 and in turn we get:

$$\frac{\partial n}{\partial t} = P_T(\rho + \beta) + \sum_{i=1}^{6} \lambda_i C_i \tag{3.11}$$

The total neutron production P_T can also be written as:

$$P_T = \nu \Sigma_f n v, \tag{3.12}$$

where ν is the average number of neutrons produced per fission, Σ_f is the macroscopic cross section for fission, and v the velocity of neutrons. It can be shown that the multiplication of $\nu\Sigma_f v$ is the time between the production of a neutron and the production of the next generation of neutrons by fission called Λ. After replacing it in we get:

$$\frac{\partial n}{\partial t} = \frac{(\rho - \beta)}{\Lambda} n + \sum_{i=1}^{6} \lambda_i C_i \tag{3.13}$$

$$\frac{\partial C_i}{\partial t} = \frac{\beta_i}{\Lambda} n - \lambda_i C_i \qquad i = 1, 2, \ldots, 6 \tag{3.14}$$

These are the final equations for point kinetics.

4. Model

This section will explain the different components and options we choose for the model. In addition we will explain the different helpful options for working with Dymola. For extra tips about working with Dymola refeer to the appendix 7.

4.1 Pipes and Flow Model

Pipes are the fundamental component in Modelica and TRANSFORM for building hydrodynamic models. Most components are created on the basis of pipes or volumes. Realistic flows of water are complex and difficult to simulate. Because of this we make certain simplifications that don't compromise too heavily on the accuracy of the simulation. The pipes used in the model are native pipes to the TRANFORM library. They use a one dimensional Finite Volume Method flow of water with mass, momentum and energy partial differential equations. The medium parameters were set to follow the IF97 standard [9] and were explicit in pressure and enthalpy. Trace substances and heat dispersion along the circuit weren't considered although the possibility to do that should be theoretically available. Simple mass flow pumps were used to drive the flow of water at a steady rate. These pumps ignore cavitation and other issues (e.g. multi-dimensional flow characteristics) that do take place in real pumps. The use of more complex pumps caused numerical instability and are not the focus of this work.

4.1.1 Pipe Parameter Selection

Since in reactors there are pipes with many different shapes, sizes and properties we need to understand how select proper parameters to best express what happens in the reactor. When we mention pipe we usually mean the component named GenericPipe_-MultiTransferSurface, there are of course other pipes for different use cases that work more or less the same way with added components such as heat capacity. We can ignore heat capacity of the pipe because compared to the heat capacity of the circulating water it is small and is something that can easily be added later when a working model is achieved. Adding the pipe component to the diagram it will have an arrow indicating flow direction and two ports. The ports will look different, one will have an outline of a circle around it and the other won't. These can be changed under the Advanced options using the exposedState setting. Further explanation of these can be found in the TRANSFORM library -> User's Guide -> Connectors. The pipe has many properties which can be changed.

First one being nParallel which allows us to basically have the same component in parallel. When changing this parameter, it is important remember that the flow coming in is then divided between the pipes or components. Next is Medium, which allows us to change the medium of the system. It is best to start off with “Water using the IF97 standard, explicit in p and h, recommended for most applications”, later one can add trace elements or change the medium as needed. Going on down the list we will see Geometry which is a drop down menu with many options. The generic options are mostly partial models and require the user to complete the equations themselves. In the beginning it is easier to use StraightPipe option. After choosing the option next to the dropdown arrow there is a gray box. Clicking on the box brings up a submenu with multiple options. While this box is small and easily missable, it is important to define the actual parameters for the geometry of the pipe. Other components will have similar unnoticeable gray boxes so it is important to pay attention to these. Within the Geometry submenu the first option that one might need to change is nV. This is the number of volume nodes in the pipe, one can imagine the pipe being cut into multiple volumes on each volume calculations are made, the more volumes we have the more “accurate” the pipe simulation will be at the cost of simulation runtime. Functionally this is the resolution of the pipe. Depending on different uses the number required will change. For example, heating a pipe and having the water turn into vapor at some point might require more nodes, more accuracy. For simple flow 4 volume nodes will be enough. The nSurfaces setting can be kept at 1 for most uses. The other primary settings to change are dimension, length, and angle. Dimension assumes that the pipe is circular if one has a different pipe they might have to change the crossArea parameter instead. It is a good idea to familiarize oneself with the concepts of wetted perimeters and hydraulic diameter. The angle parameter is very important to pay attention to as this dictates the direction of the flow. -90 degrees corresponds to a downwards pipe while 90 degrees is upward. The mistake of forgetting to change this parameter can be made easily upon copying pipes, changing their directions, and then receiving anomalous results. With this we can close the submenu and move on to the flow model. Most of the times we want to leave this on default unless it is certain that a different flow model is necessary. In the submenu minor loss coefficients can be added but are not necessary and can be left default. Heat transfer options are quite self-explanatory. Turning use_HeatTransfer option to true adds a red connection node to the pipe in diagram view. Different heat sources can now be connected to the pipe. In the settings there is also the option to change the coefficient of heat transfer to different values but for simple models this can be left as Ideal in the beginning. We also have the option for InternalHeatGen if we want to for example, simulate the heat produced by a reactor core. The GenericHeatGeneration option is sufficient for this. The submenu allows to add per volume/node heat generation. If one knows the heat generation of the whole reactor a simple trick is to divide the value by number of nodes using [Name of pipe].geometry.nV. A useful tip for navigating the different variables is to press Ctrl+Spacebar to bring up a menu of paths for different components and commands, this can only be accessed if there is no text after the cursor. We also have the same options for trace substances as we did for heat, which work in a similar manner. Trace components are outside the scope of this thesis. Next tab we take a look at is the Advanced tab. Here many features can be found like the aforementioned exposedState option. Most noteworthy category is Dynamics. Each of the submenus will have the same four options. The detailed description of the different options can be found here [10]. In general, we use the FixedInitial option. If we know our system is in steady-state and have good steady-state parameters we can use SteadyState. To find SteadyState parameters we can initialize the system using FixedInitial or DynamicFreeInitial and seeing at what values the system is in a steady state and then changing the parameters to those. Although it sounds simple it often isn't and requires a lot of tweaking to initialize properly especially with components like the heat exchanger. In the initialization tab is where the user can input the initial values of the simulations. All options will have an a and b point. The rest of the intermediate values will be by default calculated as a linear spacing between those values. It is important to note the units which can be changed by clicking on them. If the temperature option isn't available, and enthalpy isn't known, then the enthalpy can be found by using [Name of component].Medium.specificEnthalpy pT(p,T). This pressure (p) and temperature (T) need to be in Pascals and Kelvins respectively. One can find a function for conversions in Modelica.Units.Conversions. The Species Mass Fraction and Trace Substances sections can be ignored.

4.2 Pressurizer

One of the most important parts of the pressurized water reactor is the pressurizer. This component keeps the water pressure at a pre-determined high enough level so it doesn't boil and evaporate in the core, ensuring stable single-phase heat transfer from the core. The component has 3 ports: 1 connected to the main loop, 1 for the spray systems, and 1 for the pressure relief valve. There are also options to add heating to the vapor and fluid, with additional heat dispersion options. Only the fluid heating system was used in this model. The spray and pressure relief systems were driven using a proportional–integral–derivative (PID) controller and the heating was controlled by a hysteresis component, which was used to keep the pressure between certain values so the heater isn't constantly on. The balance equations used in the pressurizer were split into two, one for vapor and the other for liquid. The mass balance equations are

$$\dot{m}_{liquid} = \dot{m}_{cond} - \dot{m}_{evap} + \dot{m}_{surge} + \dot{m}_{spray} + \dot{m}_{cSpray} + \dot{m}_{wl} \tag{4.1}$$

and

$$\dot{m}_{vapor} = \dot{m}_{evap} - \dot{m}_{cond} + \dot{m}_{steam} - \dot{m}_{cSpray} + \dot{m}_{vapSpray} - \dot{m}_{wl} \tag{4.2}$$

$\dot{m}_{surge}$, $\dot{m}_{steam}$, and $\dot{m}_{spray}$ are the values of mass flow rates from the three ports. $\dot{m}_{cond}$ and $\dot{m}_{evap}$ are the mass flow rates of condensation and evaporation respectively. $\dot{m}_{cSpray}$ and $\dot{m}_{vapSpray}$ are the condensate and vaporization mass flow rates of when water is injected into the spray port. These values are calculated differently based on the enthalpy of the spray compared to the vapor and liquid inside the tank. $\dot{m}_{wl}$ is the flow rate of vapor into liquid at the interface of the two phases. In Modelica code it looks like this:

$$\begin{multline} mb\_flow\_liquid = W\_cBulk - W\_eBulk + W\_surge \\ + W\_liqSpray + W\_cSpray + W\_vl, \tag{4.3} \end{multline}$$

and

$$\begin{multline} mb\_flow\_vapor = W\_eBulk - W\_cBulk + W\_steam \\ - W\_cSpray + W\_vapSpray - W\_vl. \tag{4.4} \end{multline}$$

The third port value $\dot{m}_{spray}$ is renamed to $W\_liqSpray$ since is value changes depending on the enthalpy of the spray along with $\dot{m}_{cSpray}$ and $\dot{m}_{vapSpray}$ in the following ways. If the enthalpy of the spray H_{spray} is smaller or equal to the enthalpy of the saturated fluid H_{fsat} then the equations are formulated in the following way:

$$\dot{m}_{cSpray} = \dot{m}_{spray} * (\frac{H_{fsat} - H_{spray}}{H_{vapor} - H_{fsat}}), \tag{4.5}$$

$$H_{cSpray} = \dot{m}_{cSpray} * H_{fsat}, \tag{4.6}$$

$$\dot{m}_{vapSpray} = 0, \tag{4.7}$$

$$H_{vapSpray} = 0, \tag{4.8}$$

$$\dot{m}_{liqSpray} = \dot{m}_{spray}, \tag{4.9}$$

$$H_{liqSpray} = \dot{m}_{spray} * H_{fsat}. \tag{4.10}$$

$$\tag{4.11}$$

If H_{spray} is greater or equal to the enthalpy of saturated gas H_{gsat} then the following equations are used:

$$\dot{m}_{cSpray} = 0, \tag{4.12}$$
$$H_{cSpray} = 0, \tag{4.13}$$
$$\dot{m}_{vapSpray} = \dot{m}_{spray}, \tag{4.14}$$
$$H_{vapSpray} = \dot{m}_{vapSpray} * H_{spray}, \tag{4.15}$$
$$\dot{m}_{liqSpray} = 0, \tag{4.16}$$
$$H_{liqSpray} = 0. \tag{4.17}$$
$$\tag{4.18}$$

When the value of H_{spray} is in between H_{fsat} and H_{gsat}, the final set of equations is used:

$$\dot{m}_{cSpray} = 0, \tag{4.19}$$
$$H_{cSpray} = 0, \tag{4.20}$$
$$\dot{m}_{vapSpray} = \dot{m}_{spray} * X_{spray}, \tag{4.21}$$
$$H_{vapSpray} = \dot{m}_{vapSpray} * H_{gsat}, \tag{4.22}$$
$$\dot{m}_{liqSpray} = \dot{m}_{spray} - \dot{m}_{vapSpray}, \tag{4.23}$$
$$H_{liqSpray} = \dot{m}_{liqSpray} * H_{fsat}. \tag{4.24}$$
$$\tag{4.25}$$

Here X_{spray} is introduced which is the thermodynamic quality of the steam from the spray port and is defined as follows:

$$X_{spray} = \frac{H_{liquid} - H_{fsat}}{H_{gsat} - H_{fsat}}. \tag{4.26}$$

In addition to mass balance we also have to consider the energy balance for which the liquid part of the equations are:

$$\begin{aligned}\dot{H}_{liquid} = \dot{m}_{cond} * H_{fsat} - \dot{m}_{evap} * H_{gsat} \\ + \dot{m}_{surge} * H_{surge} \\ + \dot{H}_{liqSpray} + \dot{H}_{cSpray} \\ + \dot{m}_{vl} * H_{gsat},\end{aligned} \tag{4.27}$$
$$Q_{liquid} = Q_{lHeater} + Q_{vl} + Q_{wl}, \tag{4.28}$$
$$W_{liquid} = -p * \frac{d}{dV_{liquid}}, \tag{4.29}$$

and for vapor:

$$\begin{aligned}\dot{H}_{vapor} = \dot{m}_{evap} * H_{gsat} - \dot{m}_{cond} * H_{fsat} \\ + \dot{m}_{steam} * H_{steam} \\ - \dot{H}_{cSpray} + \dot{H}_{vapSpray} \\ - \dot{m}_{vl} * H_{gsat},\end{aligned} \tag{4.30}$$

$$Q_{vapor} = Q_{vHeater} - Q_{vl} + Q_{wv}, \tag{4.31}$$

$$W_{vapor} = -p * \frac{d}{dV_{vapor}}. \tag{4.32}$$

The first equation $\dot{H}$ denotes the enthalpy flow rate in which $\dot{m}$ is for mass flow and H is enthalpy of the liquid or vapor. The second equation Q is for heat flow. $lHeater$ and $vHeater$ are the values of a connected heater component that by default is 0 if not defined for liquid and vapor. Q_{wl} and Q_{wv} are the liquid and vapor heat transfer values with the wall or ambient temperature if wall heat capacity is unspecified. Q_{vl} is for heat transfer between liquid and vapor with positive direction into the liquid. The third equation W is for work done by pressure on the liquid or vapor. In the Modelica language the equations take the following forms:

$$\begin{aligned}Hb\_flow\_liquid = W\_cBulk * h\_fsat - W\_eBulk * h\_gsat \\ + W\_surge * h\_surge \\ + H\_liqSpray + H\_cSpray \\ + W\_vl * h\_gsat,\end{aligned} \tag{4.33}$$

$$Qb\_flow\_liquid = liquidHeater.Q\_flow + Q\_vl + Q\_wl, \tag{4.34}$$

$$Wb\_flow\_liquid = -p * der(V\_liquid), \tag{4.35}$$

for vapor they take the form:

$$\begin{aligned}Hb\_flow\_vapor = W\_eBulk * h\_gsat - W\_cBulk * h\_fsat \\ + W\_steam * h\_steam \\ - H\_cSpray + H\_vapSpray \\ - W\_vl * h\_gsat,\end{aligned} \tag{4.36}$$

$$Qb\_flow\_vapor = vaporHeater.Q\_flow - Q\_vl + Q\_wv, \tag{4.37}$$

$$Wb\_flow\_vapor = -p * der(V\_vapor). \tag{4.38}$$

Depending on the selected energy dynamics the complete energy balance equation is formed differently. If steady state option is selected the equations will be formed as:

$$0 = \dot{H}_{liquid} + Q_{liquid} + W_{liquid}, \tag{4.39}$$

$$0 = \dot{H}_{vapor} + Q_{vapor} + W_{vapor}. \tag{4.40}$$

Otherwise the following equation set is used:

$$\frac{d}{d\,U_{liquid}} = \dot{H}_{liquid} + Q_{liquid} + W_{liquid}, \tag{4.41}$$

$$\frac{d}{d\,U_{vapor}} = \dot{H}_{vapor} + Q_{vapor} + W_{vapor}, \tag{4.42}$$

where U designates the internal energy of the liquid or vapor. These are the equations of the TRANSFORM default pressurizer. They actually have slight issue in that they overestimate the energy going into the fluid which instead of a pressure decrease while pumping in cold water result in an increase instead. This problem seems to have been faced by others using the pressurizer component as well to the point where the decision to model their own pressurizer was taken [11]. The main issue seemed to be in the assumption that when the spray reaches the fluid that it has reached saturation. In the thesis model this assumption was changed to the specific enthalpy of the spray, which is not physically rigorous but a numerical approximation that ensures model stability. In the future an analytical solution should be attempted if the component will be used for real experiments.

4.2.1 Pressurizer Parameter Selection

The first thing to consider when selecting parameter values for the pressurizer is the shape of the pressurizer which can be selected from DrumType. The height and radius of the component can be set in the submenu. If one has selected the Traditional drum type then the V_total can be set to

```
0.5*4/3*Modelica.Constants.pi*(drum2Phase.drumType.r_1)^3
+0.5*4/3*Modelica.Constants.pi*(drum2Phase.drumType.r_3)^3
+drum2Phase.drumType.h_2
*Modelica.Constants.pi*(drum2Phase.drumType.r_2)^2
```

where drum2Phase can be set to the name of the component. All the closure models can be set to the Constant form. For BulkEvaporation the time was set to 15 seconds. This is a time delay for the mass flow of the evaporation. BulkCondensation was set to 15 seconds. Same as before but for condensation. MassTransfer_VL was set to $1\text{e}-3\,\frac{kg}{s\,m^2\,K}$. This parameter describes the amount of mass transported between the vapor-liquid surface.

HeatTransfer_VL was set to 100 W. This describes the amount of heat transfered between the vapor-liquid surface. Since in this model no heat transfer between wall was defined the last two options were set to 0. In the initialization tab Vfrac_liquid_start can be set to 0.5 meaing that the pressurizer is half full and p_start to the pressure of the core.

4.3 Core and Fuel model

An important part of any reactor model is the core and fuel. This is what defines our power and heat outputs. Different core arrangements and sizes are possible and some can be found online. In this thesis, a large PWR core was defined according to the values provided by MIT [12].

The core model uses a popular and efficient point kinetics based model for neutron population calculations. The energy balance equation for the core is:

$$\frac{\partial Q_f}{\partial t} = \frac{(\rho - \beta)}{\Lambda Q_f} + \sum_{i=1}^{j} \lambda_i C_i + \frac{Q_{ext}}{\Lambda}, \tag{4.43}$$

In this equation Q_f is the amount of energy generated by fission reactions, Q_{ext} is additional added external energy and j is the amount of precursor groups selected. In our model this was taken to be 6 groups and no source of external energy was added $Q_{ext} = 0$. In code the equation looks like:

```
der(Q_fission) =(rho - Beta)/Lambda*Q_fission
                + sum(lambdas .* Cs[:])
                + Q_external/Lambda;
```

The values relating to delayed neutrons and precursor groups are taken from TRANSFORM library 6 group precursor data set. The value of total reactivity rho is given by the following equation:

$$\rho = \rho_{input} + \rho_{mod} + \rho_{dop} + \rho_{fis} \tag{4.44}$$

The reactivity from fuel ρ_{dop} and moderator ρ_{mod} were considered using Doppler and moderator feedback coefficients, which can be found in Table 1. Control rod reactivity ρ_{input} was adjusted using a PID-controller, which was simplified to a positive or negative insertion of reactivity to the core by using the rho_input parameter. The thermodynamic effects of the control rod were not considered only it's impact on the reactivity. ρ_{fis} is calculated by using:

$$\rho_{fis} = \sum_{i=1}^{j} \frac{-\sigma_A(i) N_{Ci}(i)\, V}{\overline{\nu}\Sigma_F} \tag{4.45}$$

In this equation the sum is over all the delayed precursor groups. σ_A is the microscopic cross-section for absorption and Σ_F is the macroscopic cross-section for fission reactions. $\overline{\nu}$ is the amount of neutrons released per fission reaction. N_{Ci} is the amount of fission product atoms. In code the two equations look like:

```
rho = rho_input + sum(rhos_feedback[:])
    + (if toggle_ReactivityFP then -sum(fissionProducts.rhos_start)
    + sum(fissionProducts.rhos[:]) else 0)
    + sum(fissionProducts.rhos_add[:]);
```

and

```
    rhos[j] =-sigmasA[j]*mCs[j]/(nu_bar*SigmaF)/V;
```

In addition to these we have decay heat so our final equation for the total power produced is:

$$Q_{total} = Q_f + \sum_{i=1}^{k} Q_{decay}(i). \tag{4.46}$$

Decay heat was considered in 11 groups $k = 11$.

Fuel pins are modelled as 3 regions: fissile material (fuel), gap, and cladding. Heat transfer is considered only in radial direction. The number of radial nodes for each section was 3 and axial direction was 4. The fuel pellet was from Uranium dioxide, the gap was taken to be Helium, and the cladding a Zirconium alloy ZrNb_E125. The inlet temperature was held constant using a PID-controller that adjusted the flow rate of the water in the cooling loop. Heat transfer effects between moderator and gap were calculated using laminar and turbulent Nusselt numbers. Laminar Nusselt number was taken to be 4.36 and turbulent number was calculated using the Dittus–Boelter equation, these are default values for Nusselt numbers in TRANSFORM.

Table 1. *Core parameters*

Parameter	**Value**
No. Fuel Assemblies	193
No. Fuel Rods in Assembly	264
Lattice Configuration	17 x 17
Active Fuel Length	365.76 cm
Assembly Lattice Pitch	21.50364 cm
Pin Lattice Pitch	1.25984 cm
Total Core Power	3411 MW
Core Operating Pressure	156 bar
Core Flow Rate	17083 m/s
Core Inlet Temp	294.22 °C
Doppler Feedback Coef.	-2.5e-5 1/K
Moderator Feedback Coef.	-20e-5 1/K
Fuel Reference Temp	705.3 °C
Coolant Reference Temp	314.242 °C

4.3.1 Core and Fuel Model Parameter Selection

Before starting to parameterize the core component it is a good idea to set the input temperature into the core to be constant. In a test scenario with the core separated this can be easily done but in a loop test case this should be achieved with a PID-controller that controls the amount of cooling happening in the cold leg of the loop. One has the option of cooling the loop by using a heat exchanger and controlling the flow speed of the second loop or by cooling a pipe with heat flow ports and having the PID control the heat flow. In this model the speed of the simple pump on the secondary side was controlled using a PID type contoller with an initial output of 3000, k=1, k_s = 1, k_m = 1, Ti = 0.5 sec, Td = 0.1 sec, wp = 1, wd = 0, Ni = 0.9, and Nd = 10. To control the cooling by using a pipe with heat flow enable and a Heatflow_multi boundary the PID would need to be configured with different settings and might also require that the inputs, u_s and u_m, are switched. There are three types of cores in the TRANSFORM library, 1-,2-, and 3-region cores. The regions refer to the fuel, gap and cladding. So depending on the resolution required the choice can be made. This model used the 3-region version. The core component is actually one assembly that is where the "subchannel" in the name comes from. So in the parameters window nParallel is the total amount of assemblies in the whole reactor. Most of the parameters within geometry are self-explanatory, it is important to pay attention to angle again. For better accuracy the HeatTransfer can be set to “Specify Nus

2 region laminar and turbulent". In the Kinetics tab the Data_PG can be set to "Default delayed-neutron groups from TRACE Manual Table 9-1". Other values and parameters can be adjusted based on the reactor one is trying to model but first time it's recommended to model using the MIT data [12]. When using this data set the Teffref_fuel to 705.3 C and Teffref_coolant to 314.242 C. Fuel element initialization will depend on the reactor one is trying to model and the amount of negative rho_input one is starting with, which will be explained later. These values can also be found by waiting until the reactor is at steady state. If one has trouble running the simulation due to large spikes the size of the pressurizer can be set to very large e.g 100m height and then slowly reduced to more normal values as better core parameters are found by letting the system stabilize. In coolant initialization it is a good idea to initialize inlet and outlet at their respective temperatures. Inlet temperature can be found in specification sheets and outlet temp can be calculated based on core power and flow rate. It might be helpful to give steady state temperature values for each node as an array in Ts_start. Steady state initialization is not necessarily a good thing as this usually means that the solution to the system is known. Usually it is more important that the system converges to a steady state. This usually can be achieved by parameter calibration. If one wishes to start from steady state they can set the energy and mass balance of the coolant to SteadyState. This can be quite difficult to achieve however. KineticDynamics and FissionProductDynamics can be set to FixedInitial. One of the more important values to set is at the bottom of the General tab rho_input. Each reactor has some negative rho_input value that they are operating under coming from the boron in the moderator or the control rods. Additionally, there is some negative rho from the design of the reactor that the core model doesn't take into account. For this it is best to use a PID controller. The inputs are the nominal and Q_total.y values for the coresubchannel. One has to carefully tune the controller to achieve reasonable results, if the controller is too fast or too slow it can make finding the stable value impossible. This requires for some experimentation on the users' behalf. It is also a good idea to set a maximum and minimum output, within reasonable limits, when the stable values have been found. In this model the controller was set to PI type, k_s and k_m were set to 1/coreSubchannel.Q_nominal, yMax to 0.012. Under tuning controls k = 5e-4, Ti = 0.1, wp = 1 and Ni = 0.9. Under initialization one should set the controller to have an initial output otherwise they will see large jumps and spikes in power in the beginning potentially causing the system to reach undesirably high pressures and temperatures. In this model the initial output was set to -0.009. This was determined empirically by looking at the initial rho and Q_total values and adjusting the input so they are closest to steady state values.

4.4 Heat Exchanger Model

Each reactor has some way of dissipating heat from the primary loop. Usually this is done by way of heat exchangers. In PWR the heat exchanger is also known as the steam generator since the water is turned into steam in the secondary loop. TRANSFORM offers multiple different types of heat exchangers to choose from. This model uses a shell and tube type of heat exchanger. It is basically a number of multiple tubes coming from the reactor that are directed in to a large vessel with flowing water. The thermodynamic effect of the wall was also modelled in the heat exchanger. The material of the wall was taken to be Inconel 690. Thermal transfer coefficients in the primary circuit side were calculated using single-phase Nusselt numbers and Dittus-Boelter equation. The secondary side was modelled using two-phase three-region alpha coefficients. This means that thermal coefficients were calculated separately for single-phase liquid, single-phase vapor, and two-phase saturated liquid, since the water turns into steam and under goes phase changes while going through the component. For single-phase liquid and vapor turbulent part Nusselt numbers were used with Dittus-Boelter equations. For the two-phase saturated liquid Chen's correlation was used.

4.4.1 Heat Exchanger Model Parameter Selection

The heat exchanger (HX) component is probably the hardest to integrate into one's model, due to it causing issues with all other parts. Depending on the reactor type different geometries can be selected for the heat exchanger. For a pressurized water reactor, we used the shell and tube HX. One has to be careful not to select GenericHX since it is a partial model and realistically shouldn't be in the options, this is mainly used if there is a need to build a different kind of heat exchanger using some established base. In the geometry submenu, we used 20 volume nodes. This is taken from one of the examples included and is most likely to do with the phase change in the second loop. Having less nodes and a phase change can lead to issues in the simulation. Most of the values for the heat exchanger can be taken from the NSSS example reactor which is included in the TRANSFORM library under Examples/LightWaterReactor_PWR_Westinghouse. A few other things to point out are heat transfer models and shell initialization. For heat transfer on the shell side it is a good idea to use two-phase 3-region alphas model, because we have a phase change in the HX component meaning liquid turns into steam. So we need to tell the model that it has to be two-phase also the different phases have differing thermal diffusivity, alpha, so we need to calculate them in 3 regions them being single-phase liquid, single-phase vapor, and two-phase saturated liquid. For the tube side single-phase 2-region Nus can be used. If there is a need 5-regions can also be simulated adding turbulent and laminar regions for liquid and vapor, in this model such accuracy wasn't necessary. Since we know the primary circuit should be liquid we can use a single-phase model here and the 2 regions we calculate the Nus, Nusselt numbers, for are the laminar and turbulent regions of the liquid. It is important to use two-phase models if one has both vapor and liquid. The other initialization tip would be to use enthalpy on the shell side for initialization and then to initialize the medium at vapor saturated vapor enthalpy or a little bit less than that if it has trouble initializing at that value. The rest of the heat exchanger works similarly to regular pipes. If one is having issues with the shell and tube type then it a good idea to make sure if the pipes on the tube side can actually fit in the shell and that they don't cause too much flow constriction. An important factor to consider is when testing the heat dissipation of primary loop one has to be careful when setting static values as this can lead to cooling or heating over long periods of time. Thus it is best to use a PID or some sort of dynamic cooling to adjust the rates.

5. Results

The final model can be seen in figure 3 consists of 21 components excluding sources and on compilation has 15883 equations. The solver used for these simulations was Esdirk45a.

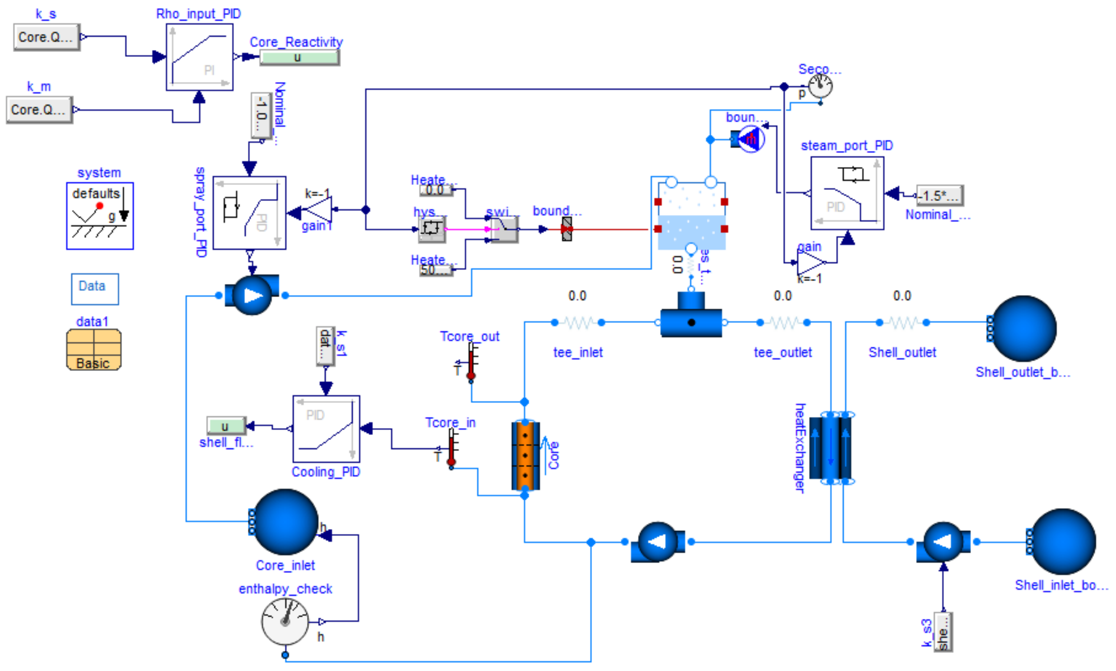

Figure 3. *Component diagram of reactor model*

On initialization one numerical Jacobian and three nonlinear systems are present. For faster computations these should be minimized or removed. As of writing this paper the source of the nonliearities and Jacobian are unknown but most likely the heat exchanger component is the one that should be looked at. A 1000000 second simulation runs for 71.719 seconds on a mid-tier laptop. Most of the time is spent in the first 5000 seconds which takes 62.269 seconds, when the system has yet to reach steady state.

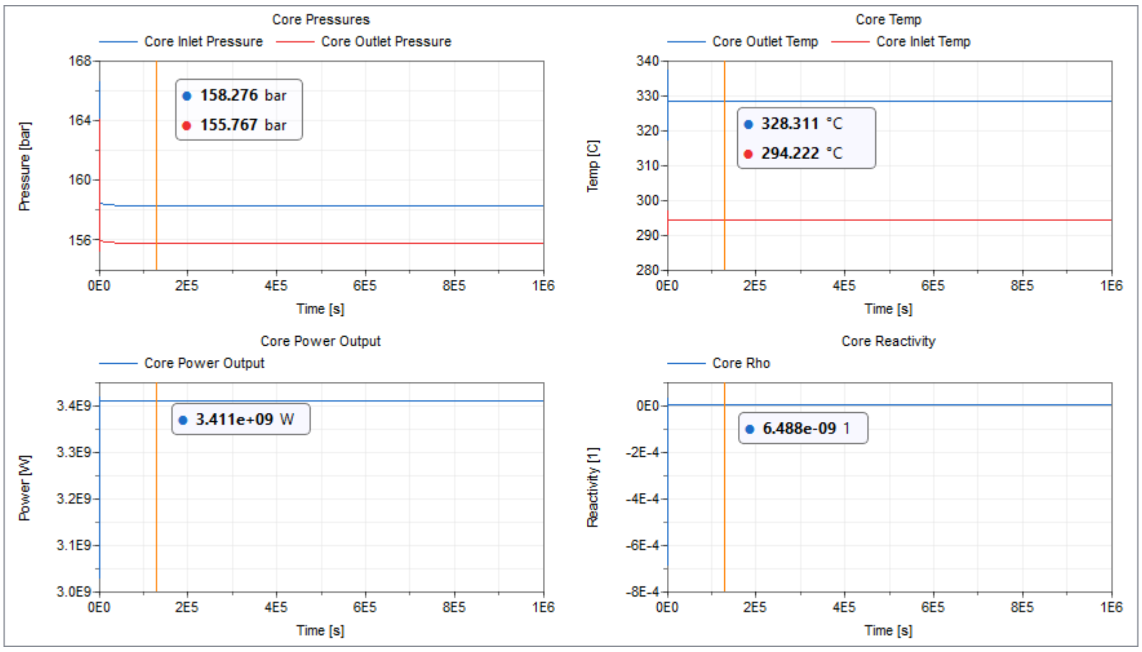

Figure 4. *Results of base model. The line represents the measurement point of steady state values which can be seen in the box next to it*

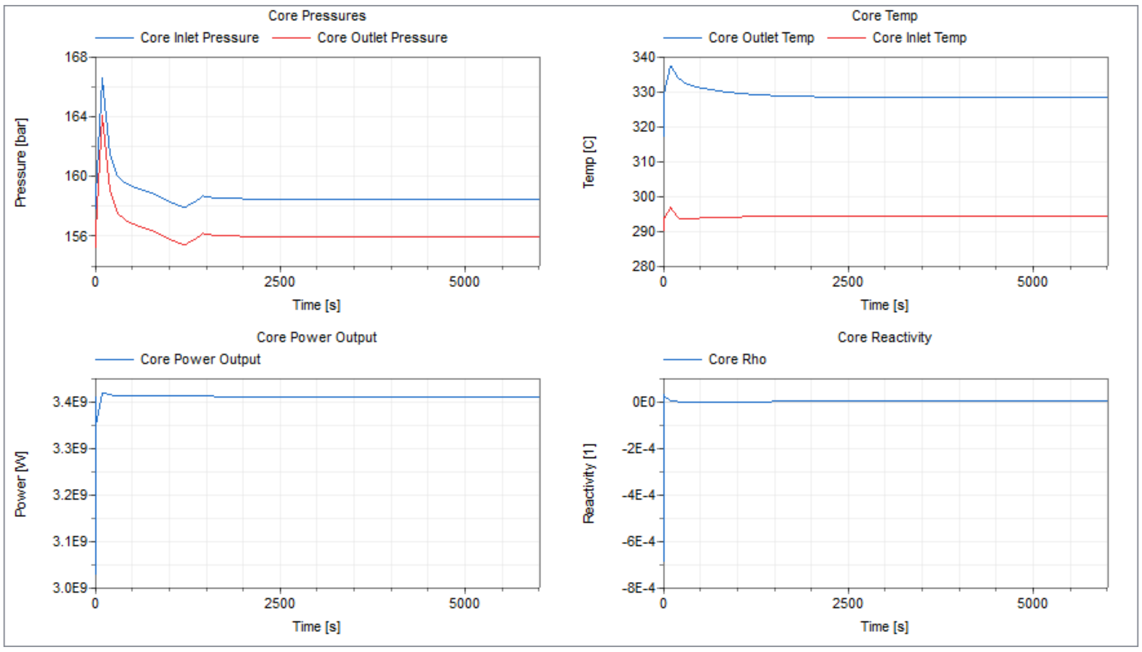

Figure 5. *Zoomed in results of base model*

As seen in figure 5 it takes about 4000 sec in order to converge to a steady state. In figure 4, it can be seen that a completely steady state is achieved at around 70000 s. Without the heat exchanger component, figure 6, it is also possible to start directly from steady state.

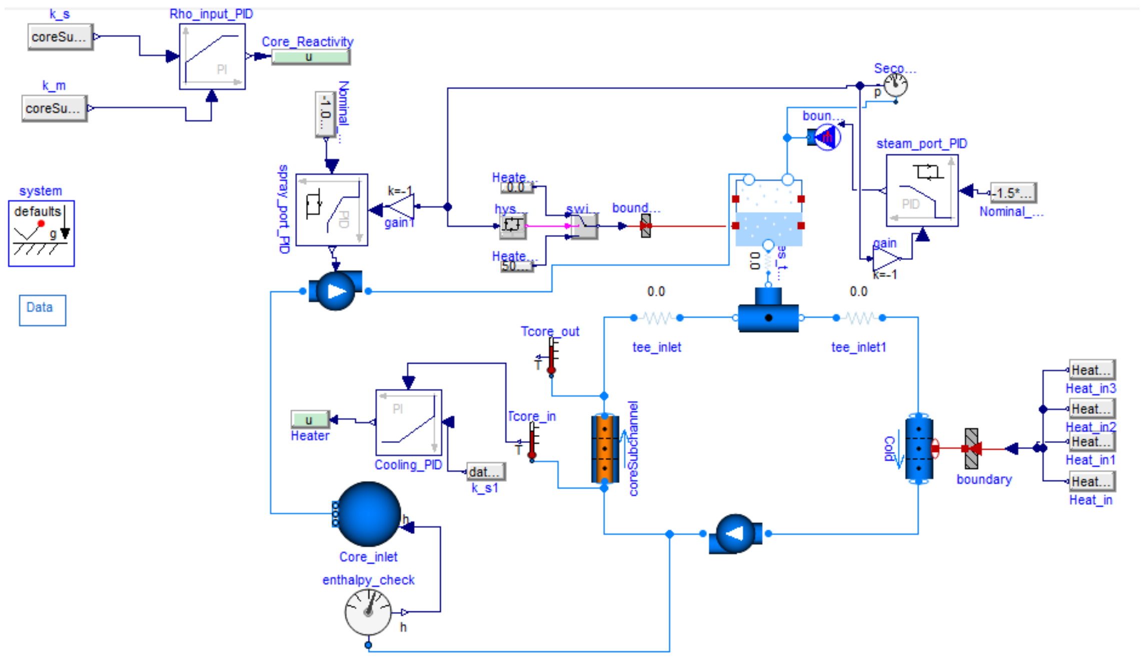

Figure 6. *Component diagram of reactor model without heat exchanger*

We can do a simple calculation for how much we can expect the water to heat up when it passes through the core using the equation:

$$Q = \dot{m}\, C_p\, \Delta T, \tag{5.1}$$

which we can convert to get temperature delta between core inlet and outlet

$$\Delta T = \frac{Q}{\dot{m}\, C_p}. \tag{5.2}$$

We can use the values provided in Table 1 and take inlet pressure to be 158 bar from simulation and calculate $C_p = 5.318 \frac{\mathrm{kJ}}{\mathrm{kg\,K}}$. We get the value of

$$\Delta T_c = 37.546\,°\mathrm{C}. \tag{5.3}$$

Comparing this to our simulation where we have

$$\Delta T_s \approx 35\,°\mathrm{C}, \tag{5.4}$$

we get a difference of

$$\Delta T_c - \Delta T_s = 2.546\,°\mathrm{C}. \tag{5.5}$$

This difference while it may seem small is in quite large. This might be simply due to the naivety of our theoretical value since the heat capacity C_p of water changes as the water goes through the core. The core has 4 volume nodes meaning 4 separate calculations with C_p should be made and could give a more accurate theoretical value.

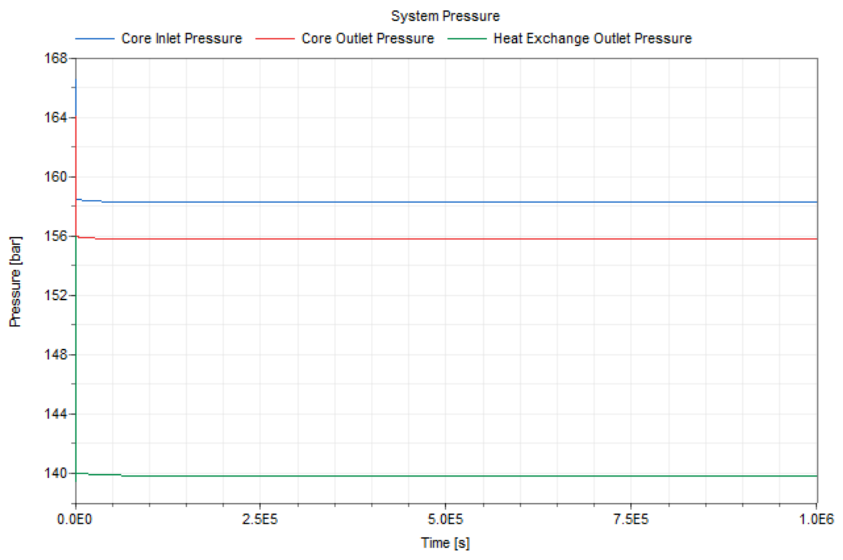

Figure 7. *Pressure across the primary loop*

It can also be interesting to look at the pressure drop across the system seen in figure 7. Since the tee for the pressurizer doesn't use any flow model and the resistance components between the core and the pressurizers are there for numerical stability, they are negligible and because of this the core outlet pressure and the heat exchanger inlet pressures are the same.

5.1 Transients

5.1.1 Core Inlet Temperature Change

The first test we ran was a 2% change in the inlet temperature of the core. This can happen if for some reason the water temperature or flow rate in the secondary loop side of heat exchanger changes. At first we waited until 7000 seconds for the system to reach more or less steady state. Then we dropped the target value for the PID that controls the temperature going into the core by 2%. The results can be seen in the charts below

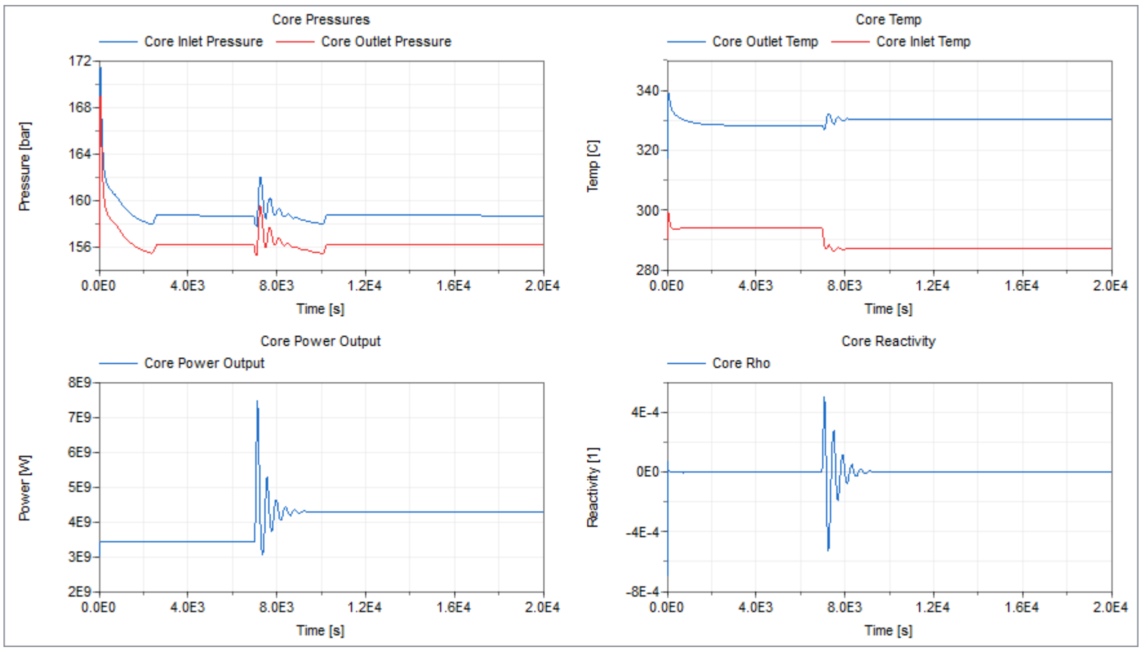

Figure 8. *Results of decreased core inlet temperature*

First thing we notice in figure 8 is a large power spike caused by a change in the reactivity ρ. This change in ρ is caused by a drop in water temperature. Since the water temperature is lower it decreases the energy of the neutrons that are reflected causing their fission cross-sections to be larger as we saw in Figure 1, resulting in an increase in the neutron population. Due to the release of more power from more fission reactions the outlet of the core is actually hotter than it was previously. This causes an increase in temperature that the PID systems on the pressurizer try to manage. After around 5000 seconds the core stabilizes at a new higher power output and larger temperature differential. This demonstrates that the model is able to capture the very important temperature-power feedbacks.

5.1.2 Loss of Secondary Loop

The second test we performed was a pump failure test in the secondary loop, which can happen if the secondary side pumps and backup pumps loose power. We wait till 10000 seconds have passed and then quickly ramp down the secondary pump speed. The simulation fails at 35823 seconds.

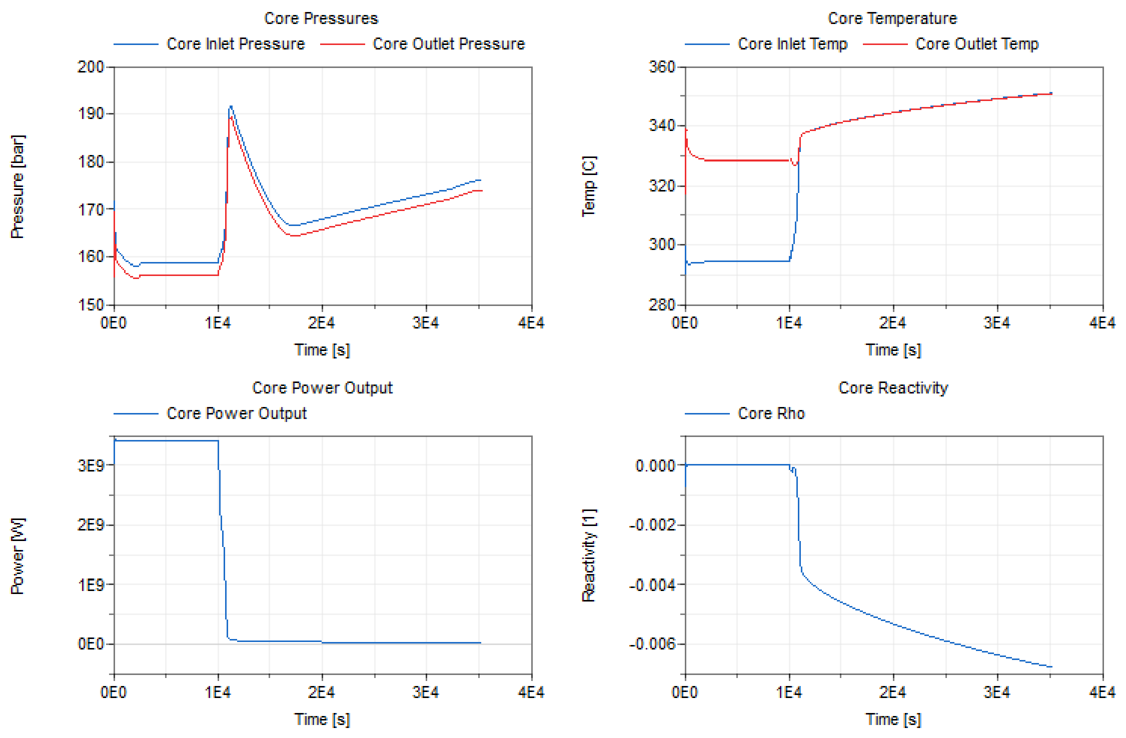

Figure 9. *Results of secondary loop pump failure*

Here in figure 9 we can see multiple things happening we see a sharp increase in core pressure followed by a slower decrease. We also see a sudden power drop and a drop in reactivity. Finally we see a sudden rise in temperature as we would expect when there is no cooling. The sudden loss in power can be explained by the sudden increase in temperature. Due to the increase in temperature of the moderator neutrons have a harder time reaching thermal energies, thus resonance escape probability decreases. The fuel also warms up causing Doppler broadening to increase further increasing resonance capture. Quite quickly there aren't enough neutrons being generated each generation and the core loses it's criticality and nuclear fission stops. From the chart we can still see that even as fission has stopped there is still some energy being generated. This is decay heat from the decay of different nuclear isotopes.

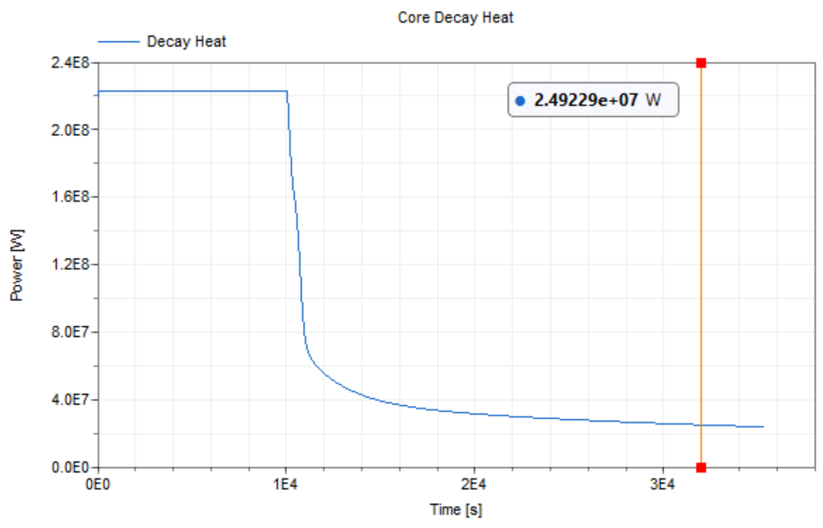

Figure 10. *Decay heat of secondary pump failure*

From the figure 10 we can see that the decay heat left over is actually quite large. Some isotopes decay in a matter of seconds while others decay for hundreds of years. We can also glean more information if we take a look at the pressurizer.

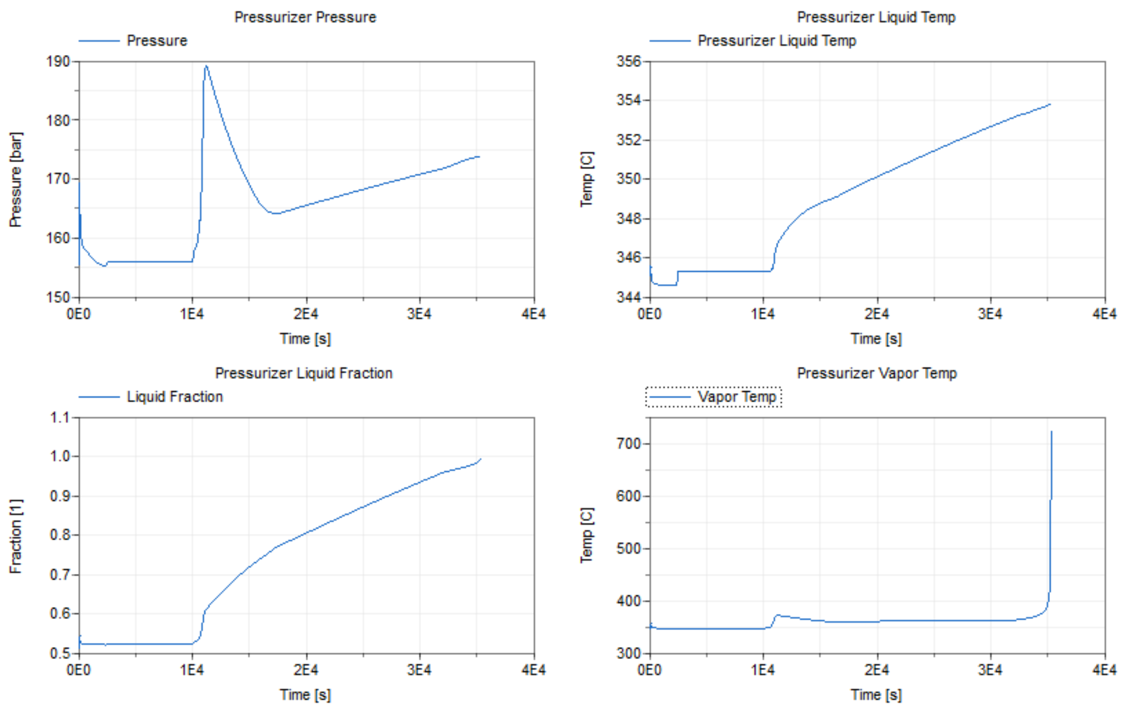

Figure 11. *Decay heat of Secondary pump failure*

In these graphs 11 we can see why the simulation fails. During the simulation the pressurizer tries to keep its designed pressure level but has a hard time doing so because of all the extra heat. It slowly fills up to try to reduce the pressure by spraying cold water but when there is no more room in the pressurizer and since the system can't overflow the simulation fails. The reason for the spike in pressure is due to a sudden spike in temperature in the cold side of the loop seen in the figure 9 where inlet and outlet temps become basically the same. The reduction in pressure happens because fission chain reaction stops in the core after that pressure keeps building due to decay heat.

5.1.3 Leak in Primary Circuit

The third test we performed was a leak test in the primary circuit of the reactor. Leaks can happen because of the harsh condition in the reactor pipes can become more brittle over time. A leak of 0.2 kg/s was introduced right before the heat exchanger at 10000 seconds into the simulation when the system was stable.

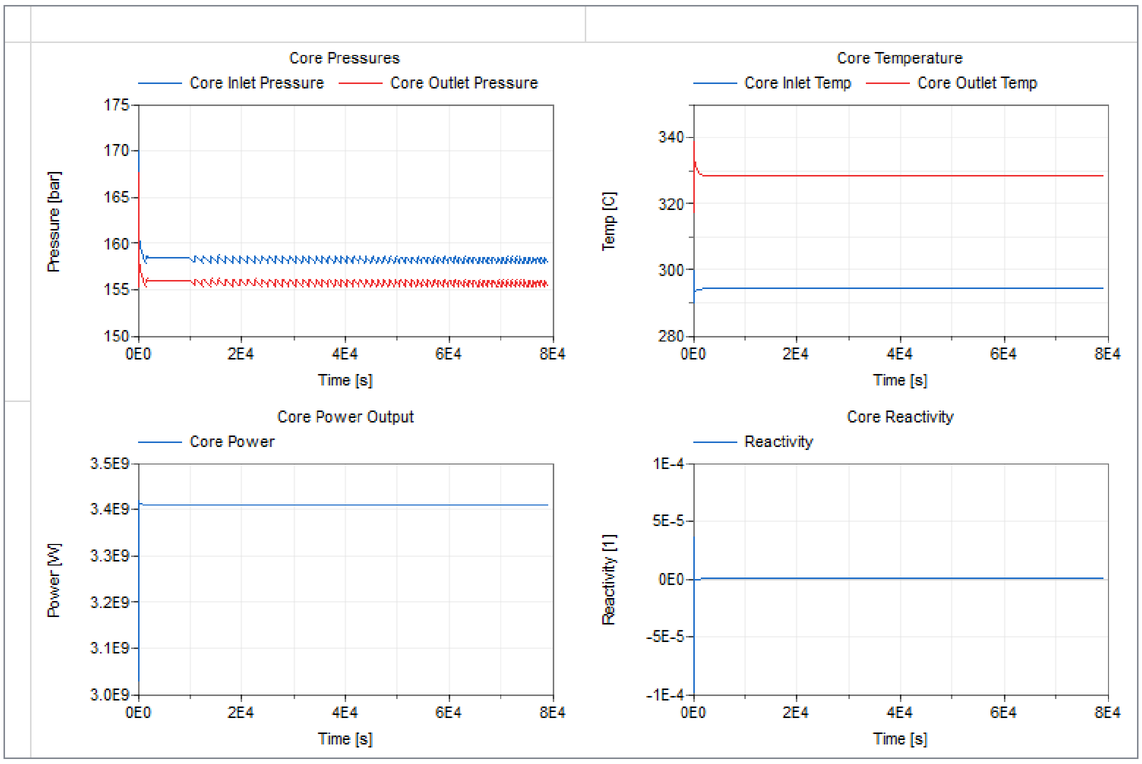

Figure 12. *Core results for the leak test*

From the core values in figure 12 we can see that it looks mostly normal except that the pressure is going up and down periodically and the simulation suddenly ends at 83500 seconds.

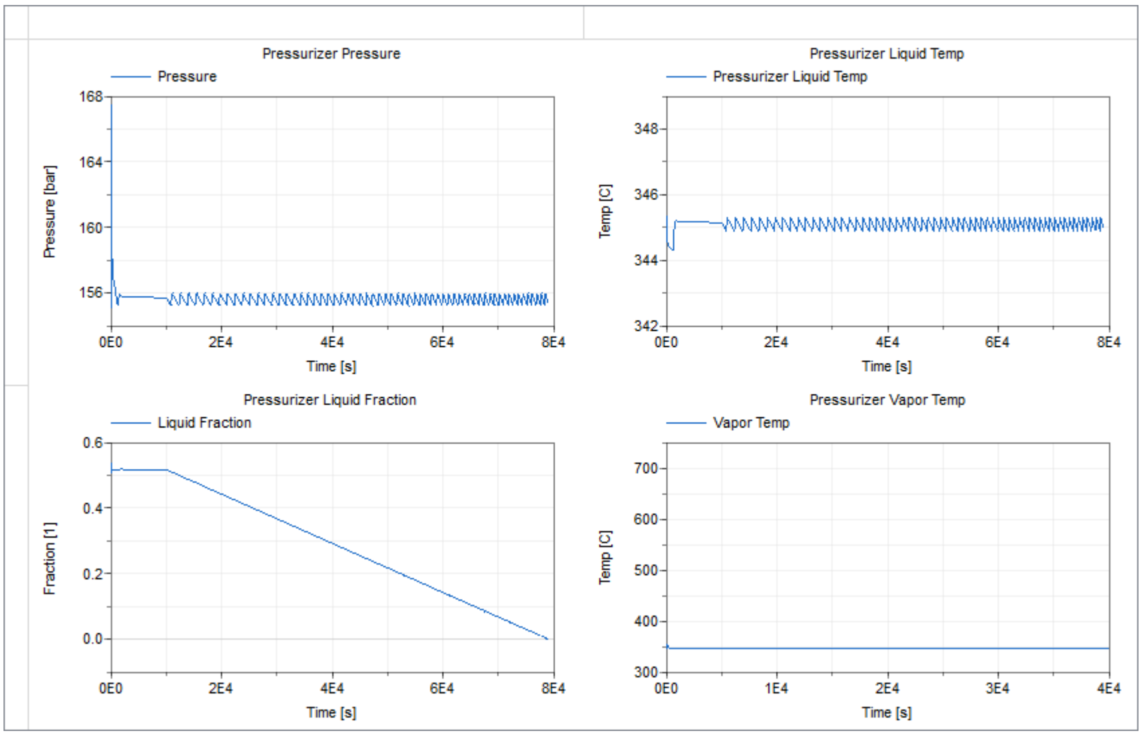

Figure 13. *Pressurizer results for the leak test*

Here in figure 13 we can see why the simulation ends, the pressurizer runs out of water. Here we see also the water temperature in the pressurizer doing the same periodic movement while in the rest of the loop it didn't show up including the vapor of the pressurizer. Using this lead we can take a look at what the liquid heater is doing.

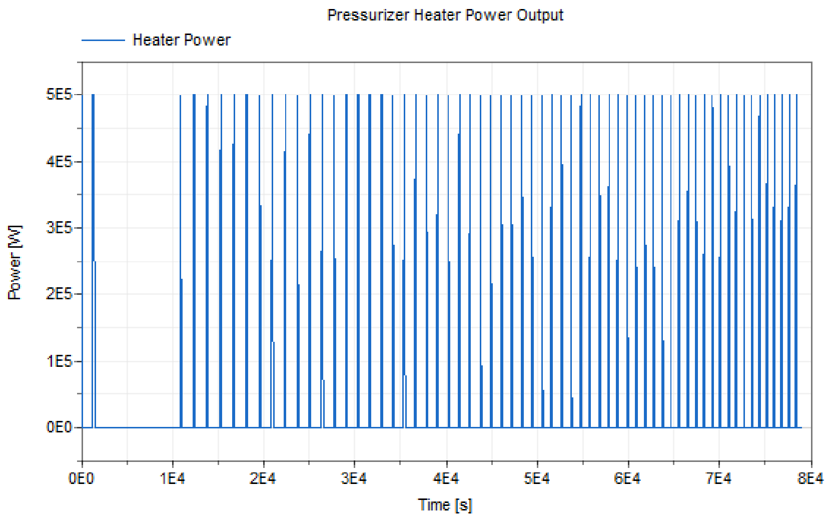

Figure 14. *Pressurizer heater values for the leak test*

This graph 14 gives us the reason for why we see this periodic jumping. It comes from the heater which is controlled by a PID to keep it above a certain threshold pressure level. The leak in the system causes a pressure drop which the heater then tries to compensate for by vaporizing more water and creating a pressure increase. In this case the system might have failed earlier in reality because the physical position and size of the heaters isn't modelled.

5.2 Future works

While this work serves as a first test to develop a full nuclear reactor simulator in a new modeling language, there are several areas of the modeling that call for improvements in the future. More detail should be added for pressure losses in the system due to friction from the piping. The pressurizer component also needs to be verified and validated more to make sure the numerical set up is as intended and the model represents also reality i.e. is valid. This work has provided first experience and feeling on how the model works and which parameters have which importance on the results, there is room for sensitivity analysis to understand the assumptions made for each component and how it affects the model as a whole. Whole another work direction is the BWR modeling. Currently there is no such component which can be used as the BWR core. More research is also required into the solvers and what short comings they have if the results are to be used to base real world decisions on. The solvers might have different idiosyncrasies' which might lead to having to change the model to fit the solver better. All in all, the model requires integral verification, validation and uncertainty quantification (VVUQ) before it can be used for any real nuclear engineering purpose.

6. Summary

A working preliminary model of a pressurized water reactor and the aim of the thesis were achieved. The model was developed using Modelica and Dymola, with the main library being TRANSFORM. The final reactor consisted of a nuclear core with a 3 region fuel model, a pressurizer and a heat exchanger. The model was shown to be able to reach steady state and 3 different test transients scenarios were also shown. In addition parameterization of different components were also shown and explained.

In conclusion developing a more complex model in Dymola using TRANSFORM seems promising but will require a lot of work to try and understand the different components on a more fundamental level so errors could be minimized, since TRANSFORM has minimal documentation. To develop a boiling water reactor further components will need to be developed since TRANSFORM natively doesn't have them. A better understanding of Modelica is also require especially on the front of solvers and what assumptions and short comings the make or have.

7. Acknowledgments

The author is grateful to his supervisor Marti Jeltsov for suggesting the thesis topic, as well as for his timely advice and support. The author is also thankful to everyone who has guided him along the way and given their input and wisdom.

References

[1] *Homepage of the Modelica Association.* Visited on 31/12/2023. URL: https://modelica.org/.

[2] *Dassault Systems Dymola homepage.* Visited on 31/12/2023. URL: https://www.3ds.com/products/catia/dymola.

[3] *Github link to TRANSFORM library.* Visited on 31/12/2023. URL: https://github.com/ORNL-Modelica/TRANSFORM-Library.

[4] *Nuclear Physics and Reactor Theory. DOE Fundamentals Handbook Volume 1.* Visited on 20/12/2023. 1993. URL: https://www.standards.doe.gov/standards-documents/1000/1019-bhdbk-1993-v1/@@images/file/.

[5] *Nuclear Physics and Reactor Theory. DOE Fundamentals Handbook Volume 2.* Visited on 20/12/2023. 1993. URL: https://www.standards.doe.gov/standards-documents/1000/1019-bhdbk-1993-v2/@@images/file/.

[6] *OECD NEA Janis database.* Visited on 18/12/2023. URL: https://www.oecd-nea.org/jcms/pl_39910/janis.

[7] *Factor dependance on moderator-to-fuel ratio.* Visited on 20/12/2023. 1993. URL: https://www.standards.doe.gov/standards-documents/1000/1019-bhdbk-1993-v2/@@images/file/.

[8] Belle R. Upadhyaya Thomas W. Kerlin. *Dynamics and Control of Nuclear Reactors.* Elsevier, 2019. URL: https://www.sciencedirect.com/book/9780128152614/dynamics-and-control-of-nuclear-reactors.

[9] *IAPWS Industrial Formulation 1997 for water.* Visited on 26/12/2023. URL: http://www.iapws.org/relguide/IF97-Rev.html.

[10] *Energy dynamics in Modelica.* Visited on 16/12/2023. URL: https://build.openmodelica.org/Documentation/Modelica.Fluid.Types.Dynamics.html.

[11] *Idaho laboratory new pressurizer components.* Visited on 26/12/2023. URL: https://github.com/idaholab/HYBRID/tree/devel/Models/NHES/Fluid/Vessels.

[12] *Core parameters*. Visited on 12/12/2023. URL: https://crpg.mit.edu/sites/default/files/css_injector_images_image/BEAVRS_2.0.2_spec.pdf.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 - Dymola tips

There are a multiple useful features that aren't enabled in Dymola by default, these can simplify the process of debugging and building models by a noticeable margin. Within Dymola there are a few useful settings that help one analyse the model they have built. In the Simulation tab under Setup -> Translation many useful options can be found. One of the first to turn on is "List continuous time states selected". This will allow one to see what variables the model actually selected for use in the simulation. Ideally the variables should all be similar for example, pressure and enthalpy for all fluid parts with no pressure and temperature variables. This can help to identify if the system is actually using the values one wants it to use and not ignoring them. Another useful setting is "List non-linear iteration variables". Since non-linear systems cause our simulations to potentially run a lot slower we should try to keep these to a minimum. By default, Dymola doesn't tell us what variables cause said non-linearities but using this setting it gives us the components involved under the Translation Logs -> Statistics -> Variables appearing in the nonlinear systems of equations. Next in the Debug tab it is useful to turn on "Detailed logging of failed nonlinear solutions". This setting helps one to troubleshoot nonlinear equations on which the simulation stops. Often it will give the names of the components related and while it might not be those components directly causing the issue, it narrows down the search by a large amount. Another useful setting within the Debug tab is "Provide variable dependencies and equation incidence". With this active one can right click variables and click "Plot dependencies" which will open up a separate window showing what other variables the selected variable is dependent on. This can help narrow down unreasonable values the user has selected or help spot cases where the compiler has selected a different value than the one entered. There are other potentially useful settings as well depending on the situation, it's always worth taking a look when dealing with non-linearities or variable initialization. When starting with a new language or tool it's best to start simple and look at documentation. The second best incase of missing documentation is to look at other examples one might find. The TRANSFORM library doesn't contain much documentation by design. The idea of the author is that one can learn the library by exploring the various examples. While this idea has its' merits it can cause an extremely steep learning curve and lengthen the process. Best practice is to start with each component separately and create test scenarios to see if they behave as expected. A good idea would be to try out different solvers within the simulation tab and find one that is most suited for what one is trying to accomplish. If the end goal is not clear, then either one of the solvers can be chosen. The reason for choosing one and sticking with it is that the different solvers have varying idiosyncrasies and getting used to one can help speed up the learning process. The more popular one’s for nuclear simulations seem to be Esdirk45, Dopri45, Cerk34 or Cerk45. This model shown in this thesis is made using Esdirk45.