Title

Digital Twins as Testbeds for Iterative Simulated Neutronics Feedback Controller Development

Author

Ong, Theodore Kay Chen

Publication Date

2024

Peer reviewed|Thesis/dissertation

Digital Twins as Testbeds for Iterative Simulated Neutronics Feedback Controller Development

By

Theodore Kay Chen Ong

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Nuclear Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Per F. Peterson, Chair
Professor Van P. Carey
Associate Professor Massimiliano Fratoni

Spring 2024

Digital Twins as Testbeds for Iterative Simulated Neutronics Feedback Controller Development

Abstract

Digital Twins as Testbeds for Iterative Simulated Neutronics Feedback Controller Development

by

Theodore Kay Chen Ong

Doctor of Philosophy in Engineering - Nuclear Engineering

University of California, Berkeley

Professor Per F. Peterson, Chair

Before a new nuclear reactor design can be constructed and operated, its safety must be demonstrated using models that are validated with integral effects test (IET) data. However, because scaled integral effects tests are electrically heated, they do not exhibit nuclear reactor feedback phenomena. To replicate the nuclear transient response in electrically heated IETs, we require simulated neutronics feedback (SNF) controllers. Such SNF controllers can then be used to provide SNF capabilities for IET facilities such as the Compact Integral Effects Test (CIET) at the University of California, Berkeley (UC Berkeley). However, developing SNF controllers for IET facilities is non-trivial. To expedite development, we present the use of Digital Twins as testbeds for iterative SNF controller development. In particular, we use a Digital Twin of the Heater within CIET as a testbed for SNF Controller Development. This Digital Twin with SNF Capabilty is run as an OPC-UA server and client written almost entirely in Rust using Free and Open Source (FOSS) code. We then validate the Digital Twin with experimental data in literature. We also verify the transfer function simulation and Proportional, Integral and Derivative (PID) controllers written in Rust using Scilab. Moreover, we demonstrate use of data driven surrogate models (transfer functions) to construct SNF controllers in contrast to using the traditional Point Reactor Kinetics Equations (PRKE) models with the hope that they can account for the effect of spatially dependent neutron flux on reactor feedback. To construct the first surrogate models in this work, we use transient data from a representative arbitrary Fluoride Salt Cooled High Temperature Reactor (FHR) model constructed using OpenMC and GeN-Foam. Using the Digital Twin as a testbed, two design iterations of the SNF controller were developed using the data driven surrogate model. Compared to the potential development time taken in using physical experiments, using the digital twin testbed for SNF controller development resulted in a significant time saving. We hope that the approaches used in this dissertation can expedite testing and reduce expenditure for licensing novel Gen IV nuclear reactor designs.

Contents

List of Figures

List of Tables

Acknowledgments

First off, I am thankful to my Principal Investigator (PI) Professor Per F. Peterson. He has been most supportive of my decisions and endeavours especially during the COVID-19 pandemic. He was also willing to support me financially as the cost of living and tuition fees became more unbearable during my course of study in the Bay Area. I thank him for his encouragement for me to develop my own ideas, and for his enthusiasm for nuclear technology. All this made my PhD experience enjoyable even though there were moments that were arduous. It was a joy having first seen Professor Peterson on YouTube expounding the Compact Integral Effects Test (CIET) to now having worked with him for more than five years. I could not have finished this dissertation without him nurturing me and guiding me, especially when I needed to scope down my work.

Special thanks to the Singapore Nuclear Research and Safety Initiative (SNRSI), for the financial support over the years of my PhD and for giving me the opportunity to work with inspiring people in the University of California, Berkeley. I appreciate the extra support given to me especially when things got hard during the COVID-19 pandemic. Thanks to Professor Chung and to Dr. Garvin Mak for helping me during these tough times.

I am also thankful for Associate Professor Fratoni, who has introduced me to the world of nuclear simulations and computing , and for first introducing me to vim in class. Since then I have been an avid user and fan of Neovim and Linux. I also appreciate the time given for discussion about my dissertation and for introducing me to his research group and his members such as Jun Shi, from whom I learnt about GeN-Foam. Also from his group I knew Yves Robert, with whom I enjoyed many discussions about OpenFOAM. I even translated models from Yves's C++ code in GeN-Foam into Rust for my work.

I give thanks for Professor Carey for making time in his busy schedule to help me spruce up my dissertation. Also thanks to him for teaching me in convection simulation as well. This has proved valuable for this dissertation.

I acknowledge Assistant Professor Raluca Scarlat and Assistant Professor GuanYu Su. They have really trained me well during my Master's Thesis on how to produce well polished academic work. I also thank Professor Scarlat for taking time to nurture students such as myself, and for encouraging me to explore my interests especially when it comes to combining chemical engineering with nuclear engineering.

From my lab group (and department), I acknowledge James Kendrick, Shane Gallagher and Christopher Poresky, my first seniors here who welcomed me into the thermal hydraulics lab and made my first year more comfortable. Many thanks for your advice. I especially thank James and Alan for their moral support when they shared about their faith journey with me in addition to providing helpful academic advice. Next, to Dane De Wet for his insight on frequency response work and sharing with me much literature about the simulated neutronics feedback. Many thanks to Ishak Johnson as well, who brought so much life and joy into the lab, and who is responsible for many fun lab outings. Thanks to Omar too, who brought fun and inspiration as he shared his many ideas about startups. In fact, I learnt

about the Rust programming language and the Typst program from him. This became central to writing performant code for me. Not only that, Omar has also allowed me to use many of the photos he took of CIET for this dissertation. I credit Clara Alivisatos, who worked with Chris to on Digital Twins and OPC-UA servers and clients, thus inspiring me to work on Digital Twins as well when my initial research plan did not work out. Many thanks to Sala and Ian for good vibes, to Oliver for taking care of me when I was sick and on other occasions as well. To Kwangchae for specially cooking for me as well.

I want to acknowledge both my parents for bringing me through to the finish line of my PhD. I could not have done any of this without their love and support. I thank them for supporting me financially and also for frequent visits to the United States, for bonding time, advice, and taking time to listen. I thank them for standing steadfast in their workplace and sharing their stories with me.

Now I give thanks also to my church family both in the Bay Area and back home in Singapore. To the Ang Lip and Lucy Lim, for treating me as their own family essentially when I was far from home, for much wise advice gained through hard knocks over the years, for much time, meals together and support when I felt down and out. Thanks to Ang Lip also for advice in changing my dissertation work from experimental to simulation work even as he shared from his own PhD experiences. To Isaac and Kayla Wolf, who helped me adjust very much to PhD life in my early years and for walking through tough times. To Miguel and Olesia Castuera, for bringing much joy and encouragement into my life through both affirming words and music. To Zach and Kristina Tronstad, for being very dear to me, for nerding out, for extending your home to me as well. To Nick and Emily Tyler, for introducing me to an exiting new way of computing with Linux through ssh, sharing about supercomputing and about your faith journey. Thanks to Nicholas Ngai for inspiring me to use tmux. Special thanks to my Bridges International family. First to Ryan and Megan Gregory, who were more than partners to me in Bridges, but also shared life, thick and thin with me over the years, and even for helping me with things such as moving out of the Bay Area. Next, to Brett and Sohee Procek who have opened their home and their very life to me, and for their support over the years. To my church family in Singapore, firstly Adrian for spending many many weeks with me over zoom and for listening, guiding and nurturing. Also to Pastor PL for nurturing me and encouraging me in my PhD and faith journey. To Mel and Bel for offering much wisdom, dedication and support from my undergraduate days even to today. For Shawn for spending much time in calls when I was overseas, sharing life and wisdom. And to many others too numerous to name as well.

Finally, thanks be to my God and Lord and Saviour Jesus Christ who was manifested in the flesh, vindicated by the Spirit, seen by angels, proclaimed among the nations, believed on in the world, taken up in glory. Now to him who is able to do far more abundantly than all we ask or think, according to the power at work within us, to him be all glory in the church and in Christ Jesus throughout all generations, forever and ever.

אל תהי חכם בעיניך ירא את יהוה וסור מרע
רפאות תהי לשרך ושקוי לעצמותיך

Part I

Digital Twin Construction

Chapter 1

Introduction

1.1 A Graphical Abstract

Time is limited. In the context of climate change, there is limited time to reduce carbon emissions. Whereas in the context of business and research, time is money. Whatever the context is, we want to use our time efficiently and purposefully. If our mission is to reduce carbon emissions quickly, nuclear energy needs to be part of the equation. Nevertheless, the licensing and construction process can hinder nuclear power's effectiveness in reducing future carbon emissions. Therefore, we want to discuss methods that may potentially save time in the context of reactor development. To save some of your reading time as well, I present Figure 1.1:

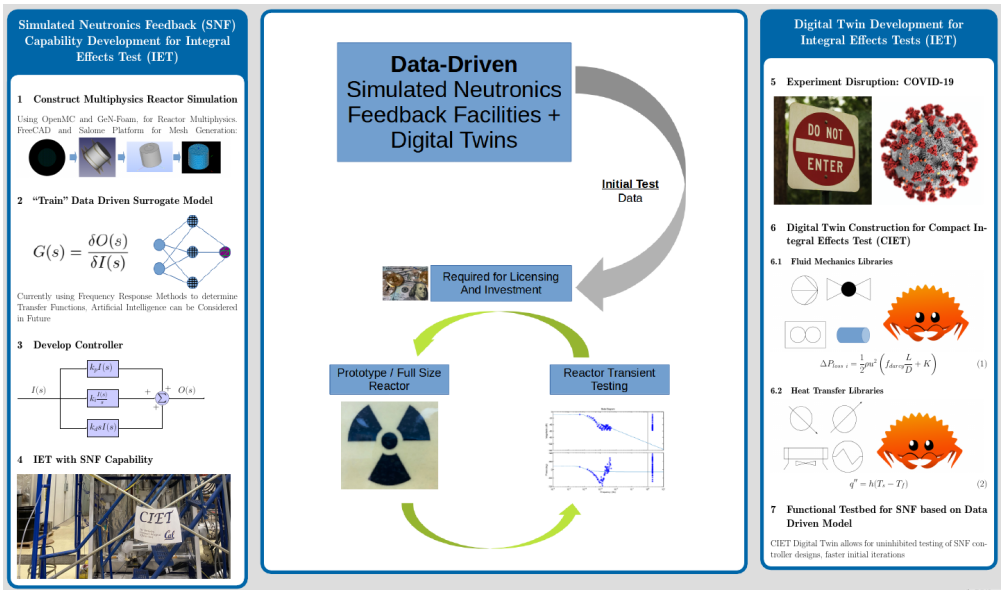


Figure 1.1: Graphical Abstract

Figure 1.1 summarises the key themes and content in this dissertation. We present two tools used in tandem to save time for nuclear reactor development. This is because we wish to make nuclear energy a practical solution for climate change. To do so, a nuclear reactor or power station often has to be run as a business or governmental operation. However, nuclear reactors tend to have long construction and licensing processes. These have made nuclear reactors notorious for cost overruns.

To reduce the time and monetary costs for licensing and developing nuclear reactors such as the fluoride salt cooled high temperature reactor (FHR), the goal of this research was to develop simulated neutronics feedback capabilities for existing scale models of the FHR, otherwise known as Integral Effects Tests (IETs). These IETs help us understand how heat transfer would work in a real nuclear reactor, except that the heat source is non-nuclear. To make the IET behave more like a nuclear reactor, it must have simulated nuclear reactor behaviour. We call this simulated neutronics feedback (SNF). This process is reflected in the left column of Figure 1.1. Nevertheless, the COVID-19 pandemic and lock-down came and rendered the IETs in my research laboratory inaccessible. Hence, with much guidance from my supervisors, mentors and peers, I changed my research direction to develop code suitable for constructing a Digital Twin for the IET for the FHR known as the Compact Integral Effects Test (CIET). This was done in the hope that the Digital Twin of CIET, or at least some of its key components, would be useful for developing SNF capability for CIET. This is reflected in the right column of Figure 1.1. With both these tools, I wanted to provide means which could potentially increase the development speed of test FHRs and commercial FHRs and reduce its development costs and time frame. This is reflected in the central column of Figure 1.1.

1.2 Dissertation Outline

This dissertation provides an example for how using a Digital Twin early in the iterative design process to create SNF Controllers can save time. The early work and availability of a SNF Controller can expedite work done in nuclear reactor design and development.

To show how this was done, we cover much of the content in Figure 1.1. Figure 1.1, of course, is shown in our introduction here in Chapter 1. Figure 1.1 covers much of content for this dissertation, but the fluid mechanics libraries were developed in my master's thesis. Therefore, that work is only referenced and continued upon in this dissertation. Moreover, we have not yet applied the new SNF capabilities in CIET, so this remains as future work in Figure 1.1. These are broad topics in themselves which require separate publications. Nevertheless, the work in this dissertation provides a foundation for future work in SNF controller and SNF facility development.

We start discussions with Digital Twin construction in the first part of this dissertation. This is because we cannot test SNF capabilities without an IET facility or its Digital Twin. Therefore, the construction methodology for Digital Twins forms one of the foundational elements for this dissertation. We review Digital Twins and previous work in Chapter 2. We

then discuss how previous work was extended by developing heat transfer libraries in the Rust Programming Language as shown in Figure 1.1 in Chapter 3. For those unfamiliar with Rust, the Ferris the Crab is Rust's unofficial mascot recognised by the Rust programming community as of 2023.

For this dissertation, we discuss how the electric heater of CIET, or its Digital Twin, is given SNF capabilities. Therefore, we review existing literature for SNF controllers for electrically heated IET facilities and how we could construct a SNF controller for CIET in Chapter 4. The SNF model is not based on Point Reactor Kinetics Equations (PRKE), but rather on more generic data driven surrogate models developed using higher fidelity multiphysics models. To demonstrate this method, a representative arbitrary reactor multiphysics model was constructed using the OpenMC Monte Carlo code and the GeN-Foam reactor multiphysics code in Chapter 5. Using this multiphysics model of an arbitrary reactor, we also discuss how a surrogate model was constructed in Chapter 5. This surrogate model was then used as a basis for SNF Controller development in Chapter 6.

For this dissertation, we do not complete the SNF IET development process in Figure 1.1. While this dissertation demonstrates how a SNF controller could be created using a data-driven surrogate model, further work is needed to apply the same method to more prototypical FHR designs. Nevertheless, the work described in this dissertation forms the basis for future work in SNF controller development. We discuss some of these future possibilities in our conclusion in Chapter 7.

Chapter 2

Literature Review and Principles for Digital Twin Construction

2.1 Digital Twins for Gen IV Reactor Development

Digital Twins have been an attractive area of research for industrial engineering and applications in general. They are useful tool in nuclear plant design, construction, operation and maintenance [Zhao and Guan, 2022]. These help in ensuring that nuclear plants are operated to better mitigate risks, and can also help nuclear plants be built faster and cheaper without compromising safety. Building nuclear plants with better economics without compromising safety is especially important since the high capital costs of nuclear power plants [Helmuth, 1988] tend to make them an unattractive investment option. The malady of high capital expenditure (CapEx) is further exacerbated for Gen IV nuclear power plants, where research and development (R&D) costs have to be considered in addition to licensing costs, construction costs and decommissioning costs.

While Gen IV power plants, especially small modular reactors (SMRs), have the potential to be built with better economics, nuclear power plants still suffer from a poor reputation of high CapEx. If one wishes to use Gen IV nuclear power plants to decarbonise the power industry, it is imperative to reduce CapEx as far as possible without compromising safety to attract investment and initiate construction activity. Therefore, any technology that can improve economics of a plant would be a welcome addition. This includes digital twin technology. Therefore, exploring digital twin technology for Gen IV reactor development and operations would be of interest.

One particular Gen IV reactor of interest is the Fluoride Salt Cooled High Temperature Reactor (FHR). This is a solid fuelled, molten salt cooled reactor studied in the University of California, Berkeley [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014; Nicolas Zweibaum, 2015] and, at the time of writing, being developed for commercial power production by Kairos Power [Blandford et al., 2020]. This interest in the FHR is due to its passive safety characteristics such as low pressure, as well as the use of TRISO fuel and

molten salt, [Nicolas Zweibaum, 2015; X. Wang, 2018] all of which help to contain fission products within the site boundary in the case of an accident. The development efforts of these reactors can also benefit from Digital Twin technology. Digital Twins, which are used in various stages of the product life cycle [Kholopov et al., 2019], could be used to test various configurations of the FHR in a relatively safe environment during the development stage even before a working FHR is constructed. This would have reduced costs for FHR development relative to constructing several FHR prototype designs and then testing them one by one. Of course, such Digital Twins need validation since they are basically simulated models. To begin validating such models, both separate effects tests (SETs) and integral effects tests (IETs) are needed. The Compact Integral Effects Test (CIET) is one such integral effects test that has been used to validate simulation codes used to model the FHR. It has previously been used to validate RELAP models [Nicolas Zweibaum, 2015] as well as models built in the System Analysis Module (SAM) [Zou, R. Hu, and Charpentier, 2019].

The need for Digital Twins in Gen IV reactor research became even more apparent during the COVID-19 pandemic, where work on CIET came to a standstill for an extend period of time. I intended then to develop simulated neutronics feedback for CIET, and that was to be the focus of my PhD Dissertation. Unfortunately, this was not to be since CIET was not usable during the COVID-19 pandemic lockdown. The COVID-19 lockdown was only one of many disruptions that could occur and slow down experimental work. The only way to continue research during such extreme circumstances was through the use of simulations such as digital twins. Even during normal disruptions, having digital twins to expedite research and development work would be beneficial. This gave me motivation to start developing a Digital Twin for CIET so that I could have a system with which to test my simulated neutronics feedback controllers at anytime of the day without fear interruption, or of damaging components or Therminol-VP1 spills in CIET. Such things would have expedited my research a lot. I reckoned that developing a Digital Twin for CIET would have been useful for more purposes than what I had originally intended. Therefore, I began my journey into Digital Twin development.

The main component I needed to construct in CIET to test my simulated neutronics feedback controller was the heater. Therefore, in this chapter, we review important literature for constructing a Digital Twin of CIET, with more emphasis on its heater. Firstly, we review why CIET was constructed and why it could use Therminol VP-1, also known as Dowtherm A, as a surrogate fluid for the molten salt used in the FHR, Li_2BeF_4 (FLiBe). We then review some previous work in creating an isothermal Digital Twin of CIET from my master's thesis. Next, we consider how we may start constructing the heat transfer libraries suitable for transient heat transfer simulation for CIET. Lastly, we review literature relevant to solver stability relevant for heat transfer solvers used by the Digital Twin.

2.2 On the use of Surrogate Fluids for the Compact Integral Effects Test (CIET)

CIET was constructed for the purpose of understanding thermal hydraulics phenomena within the FHR [Nicolas Zweibaum, 2015]. Ideally, we would use a molten salt based IET to study thermal hydraulics of FHR. Unfortunately, running test loops with hot FLiBe is problematic for a number of reasons, including Beryllium toxicity and a high salt temperature of 550-700 $°C$. These difficulties motivated studies to investigate the use of simulant fluids such as water or heat transfer oils to study thermal hydraulic phenomena of the FHR [Bardet and Per F Peterson, 2008]. These simulant fluids were much less problematic to use than FLiBe. A heat transfer oil, known as Dowtherm A or Therminol VP-1, was then found to be a suitable candidate for this very purpose. Therminol VP-1 or Dowtherm A at about 80 to 110 $°C$ matches the Prandtl number (Pr) of Li_2BeF_4 (FLiBe) at about 600 to 700 $°C$ [De wet, Per F. Peterson, and Greenwood, 2019; Nicolas Zweibaum, 2015]. Given this discovery, the next step was to build an IET using Therminol VP-1 as a surrogate fluid for FLiBe. These efforts eventually led to the construction of CIET.

We can verify that the Prandtl Numbers for both molten FLiBe and Therminol VP-1 match using thermophysical property correlations for both fluids. For Dowtherm A, the correlations are shown in Equations 2.1 to 2.4 [Nicolas Zweibaum, 2015]:

$$\rho(kg/m^3) = 1078 - 0.85(T°C) \ ; \ T(°C) = [20, 180] \tag{2.1}$$
$$\mu(Pa \cdot s) = 0.130/(T°C)^{1.072} \ ; \ T(°C) = [20, 180] \tag{2.2}$$
$$c_p(J/(kg \cdot K)) = 1518 + 2.82(T°C) \ ; \ T(°C) = [20, 180] \tag{2.3}$$
$$k(W/(m \cdot K)) = 0.142 - 0.00016 \cdot (T°C) \ ; \ T(°C) = [20, 180] \tag{2.4}$$

Here, ρ refers to density, μ is dynamic viscosity, c_p is specific heat capacity and k is thermal conductivity. For FLiBe, the same four thermophysical properties are presented in [Sohal et al., 2010] Equation 2.5 to 2.8:

$$\rho(kg/m^3) = 2415.6 - 0.49072(T(K)) \ ; \ T(K) = [732.2, 4498.8] \tag{2.5}$$
$$\mu(Pa \cdot s) = 0.000116 \exp\left(\frac{3755}{T(K)}\right) \ ; \ T(K) = [873, 1073] \tag{2.6}$$
$$c_p(J/(kg \cdot K)) = 2415.8 \ ; \ T(K) = [600, 1200] \tag{2.7}$$
$$k(W/(m \cdot K)) = 0.629697 + 0.0005 \cdot (T(K)) \ ; \ T(K) = [500, 650] \tag{2.8}$$

The thermal conductivity correlation [Sohal et al., 2010] developed was initially meant for the temperature range of 500 K to 650 K. This meant that it could not predict viscosities normal FHR working temperatures of about 600 to 700 $°C$. Nevertheless, we could investigate if this correlation can be extrapolated to work for this temperature range if we have

thermal conductivity measurements within 600 to 700 $°C$. In literature, several of these measurements were reported [Romatoski and L.-W. Hu, 2017] within this temperature range with a measurement uncertainty of ±10%. I plot the literature data along with Sohal's correlation in Figure 2.1:

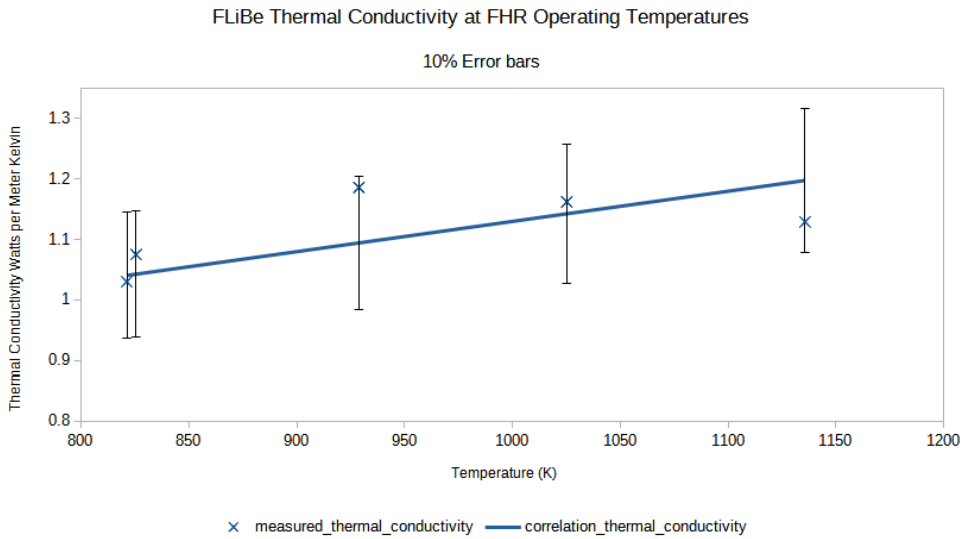


Figure 2.1: FLiBe Thermal Conductivity Correlation and Measured Data

We can see in Figure 2.1 that the correlation can indeed predict the thermal conductivity of FLiBe within this 10% measurement error. Thus, extrapolating Sohal's correlation to the operating temperature range of the FHR would be acceptable. We can then use this correlation for calculating the Prandtl number of FLiBe at about 600 to 700 $°C$.

Now, when comparing the Prandtl number of both Therminol VP-1 and FLiBe, we want to check if they match closely enough. Thus, we should do some uncertainty propagation in order to ascertain if the Prandtl numbers match within the experimental uncertainty. The Prandtl number is defined as [Perry and Green, 2015; Bejan, 2013]:

$$\mathrm{Pr} = \frac{\nu}{\alpha} = \frac{\mu c_p}{k} \tag{2.9}$$

Where ν is momentum diffusivity or kinematic viscosity. It can be expressed as $\nu = \frac{\mu}{\rho}$. α is thermal diffusivity and this can be written as $\alpha = \frac{k}{\rho c_p}$. Based on Equation 2.9, the measurement uncertainties of μ, c_p and k will contribute to the uncertainty of Pr. As mentioned previously, c_p has an measurement uncertainty of about 20% [Lichtenstein et al., 2022; Sohal et al., 2010; Romatoski and L.-W. Hu, 2017] and k has a measurement uncertainty of about 10% [Romatoski and L.-W. Hu, 2017]. For the correlation (Cantor's correlation) presented [Sohal et al., 2010], μ has a reported uncertainty of about 15 to 20%. We can do simple uncertainty propagation as follows [Todreas, Kazimi, and Massoud, 2021; BIPM et al., 2008]:

$$\begin{aligned}
\delta \mathrm{Pr}^2 &= \left(\frac{\partial \mathrm{Pr}}{\partial \mu}\right)^2 \delta\mu^2 + \left(\frac{\partial \mathrm{Pr}}{\partial c_p}\right)^2 \delta c_p^2 + \left(\frac{\partial \mathrm{Pr}}{\partial k}\right)^2 \delta k^2 \\
&= \left(\frac{c_p}{k}\right)^2 \delta\mu^2 + \left(\frac{\mu}{k}\right)^2 \delta c_p^2 + \left(-\frac{\mu c_p}{k^2}\right)^2 \delta k^2 \\
&= \left(\frac{\mathrm{Pr}}{\mu}\right)^2 \delta\mu^2 + \left(\frac{\mathrm{Pr}}{c_p}\right)^2 \delta c_p^2 + \left(-\frac{\mathrm{Pr}}{k}\right)^2 \delta k^2 \\
\left(\frac{\delta \mathrm{Pr}}{\mathrm{Pr}}\right)^2 &= \left(\frac{\delta\mu}{\mu}\right)^2 + \left(\frac{\delta c_p}{c_p}\right)^2 + \left(\frac{\delta k}{k}\right)^2
\end{aligned} \tag{2.10}$$

In this context, $\delta\mathrm{Pr}$, $\delta\mu$, δc_p and δk are the associated uncertainties of Pr, μ, c_p and k. If we take the fractional uncertainties of μ, c_p and k to be 0.15, 0.20 and 0.10 respectively, then:

$$\begin{aligned}
\left(\frac{\delta \mathrm{Pr}}{\mathrm{Pr}}\right)^2 &= \left(\frac{\delta\mu}{\mu}\right)^2 + \left(\frac{\delta c_p}{c_p}\right)^2 + \left(\frac{\delta k}{k}\right)^2 \\
&= 0.15^2 + 0.20^2 + 0.10^2 \\
\left(\frac{\delta \mathrm{Pr}}{\mathrm{Pr}}\right) &= 0.2693 \approx 0.27
\end{aligned} \tag{2.11}$$

For the purposes of error bars, the fractional uncertainty can be rounded down to 0.26. The 20% uncertainty of c_p contributes the most to the fractional uncertainty of Pr. This is supported in literature as there is a large disparity in c_p values for FLiBe. While we have c_p of 2415.8 $J/(kg \cdot K)$ provided in literature [Sohal et al., 2010], other values in literature for c_p of FLiBe could be as low as 1840 $J/(kg \cdot K)$ [Lichtenstein et al., 2022]. I plotted Pr of Therminol VP-1 along with Pr for FLiBe using both these c_p values in Figure 2.2:

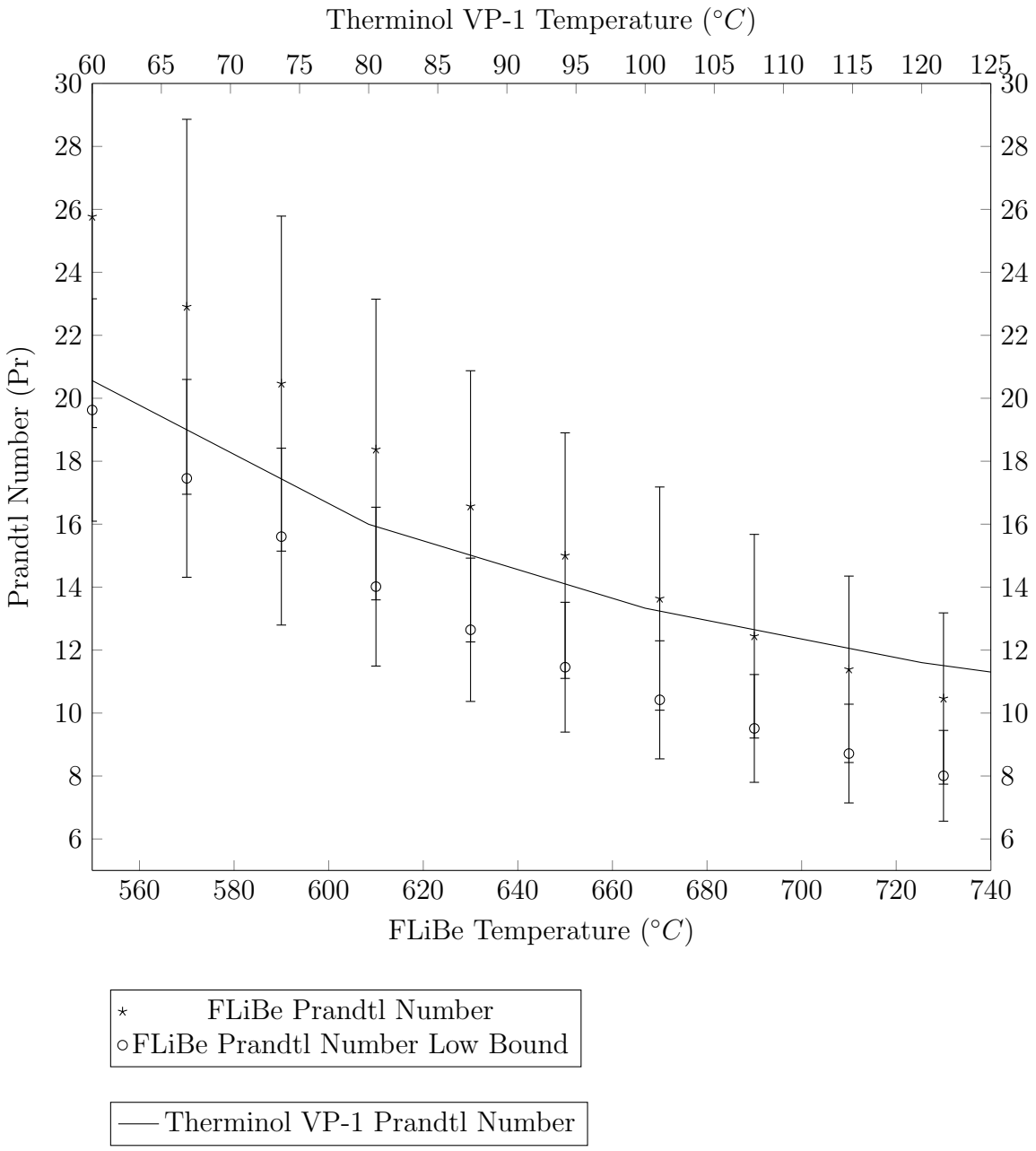


Figure 2.2: Comparison of FLiBe Prandtl Number with Therminol VP-1 (also known as Dowtherm A) Prandtl Number [Nicolas Zweibaum, 2015], Low Bound FLiBe Prandtl Number [Lichtenstein et al., 2022] added for comparison

For the reader's convenience in obtaining values to plot the Prandtl number comparisons in Figure 2.2, a correlation describing the Prandtl number of Therminol VP-1 in the range of $60°C$ to $180°C$ is shown in Equation 2.12:

$$\mathrm{Pr}_{Therminol\ VP-1} = \frac{442}{T(^{\circ}C)^{0.757}} \tag{2.12}$$

This was constructed using regression algorithms in LibreOffice Calc to produce a trendline of Prandtl numbers produced using Equations 2.1 to 2.4. This correlation agrees with the said trendline to within 3.2% of the Prandtl number value calculated from the thermophysical property data. Similarly, a low bound Prandtl number correlation for FLiBe in the range $550^{\circ}C$ to $730^{\circ}C$ is shown in Equation 2.13:

$$\mathrm{Pr}_{FLiBe} = 2.91 \times 10^{-4} T(^{\circ}C)^2 - 4.56 \times 10^{-1} T(^{\circ}C) + 1.88 \times 10^2 \tag{2.13}$$

Equation 2.13 reproduces plotted Prandtl number data in Figure 2.2 to within 4.4%.

Plotting Pr of Therminol VP-1 and FLiBe in Figure 2.2, we can see that these match within the uncertainty of Pr. This confirms the notion that Pr matches between FLiBe and Therminol VP-1 (also known as Dowtherm A) in the literature [Bardet and Per F Peterson, 2008; Nicolas Zweibaum, 2015]. In Figure 2.2, I have also added new measurement data for FLiBe c_p post the construction and operation of CIET (about 2015). As mentioned before, a new c_p measurement for FLiBe was reported at 1840 $J/(kg \cdot K) \pm 5\%$ [Lichtenstein et al., 2022]. This was deemed to have matched the existing literature values [Lichtenstein et al., 2022] about 2415.8 $(J/(kg \cdot K))$, due to the 20% uncertainties of FLiBe's c_p. On Figure 2.2, the new c_p data [Lichtenstein et al., 2022] was plotted as a low bound Pr for FLiBe. Similar to Equation 2.13, a low bound Prandtl number correlation for FLiBe in the range $550^{\circ}C$ to $730^{\circ}C$ is shown in Equation 2.14:

$$\mathrm{Pr}_{FLiBe\ Lichtenstein2022} = 2.21 \times 10^{-4} T(^{\circ}C)^2 - 3.46 \times 10^{-1} T(^{\circ}C) + 1.43 \times 10^2 \tag{2.14}$$

Equation 2.14 reproduces plotted Pr data in Figure 2.2 to within 2.3%.

The associated uncertainty is calculated as:

$$\begin{aligned} \left(\frac{\delta \mathrm{Pr}}{\mathrm{Pr}}\right)^2 &= \left(\frac{\delta \mu}{\mu}\right)^2 + \left(\frac{\delta c_p}{c_p}\right)^2 + \left(\frac{\delta k}{k}\right)^2 \\ &= 0.15^2 + 0.5^2 + 0.10^2 \\ \left(\frac{\delta \mathrm{Pr}}{\mathrm{Pr}}\right) &= 0.187 \approx 0.19 \end{aligned} \tag{2.15}$$

For the purposes of plotting error bars, I round down the fractional uncertainty of Pr to 0.18.

Thus, we have a high bound and low bound Pr due to different literature values for c_p. Figure 2.2 shows is that shows that the Therminol VP-1 Pr at 60 $^{\circ}C$ to $105^{\circ}C$ lies within the high and low bound FLiBe Pr envelope which exists due to variability in c_p from literature data. Therefore, it further strengthens the notion that Therminol VP-1 can be used as a

surrogate fluid for FLiBe at 560 to 680 $°C$. While Pr of Therminol VP-1 may not match Pr of the low bound data within the uncertainty bounds shown in Figure 2.2, it still shows that Therminol VP-1 can be used as a surrogate fluid for FLiBe for a FLiBe temperature of up to about 615 $°C$. We could, of course, tweak the Therminol VP-1 Pr curve to make it match the FLiBe data by adjusting the scaling of the graphs, but that is outside the scope of this work. For now, Figure 2.2 shows that the original Pr of FLiBe and Pr of Therminol VP-1 matches within measurement uncertainties, confirming what was already stated in literature [Bardet and Per F Peterson, 2008; Nicolas Zweibaum, 2015].

The matching Pr for FLiBe and Therminol VP-1, was a strong justification for using Therminol VP-1 as a surrogate fluid for FLiBe in CIET. However, to obtain thermo-hydraulic similitude with the FHR, one would also have to match the Reynolds (Re), Froude (Fr) and Grashof number (Gr) in CIET to that of the prototypical FHR. If we do so, the Nusselt Number (Nu) for both forced and natural convection would be matched [Bardet and Per F Peterson, 2008], and there would be similitude between CIET and the prototypical FHR [Nicolas Zweibaum, 2015]. Hence, CIET was built in such a way to achieve similitude. This is approximately a height scaling of 50% and heater power scaling of 2% at prototypical conditions [Nicolas Zweibaum, 2015]. Therefore, CIET was used to study natural circulation phenomena [Nicolas Zweibaum, 2015] as well as forced circulation transients [De wet, Per F. Peterson, and Greenwood, 2019] for the FHR. The details of scaling CIET to different reactor types is beyond the scope of this dissertation. Interested readers can read Johnson's dissertation for a more in depth look at scaling specifically for CIET [I. M. B. Johnson, 2022]. In any case, we established use of CIET for FHR thermal hydraulic studies. Due to this, it was also natural for system level codes meant to simulate the FHR to also be validated using experimental data from CIET. For example, CIET has been used to validate models in RELAP [Nicolas Zweibaum, 2015] and the System Analysis Module (SAM) [Zou, R. Hu, and Charpentier, 2019].

2.3 Review of Digital Twins and Previous Work

Now that we have reviewed why CIET was used as a scaled IET for the FHR, we shall now review previous work done to developed Digital Twins for CIET. I began my work in Digital Twins previously in my master's thesis where I explored means of constructing an isothermal digital twin of the compact integral effects test (CIET) [Ong, 2023]. As mentioned earlier, CIET is a scaled surrogate fluid integral effects test (IET) of the Gen IV reactor known as the fluoride salt cooled high temperature reactor (FHR) [Zweibaum, Guo, et al., 2016]. This was merely a first step towards constructing a fully fledged Digital Twin of CIET which is free and open source (FOSS). The work for developing such a fully fledged Digital Twin entails many details which can span several publications, some of which are beyond the scope of this Dissertation. In this Dissertation, I only focus on developing a Digital Twin of CIET's Heater along with its required libraries. This is because CIET's Heater is the most relevant component required for development of a simulated neutronics feedback controller, which

was my original goal.

While developing the Digital Twin initially, I also focused on making the libraries used for it and its source code FOSS. FOSS Digital Twins are practically non-existent in the context of the nuclear industry to the best of my knowledge. However, constructing a FOSS Digital Twin along with its libraries would be beneficial for students and academic work as it would make thermal hydraulics research more accessible and repeatable. FOSS is preferred in contrast to proprietary Digital Twins because barriers such as paywalls can make Digital Twins inaccessible to some parties [Al-Geddawy, 2020]. Furthermore, any academic researcher can adapt FOSS code to his or her use case. Therefore, I started writing libraries for the Digital Twin of CIET with a strong emphasis on FOSS methodology. We shall first recap some of the highlights of my previous work in the master's thesis to justify some methods taken for this work as well as the nomenclature used. After that, we can then review some literature that addresses some potential challenges faced during the development of the thermal hydraulics library used to model CIET in real-time.

Classification of Digital Twins

Academic work for Digital Twins had to first start by addressing the definitions of Digital Twins. This is because there are many differing definitions in literature of what exactly constitutes a digital twin [Sleiti, Kapat, and Vesely, 2022; Kochunas and Huan, 2021; Van der Valk et al., 2020]. Sometimes these definitions conflict, thus causing confusion and frustrating the reader.

Therefore, I decided to classify digital twins into three types. This was to ensure that all existing definitions of digital twins in literature were included, and also to ensure that distinctions between the different types of digital twins are maintained. The typing methodology is based on a review work by Kochunas et al. [Kochunas and Huan, 2021]. Where digital representations of a physical system were classed as different digital entities based on the flow of real-time data between digital entity and physical asset. These digital entities were called "digital model", "digital shadow" and "digital twin". Rather than use three new terms, I decided to classify these entities into Type I, Type II and Type III Digital Twins respectively. For the reader's convenience, I reproduce a table used from my master's thesis here to illustrate the main aspects between the types of Digital Twins [Ong, 2023]:

Digital Twin Type	Real-Time Flow of Data	Similar Terms used in Literature	Applications
Type I	None	Digital Sibling [Rasheed, San, and Kvamsdal, 2020], Digital Thread [Boschert and Rosen, 2016] Digital Model [Kochunas and Huan, 2021], Digital Twin Prototype [Enders and Hoßbach, 2019; Grieves and Vickers, 2017]	Scenario and Risk Assessment [Rasheed, San, and Kvamsdal, 2020], Penetration Testing [Eckhart and Ekelhart, 2018]
Type II	From Physical Asset	Digital Shadow [Kochunas and Huan, 2021] Digital Angel [Van der Valk et al., 2020] Digital Twin Instance [Enders and Hoßbach, 2019; Grieves and Vickers, 2017]	Fault Detection [Palak Jain et al., 2019] Predictive Maintenance
Type III	To and From Physical Asset	Digital Twin [Kochunas and Huan, 2021] Digital Twin Instance [Enders and Hoßbach, 2019; Grieves and Vickers, 2017]	Model-Based Control [Enders and Hoßbach, 2019; Semeraro et al., 2021]

Table 2.1: Digital Twins and their Types

The Digital Twin typing system works by classifying digital twins according to the flow of real-time information between the digital twin and the physical asset. As seen in table 2.1, Type I Digital Twins do not necessitate flow of real-time information between physical asset

and digital twin. For the reader's convenience, I will again reproduce a figure used in my master's thesis to illustrate how real-time data flows between the physical asset, user and digital twin in real-time [Ong, 2023]. This is Figure 2.3:

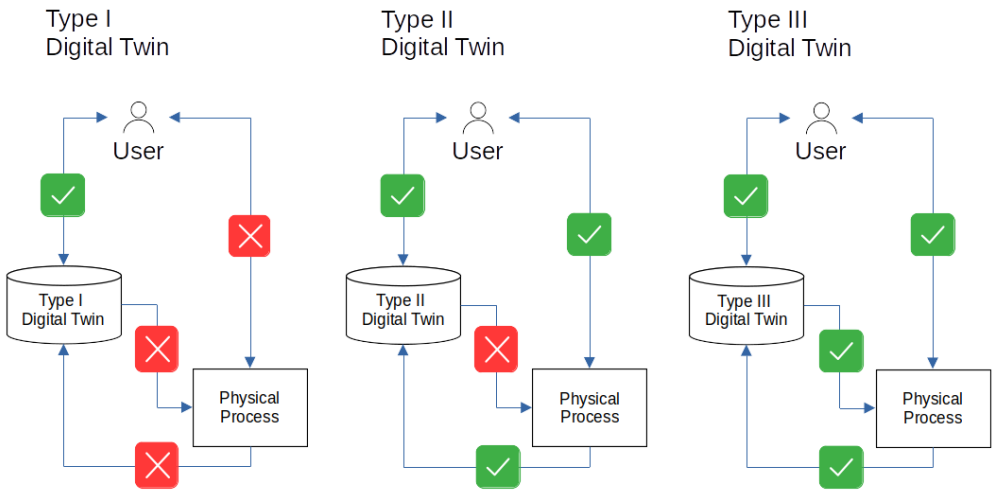


Figure 2.3: Real-Time Data Flow Comparison for each Type of Digital Twin [Ong, 2023]

In contrast, Type III digital twins necessarily require two way flow of real-time information between physical asset and digital twin. Classification of Digital Twins in such a manner is meant to help the reader have a more intuitive grasp as to what a digital twin is when reading literature. At the same time, the Type classification necessarily implies a set of functions each type of digital twins is best suited for. This is because the flow of real-time data is what ultimately impacts capabilities of the said type of Digital Twin. At the same time, the differing requirements for real-time data flows between the types of Digital Twins and their physical assets will ultimately determine development time and costs. This knowledge is very important for the manager making decisions during the product development life cycle. Therefore, I chose this naming system. A fuller explanation of the rationale for this Digital Twin typing system is further explored in my master's thesis [Ong, 2023], and I will not repeat it here.

I hoped that this nomenclature would not confuse readers with extra terminology and instead help the reader make some sense of the inconsistent terminology used in Digital Twin literature. Hence, this naming convention will continue to be used in this work. In this work, the Digital Twin I am constructing for CIET is necessarily a Type I Digital Twin. However, this Type I Digital Twin is meant to be FOSS so that it can be extended with Type II or Type III capability in future work.

Isothermal Digital Twin of CIET Design

Constructing a Type I Digital Twin of CIET would be quite difficult if I were to write all the libraries from scratch. Therefore, I only developed fluid flow libraries in previous work. This was meant only as a first iteration or first step towards constructing a fully fledged Type I Digital Twin of CIET.This is because CIET had several components, and even constructing a Type I Digital Twin of CIET took quite awhile.

A figure of CIET used in my master's thesis is reproduced here for the reader's convenience [Ong, 2023]:

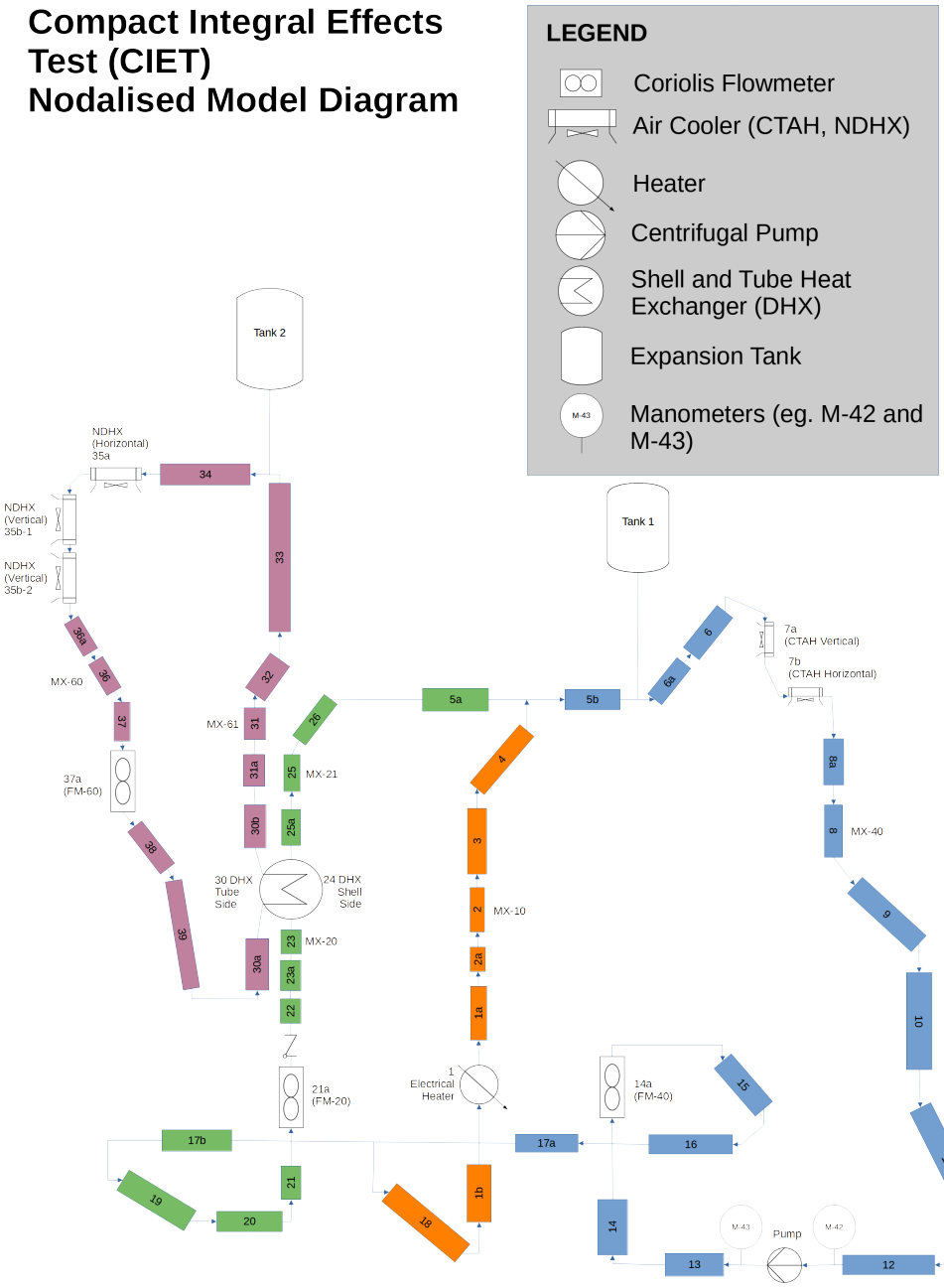


Figure 2.4: Nodalised Model of CIET without Bypass Branch or Direct Reactor Auxiliary Cooling System (DRACS) Loop

Figure 2.4 is a simplified model of CIET because I did not include the bypass branch. This

is because previous models developed, such as those in SAM, excluded the bypass branch [Zou, R. Hu, and Charpentier, 2019]. Furthermore, most experimental tests I could use for code validation excluded the involvement of the bypass branch [De Wet and Per F Peterson, 2020; Nicolas Zweibaum, 2015]. Therefore, I excluded it. Additionally, while the Direct Reactor Auxiliary Cooling System (DRACS) loop was included in CIET's facility physically [Nicolas Zweibaum, 2015], I neglected to include the DRACS loop because it was mostly not used in isothermal operation. Hence, the both the DRACS loop and bypass branch was effectively ignored in my isothermal Digital Twin.

The first Digital Twin was meant to mimic the CIET setup as much as possible at least for isothermal operations. This meant that the mass flowrates for each branch needed to be calculated in real-time (roughly every 100 ms) because the Labview Client received data from the Data Acquisition System every 100 ms. Furthermore, since CIET communicated with the user via a server client interface through a local network and this, too, was meant to be replicated. The server-client interface was important because signals to and from CIET via the Data Acquisition System (DAQ) had a few milleseconds of lag. This time delay would ultimately affect the behaviour of controllers we wish to implement. Furthermore, CIET was being used for remote operation studies [Poresky et al., 2022], and building a digital twin with a server-client interface over a network was meant to facilitate such remote operations research.

These two requirements drove the design of the isothermal digital twin, and will continue to drive the design as we construct a digital twin with heat transfer capabilities.

Server Client Interface

The server-client interface with remote operation via Open Platform Communications Unified Architecture (OPC-UA) protocol can be visualised by Figure 2.5 [Ong, 2023]:

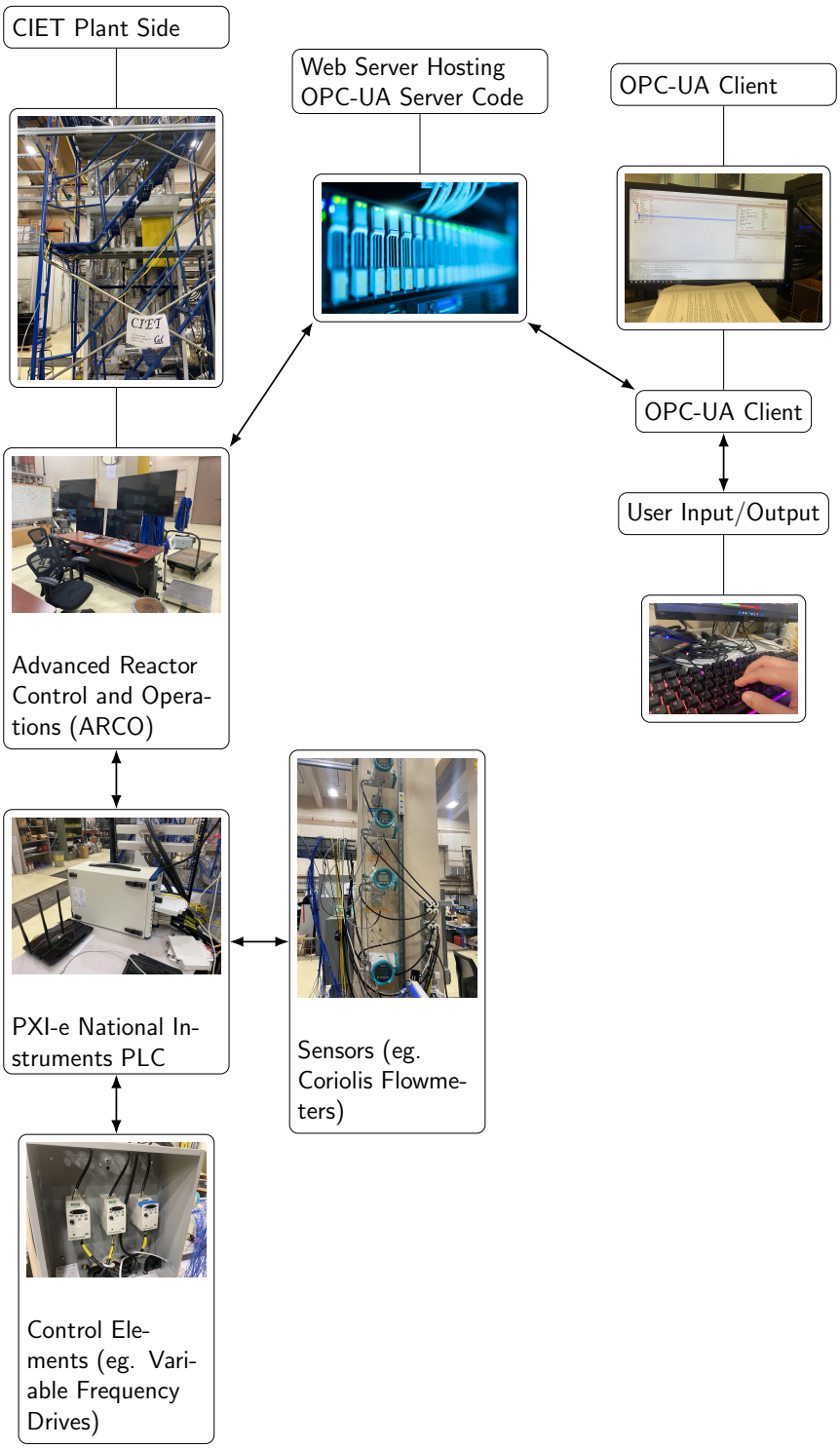


Figure 2.5: Approximate Remote Operation Architecture for CIET

We can see that operators are able to control CIET remotely via a web server run-

ning an OPC-UA server via an OPC-UA client. These servers communicate directly with ARCO which obtains process data from a National Instruments PXI-express or "PXI-e" Programmable Logic Controller (PLC). The PLC is able to obtain process data and control CIET via communication with sensors and control elements.

The Digital Twin was meant to somewhat mimic this architecture, or at least be built in a way that extension to this architecture would be possible without rewriting the entire code. Its server-client architecture is shown in Figure 2.6:

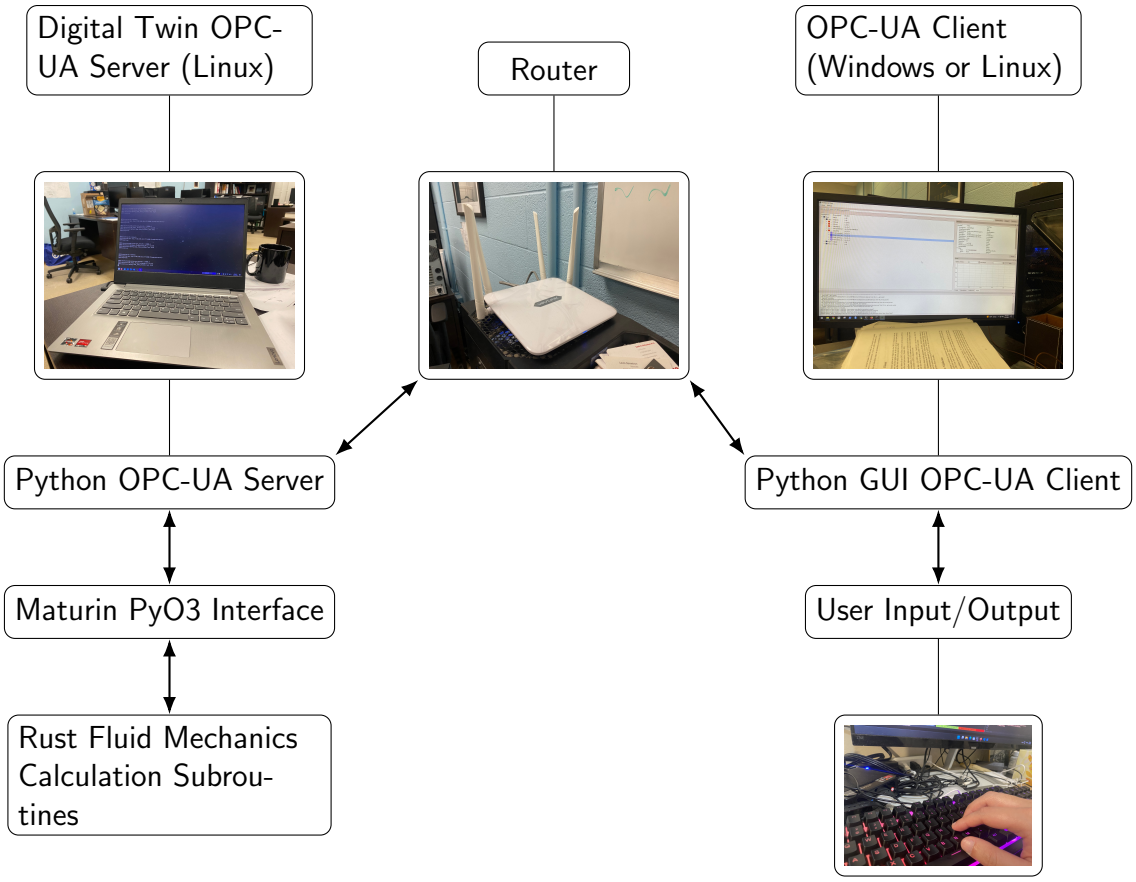


Figure 2.6: Architectural Diagram of Isothermal Type I Digital Twin

Designing CIET in this manner ensured that remote operation studies could be performed in future using this digital twin as a testbed. This server-client interface also allowed for a variety of OPC-UA clients to interface with the digital twin. This made the digital twin flexible with regards to its choice of OPC-UA client. I retained this capability and architecture when I constructed a digital twin of CIET's Heater with heat transfer capabilities or other such features.

Rust and Python Libraries

In Figure 2.6, we mentioned the use of the Rust and Python programming languages to construct this FOSS digital twin of CIET.This was because Rust had sufficiently high speed to perform calculations within the 100 ms even when the binary was not optimised for speed even on an everyday consumer laptop (Ideapad) [Ong, 2023]. This gave some assurance that when more calculation demands were placed on the Digital Twin, the Digital Twin could still perform in a satisfactory manner even on a single thread. Python was used in conjunction with Rust code because it contained an OPC-UA server and client package [Oroulet et al., 2022] which was relatively easy to use (compared to the C# server and client FOSS codes) and established in literature[Zidek et al., 2020]. As a result, Python and Rust was used in using Maturin PyO3 as an interface. Further justifications for using Rust are described below, with a fuller version available in my masters thesis [Ong, 2023]. Due to a successful programming experience using the Rust programming language, I decided to continue using the Rust programming language to develop heat transfer libraries for a Type I Digital Twin of CIET's Heater.

Unit Safety with Rust Rust features a package which helps the user write code in a unit safe manner. This means that calculations with the wrong units can be prevented at compile time. These erroneous calculations include adding a quantity in feet to a quantity in meters, or adding a quantity of length to a quantity of energy by accident. Such mistakes have proved costly as shown by the crash of the Mars Polar lander [Oberg, 1999], where errors due to incorrect units caused US$125 million (in 1999 dollars) to go up in smoke.

While we are not in immediate danger of wasting several million US dollars, we still wish to have safeguards against as this could allow us to avoid unit based errors which could be costly in terms of time. The package that provides this safety is called "uom" which is short for "Units of Measure" [Boutin, 2023]. This package has proved useful and I used it when developing heat transfer libraries for the digital twin.

Now, unit safety is quite dependent on Rust's static type system. Static typing is absent by default in Python, and therefore quantities are most simply represented as floating point values (floats). These quantities do not have units attached to them, which makes it "unit unsafe". Having the OPC-UA server run in Python means that every unit safe quantity that we wish to transmit over the OPC-UA server-client architecture needs to be converted into floats. For an isothermal simulation, this was workable. However, for a fully fledged Digital Twin of CIET with Heat Transfer simulation capabilities, we would need to at least replicate all the thermocouples found within CIET. Since CIET has on the order of 30 or more thermocouples, we would have to include all these sensors in our Digital Twin client. This means we need 30 or so Python float values in our Python client. If we were to use unit safe calculations in Rust, we would need to translate as many as 30 unit safe quantities from Rust to Python via the Maturin PyO3 interface. This may not be practical, and hence, we used a Rust OPC-UA server as opposed to Python for this very reason.

Data Race Safety with Rust Furthermore, Rust is able to guide the programmer to build multithreaded applications with significantly less fear of data races. This gives the assurance that when extra speed is required using parallel computing, the code can be upgraded using Rust without fear of extremely hard to debug data races [Saligrama, Shen, and Gjengset, 2019]. Having the compiler perform compile time checks for memory safety and thread safety is extremely helpful when the codebase gets more complex. I perceive that as more features are added to the Digital Twin, the codebase would become larger, and the Digital Twin may require parallelism some time in future to ensure that calculations are fast enough. Using Rust would ensure Data Race Safety at compile time so that parallelism would be less painful to implement in future as compared to other languages. Therefore, Rust will continue to be used in writing Digital Twin libraries and constructing the Digital Twin.

Parallel Branch Flow Solver with Two Levels of Nested Iteration

One key feature of the Digital Twin constructed for CIET is its solver for flow in parallel branches. The key issue with solving for flow across parallel branches is nested iterations. Increasing the level of nested iterations would tend to exponentially increase the time required to obtain a solution. This can be shown in Figure 2.7 [Ong, 2023]:

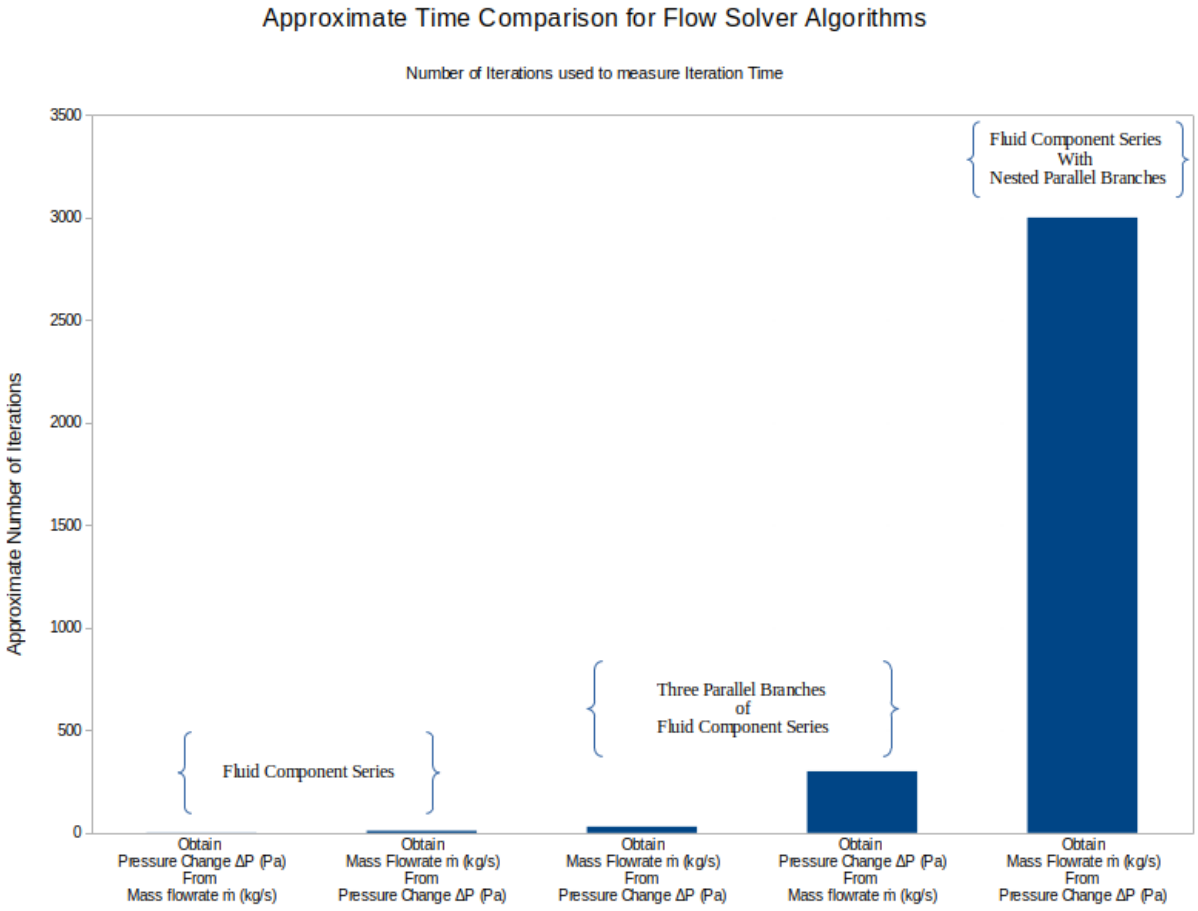


Figure 2.7: Approximate Computation Time for Nested Iteration Loops [Ong, 2023]

These nested iterations arise because for a series of pipes, finding mass flowrate $\dot{m}$ from pressure loss or pressure change ΔP normally requires iteration. One could conceivably pre-calculate the system curve and thus save the iteration time. However, system curves would change with temperature distribution. Therefore, this pre-calculation approach was not taken. This is one level of iteration.

For flow through a collection of parallel branches, finding ΔP given a total $\dot{m}$ across the branches requires iteration. Given that we must iteratively guess individual $\dot{m}$ given a ΔP across each branch, we would then arrive at an additional level of iteration. We would thus have two levels of nested iterations.

Now, suppose the parallel branches are part of a larger collection of pipes or components in series, finding $\dot{m}$ from ΔP across this collection would require a third level of nested iteration. I found that this took exceedingly long for the solver to find a solution. Figure 2.7 shows that the exponential increase in the number of iterations looked like.

To solve this problem, CIET was modelled as three fluid branches in parallel so that

only two levels of nested iteration were required. I intend to continue using this approach in this work. A fuller explanation is available in my masters's thesis [Ong, 2023] for interested readers.

Summary of Previous Results

The Digital Twin (Type I) was built with a server-client interface using Rust and Python. It was modelled as three parallel flow branches so that the solver was able to obtain solutions within 100ms. Experimental data from previous isothermal flow tests [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014; Nicolas Zweibaum, 2015] was used to validate it. For the CIET Heater within the Heater Branch, forced convection between the heater and CTAH branches is very relevant. Therefore, the plot is presented here in figure 2.8 [Ong, 2023]:

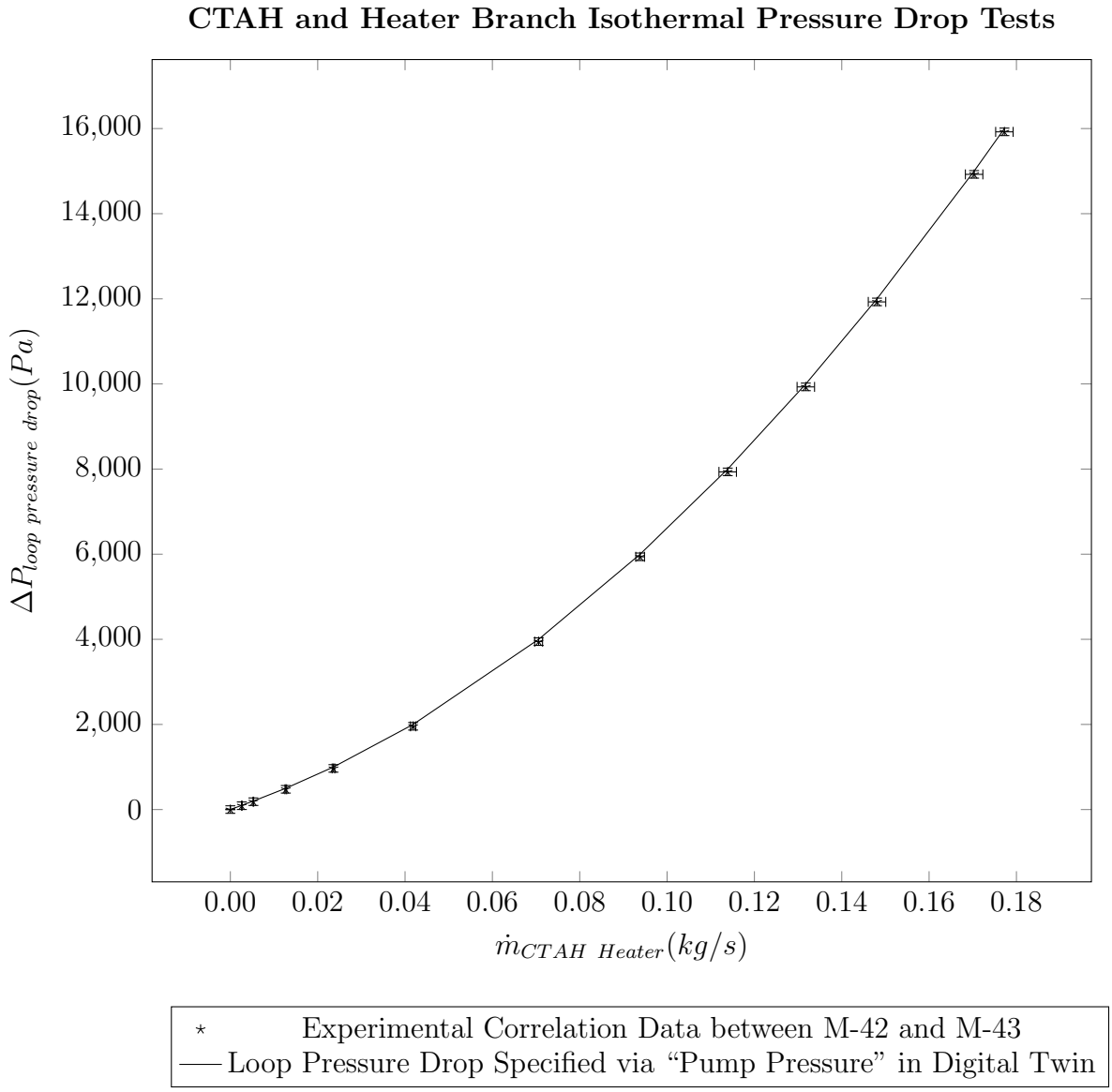


Figure 2.8: Digital Twin System Curve Compared to Experimental System Curve from Manometer M-42 and M-43 Plotted in Absolute Pressure Units (CTAH and Heater Branch)

The dataset used for Figure 2.8 is presented in table 2.2 [Ong, 2023]:

FM-40 Mass Flowrate (kg/s)	Digital Twin Loop Pressure Drop (Pa)	Experimental Data Correlation Pressure Drop (Pa)
0.177	16000	15920
0.170	15000	14920
0.148	12000	11930
0.132	10000	9930
0.114	8000	7930
0.0938	6000	5940
0.0706	4000	3950
0.0418	2000	1960
0.0236	1000	970
0.0127	500	480
0.00527	200	180
0.00263	100	90
0	0	0
-0.0418	-2000	outside data range
-0.132	-10000	outside data range

Table 2.2: Digital Twin System Curve Compared to Experimental System Curve from Manometer M-42 and M-43 Plotted in Absolute Pressure Units (CTAH and Heater Branch)

We present also the residual plots, we calculate the residuals ($\delta\Delta P_{loop\ pressure\ drop}$) by:

$$\delta\Delta P_{loop\ pressure\ drop} = \Delta P_{digital\ twin\ prediction} - \Delta P_{experimental\ data\ correlation} \tag{2.16}$$

These plots are shown in figure 2.9 [Ong, 2023]:

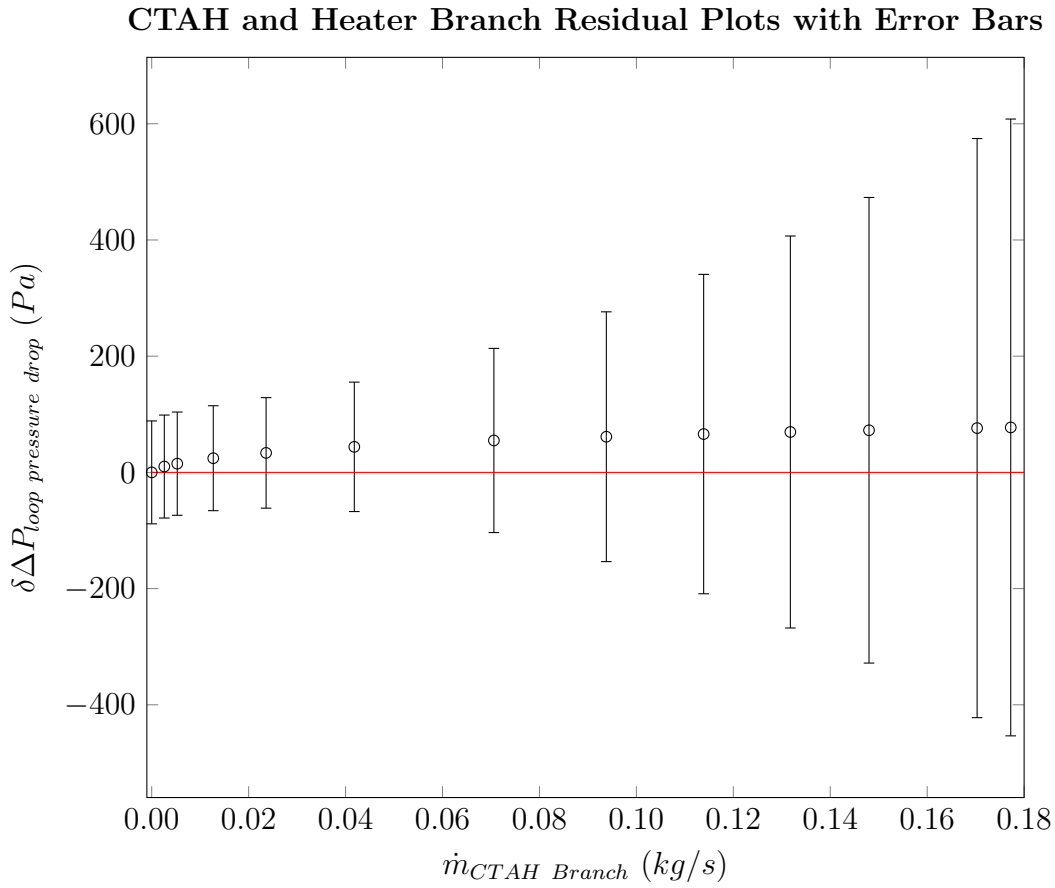


Figure 2.9: Absolute Residual Plot for CTAH and Heater Branch System Curves with Error Bars

These error bars constitute a ±2% flowmeter error plus uncertainty with regards to using graphereader [K. P. Larsen, 2022] to read manometer height data from graphs. The latter of which constitutes an additional uncertainty of $\pm 50Pa$. More details can be seen in my master's thesis [Ong, 2023]. The results were deemed satisfactory as the model matched the experimental data within these error bars.

Since the results were satisfactory in previous work, CIET's future iterations of Digital Twins with heat transfer capability will use this isothermal version of the Type I digital twin as a baseline. For now, however, we focus on developing heat transfer libraries to first construct a real-time Digital Twin of CIET's Heater and some of the components surrounding it before embarking on full loop testing and validation in future work.

2.4 Review of Heat Transfer Libraries and Correlations

Now, to give the digital twin heat transfer capability, libraries need to be written to perform these calculations. In this section, we shall discuss some important equations, correlations and data to be used for the development of a heat transfer library. Most of the efforts will focus on developing a model on CIET's heater. However, to complete the heat transfer library, we will need to model heat exchangers. This is mostly left for future work since we only need a digital twin of the heater in order to test if the controller is working as intended. Lastly, we shall explore some literature important for developing numerically stable solvers.

Mass, Energy and Momentum Balance

Preliminaries

To start modelling CIET's Heater, we need to first consider the general equations to solve for heat transfer. These are the mass, energy and momentum balances. For the purposes of real-time simulations, we need to ensure that the models are simple enough to solve in real-time. For flow in pipe geometries, it is common to use a 1D nodalised model for fluid flow through CIET [Zou, R. Hu, and Charpentier, 2019; De Wet and Per F Peterson, 2020; Nicolas Zweibaum, 2015]. Of course, the pipes surrounding the fluid may be represented by a 2D mesh, but the fluid mesh for system level codes is a 1D mesh. Therefore, we will only consider 1D models in this subsection. For this purpose, it is useful to review some of the 1D mass, energy and momentum balances in literature, such as those presented in Modelica [Casella, Leva, et al., 2003]. This is because Modelica is a well known systems level code for modelling transients for power plants.

The mass balance equation for a 1D geometry is [Casella, Leva, et al., 2003]:

$$A_{XS}\frac{\partial \rho}{\partial t} + \frac{\partial \dot{m}}{\partial x} = 0 \tag{2.17}$$

Where A_{XS} is the cross sectional area, ρ is the density of the differential volume dV, $\dot{m}$ is the mass flow rate, and x is the length coordinate in the direction of flow.

The momentum balance is [Casella, Leva, et al., 2003]:

$$\frac{\partial \dot{m}}{\partial t} + \frac{f_{fanning} P_w}{2\rho A_{XS}^2}\dot{m}|\dot{m}| + \rho g A_{XS}\frac{\partial z}{\partial x} + A_{XS}\frac{\partial P}{\partial x} = 0 \tag{2.18}$$

In Equation 2.18, in addition to the terms already defined for Equation 2.17, we have Fanning friction factor $f_{fanning}$, wetted perimeter P_w, acceleration due to gravity g, height for calculating hydrostatic pressure z and pressure P. The absolute mass flow rate $\dot{m}$ is used to account for flow reversal in the momentum equation. Additionally the derivative $\frac{\partial \dot{m}}{\partial t}$ is often set to zero to avoid fast pressure oscillations [Casella, Leva, et al., 2003]. The timescale of these oscillations is quite often connected to the speed of sound in that medium.

The energy balance is [Casella, Leva, et al., 2003]:

$$\rho A_{XS}\frac{\partial h_{enthalpy}}{\partial t} + A_{XS}\frac{\partial P}{\partial t} = -\rho A_{XS} u \frac{\partial h_{enthalpy}}{\partial x} + q_{wall}\frac{A_{wall}}{L} \tag{2.19}$$

In Equation 2.19, in addition to the previously defined terms for Equation 2.17 and 2.18, I define $h_{enthalpy}$ for specific enthalpy per unit mass for the control volume, average fluid velocity u and q_{wall}, the heat flux received or lost through lateral surfaces. $h_{enthalpy}$ is denoted as such to prevent confusion with heat transfer coefficient, also commonly denoted h. $\frac{A_{wall}}{L}$ is the heated surface area of the wall A_{wall} per unit length, where L is the length of the 1D geometry. For tubes, $\frac{A_{wall}}{L} = P_w$ [Casella, Leva, et al., 2003].

Review of Previous Work in Solving Momentum and Mass Balances

Previously in my master's thesis, I discussed how to solve the mass and momentum equations in real-time using the Boussinesq approximation [Ong, 2023]. For this, I ignored density changes in the fluid except when it comes to calculating hydrostatic pressure for the momentum equation [Bejan, 2013]. Using the Boussinesq approximation, the mass balance reduces to [Ong, 2023]:

$$\dot{m}_i = constant \tag{2.20}$$

For a circuit of fluid components in series, the mass flowrate through each component ($\dot{m}_i$) is the same. For momentum balance, I essentially solved a discretised form of equation 2.18 where the pressure oscillations are ignored and each component is represented by a control volume. When I considered the source terms such as pumps in the system, I arrived at Equation 2.21 [Ong, 2023]:

$$\Delta P_{change} - \sum_i^n \Delta P_{hydrostatic\ i} - \sum_i^n \Delta P_{source\ i} = -\sum_i^n \frac{1}{2}\frac{{\dot{m}_i}^2}{\rho A_{XS,i}^2}(f_{darcy,i}\frac{L_i}{D_i} + K_i) \tag{2.21}$$

In Equation 2.21, ΔP_{change} is the total pressure change across a set of n fluid components connected in series, $\sum_i^n \Delta P_{hydrostatic\ i}$ is the summation of hydrostatic pressures across the same n fluid components, $\sum_i^n \Delta P_{source\ i}$ is the summation of pressure sources (such as pumps) across these n components. $\dot{m}_i$ is the mass flow rate across component i, $A_{XS,i}$ is the cross sectional area for component i, $f_{darcy,i}$ is the Darcy friction factor, L_i is the length of component i, D_i is the hydraulic diameter for component i and K_i is the form loss coefficient for component i. In contrast to Equation 2.18, I solved the mass flow reversal issue in my solver by using a conditional "if" block rather than the modulus sign. I then used iterative procedures to solve for $\dot{m}$ across series of fluid components (branches) and therefore obtained the mass flow rate of the system [Ong, 2023]. I reproduced a figure from my master's thesis in Figure 2.10 with some small edits for the reader's convenience [Ong, 2023]:

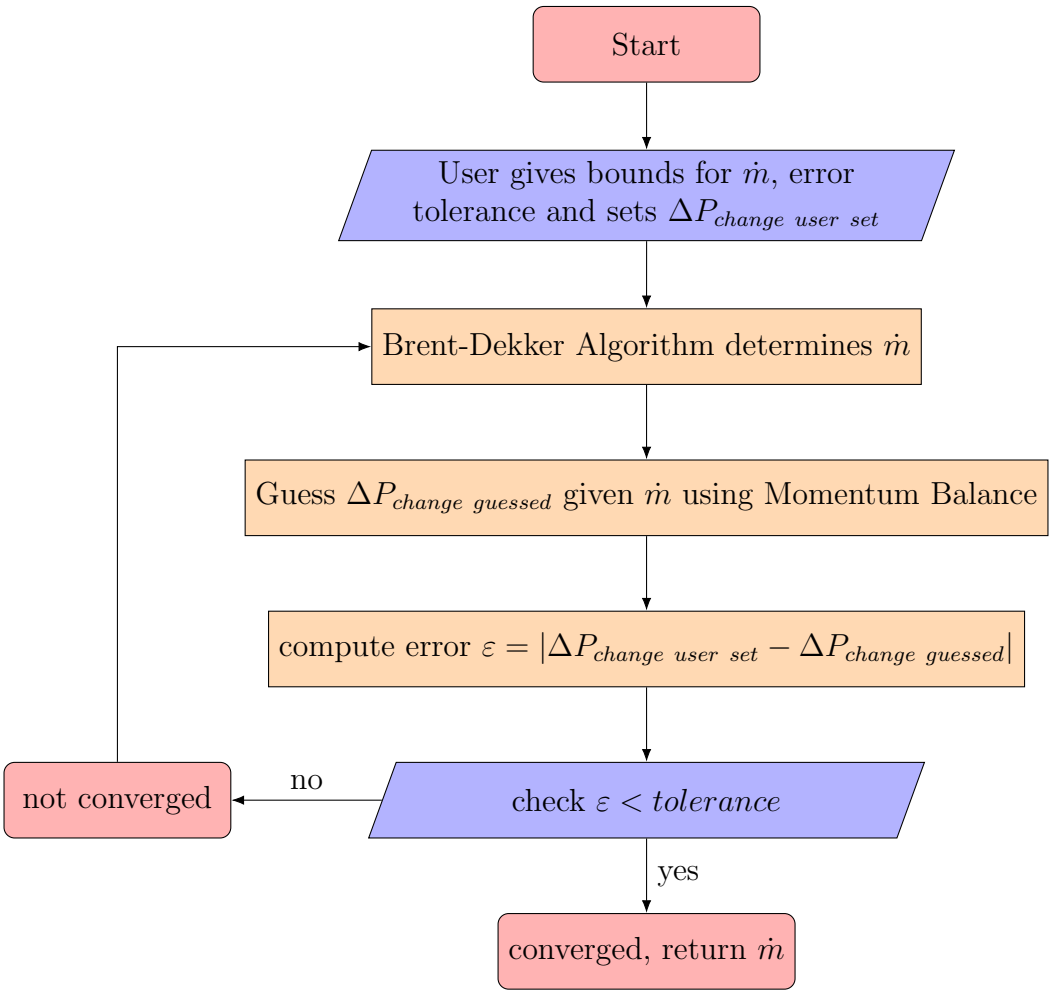


Figure 2.10: Iterative Solution Procedure for a Series of Fluid Components

Figure 2.10 describes how the momentum and mass balances are iteratively solved over each branch of fluid components. The momentum balance in Figure 2.10 refers to Equation 2.21. In the special case that the branch is a complete loop, $\Delta P_{change} = 0\ Pa$. Much of the detail, including the Brent-Dekker algorithm, and the methodology to solve flow in parallel branches, is already covered in my master's thesis [Ong, 2023], and briefly in Figure 2.7 for the parallel branch solver. Hence, I will not repeat the discussion here.

As discussed in my master's thesis, I intended to solve the momentum and mass balances the same way such that the mass and momentum balances are not tightly coupled to the heat balance equations at each time step [Ong, 2023]. This procedure is known as operator splitting [MacNamara and Strang, 2016]. We can visualise the overall process for solving mass, momentum and energy balances using operator splitting using Figure 2.11:

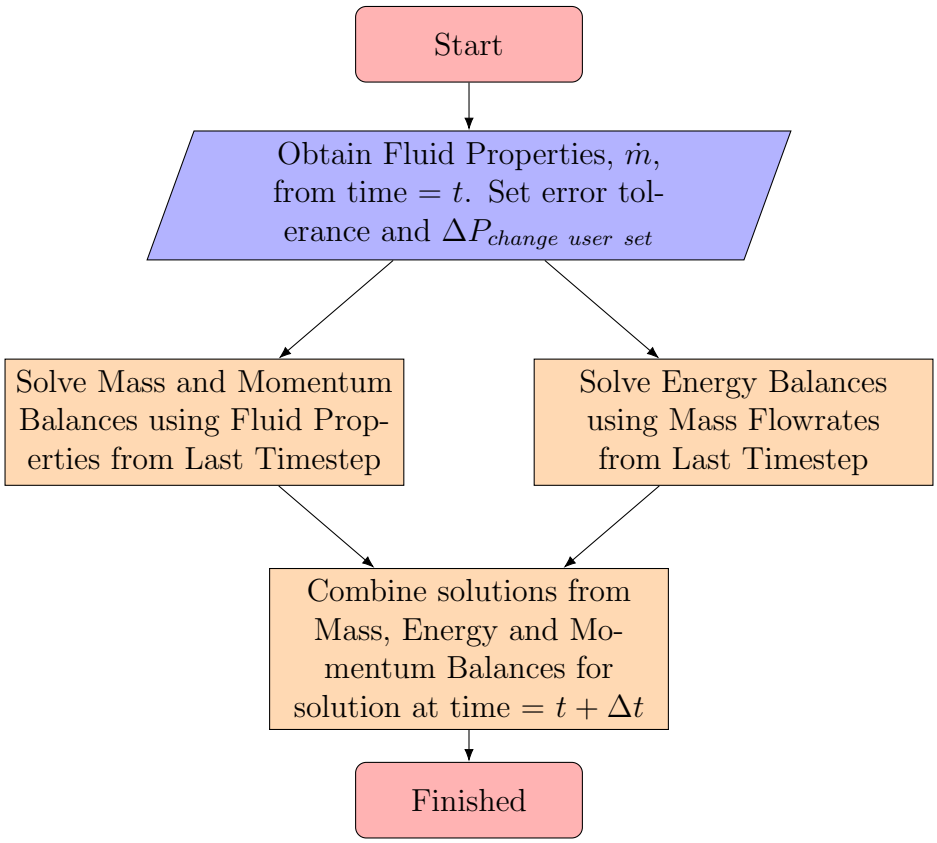


Figure 2.11: Operator Splitting Solution Procedure at Each Timestep

By using operator splitting, I can assign a computation thread to the mass and momentum balance, and then assign several other computing threads to solve the energy balance equations. Thus, I can parallelise my calculations. Having parallelised computing capability is important because it helps me achieve real-time simulation capability. Real-time simulation is, in turn, a must for Digital Twin construction. In addition to parallelisation, operator splitting also allows me to decouple the solution procedure of energy balances and the mass and momentum balances at each time step. Therefore, for liquid flows such as those in CIET, I can re-use the mass and momentum balance procedures in my master's thesis and consider the solution procedure for energy balances at each timestep separately. For this dissertation, I focus only on solving the energy balances in Figure 2.11 assuming that the mass flow rates from the previous timestep are already provided.

Discretisation of Energy Balance Equations

Derivation of Discretised 1D Heat Transport Equations for Fluid To start our discussion on energy balance, let us revisit Equation 2.18:

$$\rho A_{XS} \frac{\partial h_{enthalpy}}{\partial t} - A_{XS} \frac{\partial P}{\partial t} = -\rho A_{XS} u \frac{\partial h_{enthalpy}}{\partial x} + q_{wall} \frac{A_{wall}}{L} \tag{2.19}$$

For liquids, it is common to assume that $c_v \approx c_p$, where c_v is constant volume heat capacity and c_p is constant pressure heat capacity [Perry and Green, 2015]. Therefore, the enthalpy change is approximately equal to the internal energy change. By using this approximation, we can neglect the pressure term in Equation 2.18.

$$\rho A_{XS} \frac{\partial h_{enthalpy}}{\partial t} = -\rho A_{XS} u \frac{\partial h_{enthalpy}}{\partial x} + q_{wall} \frac{A_{wall}}{L}$$

We can then discretise the equation using an explicit or implicit time marching scheme. For simplicity, I show the explicit time marching scheme first.

$$\begin{aligned} \rho A_{XS} \frac{h^{t+\Delta t}_{enthalpy,x} - h^{t}_{enthalpy,x}}{\Delta t} = &- \rho A_{XS} u \frac{h^{t}_{enthalpy,x+0.5\Delta x} - h^{t}_{enthalpy,x-0.5\Delta x}}{\Delta x} \\ &+ q^{t}_{wall} \frac{A_{wall}}{L} \end{aligned}$$

Where $h^{t}_{enthalpy,x}$ is $h_{enthalpy}$ at length coordinate x and time t. Δt is the user specified time step, and Δx is the user specified mesh length. If we consider the length of just one 1D control volume, $\Delta x = L$. We can multiply Δx to all sides of the equation to obtain an equation in terms of control volume masses m_{CV}. Where $m_{CV} = \rho A_{XS} \Delta x$. Moreover, I take the prevailing mass flow rate $\dot{m} = \rho A_{XS} u$. Note also that based on the operator split method, we use $\dot{m}$ from the last time step. So $\dot{m}$ is assumed to be known or solved for. With these in mind, resulting equation becomes:

$$\begin{aligned} m_{cv} \frac{h^{t+\Delta t}_{enthalpy,x} - h^{t}_{enthalpy,x}}{\Delta t} = &- \dot{m} (h^{t}_{enthalpy,x+0.5\Delta x} - h^{t}_{enthalpy,x-0.5\Delta x}) \\ &+ q^{t}_{wall} A_{wall} \end{aligned}$$

Basically, over each control volume, we sum the advection terms of enthalpy entering the control volumes, and the heat fluxes entering the control volume due to the user set boundary conditions. For q_{wall} at a solid-fluid interface with a known wall temperature, the heat flux can be expressed as:

$$q_{wall} = -h_{wall}(T_x - T_{wall})$$

When we include this back in our derivation:

$$\begin{aligned} m_{cv} \frac{h^{t+\Delta t}_{enthalpy,x} - h^{t}_{enthalpy,x}}{\Delta t} = &- \dot{m} (h^{t}_{enthalpy,x+0.5\Delta x} - h^{t}_{enthalpy,x-0.5\Delta x}) \\ &+ h^{t}_{wall} A_{wall,x} (T^{t}_{wall} - T^{t}_{x}) \end{aligned}$$

To find T^{t}_{x}, we shall need a way to relate $h_{enthalpy}$ to T. Additionally, we shall need to find h_{wall}. Now, $h_{enthalpy}$ is, in general, a function of T:

$$h_{enthalpy}(T) = \int_{T_{ref}}^{T} c_p(T)dT$$

Where c_p is the constant pressure heat capacity of the fluid. For Therminol VP-1, we can use existing correlations [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015] to find:

$$h_{enthalpy}(T) = \int_{T_{ref}}^{T} 1518 + 2.82(T^{\circ}C)dT$$

These expressions have already been given in literature [Zou, R. Hu, and Charpentier, 2019]:

$$h_{enthalpy}(T) = h_{ref} + 1518(T - T_{ref}) + \frac{2.82}{2}(T^2 - T_{ref}^2)$$

For Therminol VP-1, I take $h_{ref} = 0\ J/kg$ at $T_{ref} = 20^{\circ}C$. Hence, I obtain Equation 2.22:

$$h_{enthalpy}(T^{\circ}C)(J/kg) = 1518(T) + \frac{2.82}{2}(T^2) - 30924 \tag{2.22}$$

Now, when it comes to calculating h_{wall} and the heat flux at the walls of the fluid volume, I found it convenient to think of the problem in the form of a thermal circuit as was done in literature [Perry and Green, 2015]. In this regard:

$$R_{thermal} = \frac{\Delta T}{Q}$$

Where Q is the heat flow in watts, and ΔT is the respective temperature difference. Thermal resistance in SI units is in $\frac{Kelvin}{Watts}$. We can also define a thermal conductance $H_{thermal}$:

$$H_{thermal} = \frac{1}{R_{thermal}} = \frac{Q}{\Delta T}$$

$H_{thermal}$ in SI units is $\frac{Watts}{Kelvin}$. For the convective thermal conductance at the solid-fluid interface, we can express this as:

$$H_{thermal,convection,x} = h_{wall} A_{wall,x}$$

h_{wall} can be found via Nusselt Number (Nu) correlations depending on its geometry, Re and Pr. In the explicit time marching scheme, Nu is based on Re, Pr of the last time step. This is assumed to be known at the time of calculation. We shall discuss the correlations used when reviewing models of CIET's Heater specifically. If we were to substitute expressions for thermal conductance in our equations, we arrive at:

$$m_{cv}\frac{h^{t+\Delta t}_{enthalpy,x}-h^{t}_{enthalpy,x}}{\Delta t} = -\dot{m}(h^{t}_{enthalpy,x+0.5\Delta x}-h^{t}_{enthalpy,x-0.5\Delta x}) + H^{t}_{thermal,convection,x}(T^{t}_{wall}-T^{t}_{x})$$

Now, when solving the convection heat transport equations, we present them as a matrix with i elements representing i control volumes. We can express the heat transfer equations in this form as well:

$$m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i}-h^{t}_{enthalpy,i}}{\Delta t} = \dot{m}(-h^{t}_{enthalpy,i}+h^{t}_{enthalpy,i-1}) + H^{t}_{thermal,convection,i}(T^{t}_{wall,i}-T^{t}_{i})$$

Where $h^{t}_{enthalpy,i}$ is the $h_{enthalpy}$ at control volume i and time t, $T^{t}_{wall,i}$ is the temperature of the wall at control volume i and time t. T^{t}_{i} is the temperature of the fluid control volume in control volume i at time t. For a control volume, we assume it is well mixed. Therefore, the flows leaving the control volume have the enthalpy of $h^{t}_{enthalpy,i}$. The enthalpy entering the control volume has the enthalpy of $h^{t}_{enthalpy,i-1}$ assuming flow is in the positive x direction. If flow is in the negative x direction, such that $\dot{m}$ <0 kg/s, we should use:

$$m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i}-h^{t}_{enthalpy,i}}{\Delta t} = |\dot{m}|(-h^{t}_{enthalpy,i}+h^{t}_{enthalpy,i+1}) + H^{t}_{thermal,convection,i}(T^{t}_{wall,i}-T^{t}_{i}) \tag{2.23}$$

$|\dot{m}|$ is the absolute value $\dot{m}$. This can be programmed in using a conditional statement or "if" code block based upon the how $\dot{m}$ compares to 0 kg/s. Now, we have discussed most of the terms in the fluid convection equations except for $T_{wall,i}$. We shall discuss treatment of $T_{wall,i}$ in the following paragraphs.

Derivation of Discretised 1D Heat Transport Equations for Solid Now, in these 1D control volume equations for heat transport, we assume that these fluids are in contact with some wall with temperature $T_{wall,i}$. $T_{wall,i}$ can be set by the user to be a constant wall temperature boundary condition. Alternatively, as discussed earlier, the user may set $q_{wall,i}$, the wall heat flux at control volume i directly, thus setting the boundary condition. However, this is quite unrealistic. It is more common in literature to model the pipe walls with some finite thermal inertia and thermal diffusivity α [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015; De Wet and Per F Peterson, 2020]. Thus, we shall need to consider conduction heat transport equations as well. For this purpose, we can use the Equation 2.23, but set the convection terms to zero by setting $\dot{m} = 0$ kg/s. Additionally, the heat flow into the fluid $H_{thermal,convection,i}(T^{t}_{wall,i}-T^{t}_{i})$ is now heat lost from the wall $H_{thermal,conduction,i}(-T^{t}_{i}+T^{t}_{wall,i})$. With these considerations, we get:

$$m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} = H^{t}_{thermal,conduction,i}(-T^{t}_{i} + T^{t}_{wall,i})$$

In this form, there are a few things of note. Firstly, the thermal conductance between the control volume centre and the wall must be calculated. This depends on the geometry of the system. For pipes, it is often common to use cylindrical geometry. For cylindrical geometry, the thermal resistance can be expressed as [Perry and Green, 2015]:

$$R_{thermal,cylinder} = \frac{\ln(r_2/r_1)}{2\pi kL}$$

Where $R_{thermal,cylinder}$ measures the thermal resistance between two radii r_1 and r_2 where $r_2 > r_1$. ln is the natural logarithm, k is the thermal conductivity, and L is the cylinder length. Based on the thermal resistance model, we note that the magnitude of heat flow at the wall can be expressed as:

$$|Q_{wall,i}| = \frac{|T_{solid,i} - T_{fluid,i}|}{R_{thermal,cylinder,i} + R_{thermal,convection,i}} = H^{t}_{thermal,conduction,i}(|-T^{t}_{i} + T^{t}_{wall,i}|)$$

$$|Q_{wall,i}| = H^{t}_{thermal,solid\leftrightarrow fluid,i}|T_{solid,i} - T_{fluid,i}|$$

Where:

$$H^{t}_{thermal,solid\leftrightarrow fluid,i} = \frac{1}{R^{t}_{thermal,cylinder,i} + R^{t}_{thermal,convection,i}}$$

Thus, we can eliminate the wall temperature T_{wall}, which is an unknown, in both equations. In the conduction equation, this is expressed as:

$$m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} + H^{t}_{thermal,solid\leftrightarrow fluid,i}(-T^{t}_{i} + T^{t}_{fluid,i})$$

Next, the conduction term is neglected because we implicitly assumed in Equation 2.19 that the heat advection term dominates the heat transport such that heat diffusion or conduction term is relatively negligible. This relative importance of heat advection to diffusion is quantified by the Peclèt Number (Pe) where Pe = Re Pr. When nondimensionalising the heat transport equation, it can be shown that the conduction term scales as $\frac{1}{\mathrm{Pe}}$ [Bejan, 2013; April Novak, 2020]. This derivation is already well known in literature [Bejan, 2013; April Novak, 2020] and I will not repeat it here. Hence, conduction in the axial direction is often neglected in literature as it was for Equation 2.18 [Casella, Leva, et al., 2003]. For us, we shall need to add the conduction term back into the equation. This is because we want a system of equations which works regardless of Pe. The conduction term is described by Fourier's Law [Bejan, 2013; Perry and Green, 2015]. I show Fourier's law for a 1D geometry:

$$Q = -kA\frac{\partial T}{\partial x} \tag{2.24}$$

Where Q is heat flow in watts, k is thermal conductivity, A is the area for which heat is transported. Fourier's law can be discretised using an explicit time marching scheme. If we do so and consider heat flows between control volume i and control volume $i-1$, we obtain:

$$Q_i^t = -k_i^t A_i^t \frac{T_i^t - T_{i-1}^t}{\Delta x}$$

Now, Q_i^t is the heat flow into control volume i at time t, k_i^t is the relevant thermal conductivity to be used for control volume i at time t, A_i^t is the relevant heat transfer area for control volume i and time t. The area can usually be assumed to be constant as the geometry should not change significantly with time in this case. For convenience, we can once again use thermal resistance and conductance $H_{thermal,conduction}$:

$$Q_i^t = -H_{thermal,conduction,i}^t (T_i^t - T_{i-1}^t)$$

Now, if k_i^t was constant, our lives would be easier. However, k_i^t is usually temperature dependent. Therefore, the thermal resistance between two adjacent nodes interfacing through heat transfer area A can be described as:

$$R_{thermal,conduction,i\leftrightarrow(i-1)}^t = \frac{\Delta x}{2k_i^t A} + \frac{\Delta x}{2k_{i-1}^t A}$$

Where $R_{thermal,conduction,i\leftrightarrow(i-1)}$ is the thermal resistance between two adjacent nodes i and $i-1$. Therefore, the conduction heat flow between two adjacent bodies becomes:

$$Q_{i\leftrightarrow(i-1)}^t = -H_{thermal,conduction,i\leftrightarrow(i-1)}^t (T_i^t - T_{i-1}^t)$$

Now, for 1D bodies, we can assume that the control volume has two adjacent control volumes with which to conduct heat to and from. This is true for all control volumes except those found at the boundaries.

$$\begin{aligned} m_{cv}\frac{h_{enthalpy,i}^{t+\Delta t} - h_{enthalpy,i}^t}{\Delta t} &= H_{thermal,solid\leftrightarrow fluid,i}(-T_i^t + T_{fluid,i}^t) \\ &+ H_{thermal,conduction,(i-1)\leftrightarrow i}^t(-T_i^t + T_{i-1}^t) \\ &+ H_{thermal,conduction,(i+1)\leftrightarrow i}^t(-T_i^t + T_{i+1}^t) \end{aligned}$$

For the first control volume:

$$\begin{aligned} m_{cv}\frac{h_{enthalpy,1}^{t+\Delta t} - h_{enthalpy,1}^t}{\Delta t} &= H_{thermal,solid\leftrightarrow fluid,1}^t(-T_1^t + T_{fluid,1}^t) \\ &+ H_{thermal,conduction,2\leftrightarrow 1}^t(-T_1^t + T_2^t) \\ &+ Q_{axial\ boundary,start}^t \end{aligned}$$

Where $Q_{axial\ boundary,start}$ is the user specified boundary condition at the start of the control volume array.

For the last control volume:

$$\begin{aligned} m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} &= H^{t}_{thermal,solid\leftrightarrow fluid,i}(-T^{t}_{i} + T^{t}_{fluid,i}) \\ &+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i-1}) \\ &+ Q^{t}_{axial\ boundary,end} \end{aligned}$$

Where $Q_{axial\ boundary,end}$ is the user specified boundary condition at the end of the control volume array. One simplifying assumption we can make is for there to be zero heat flux in direction of the flow. Alternatively, we may choose to neglect axial heat flux in comparison to the other terms. In either case, $Q_{axial\ boundary,start}$ and $Q_{axial\ boundary,end}$ are neglected.

Now, for the heater specifically, we can add in a volumetric heat generation term $Q^{t}_{gen,i}$. We can also consider that the solid loses some heat to the air at temperature T^{t}_{air}. We can add these two terms in to obtain:

$$\begin{aligned} m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} &= H^{t}_{thermal,solid\leftrightarrow fluid,i}(-T^{t}_{i} + T^{t}_{fluid,i}) \\ &+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i-1}) \\ &+ H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i+1}) \\ &+ Q^{t}_{gen,i} + H^{t}_{thermal,air\leftrightarrow i}(-T^{t}_{i} + T^{t}_{air}) \end{aligned}$$

Where $H_{thermal,air\leftrightarrow i}$ is the thermal conductance between the air and the centre of the control volume. If we substitute $H_{thermal,air\leftrightarrow i}$ into Equation 2.23, we obtain:

$$\begin{aligned} m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} &= |\dot{m}|(-h^{t}_{enthalpy,i} + h^{t}_{enthalpy,i+1}) \\ &= H^{t}_{thermal,solid\leftrightarrow fluid,i}(-T^{t}_{i} + T^{t}_{solid,i}) \end{aligned}$$

Discretised 1D Heat Transport Equations for Conjugate Heat Transfer (CHT)
When solving these two 1D heat transport equations, we essentially solve for the following layout of control volumes:

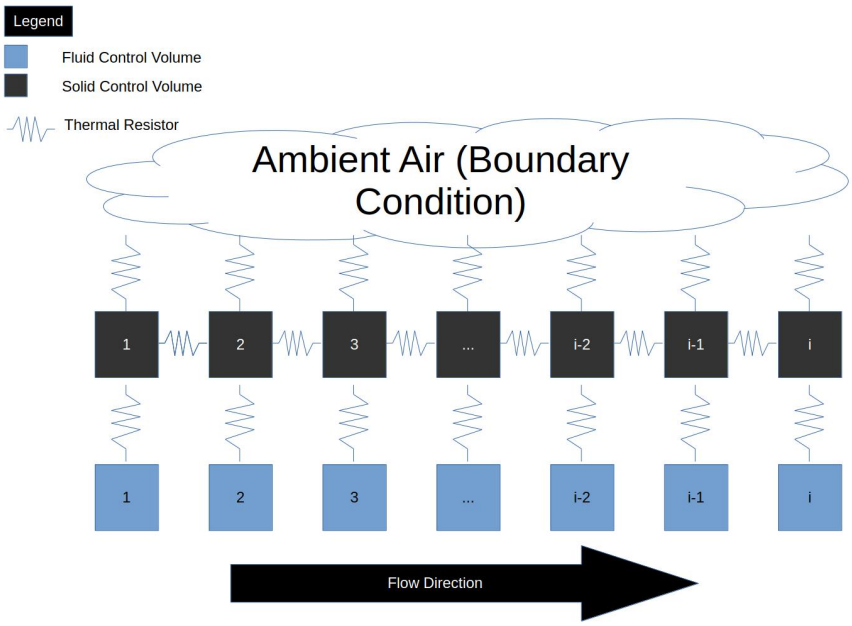


Figure 2.12: Control volume arrangement for 1D conjugate heat transfer

In Figure 2.12, I visualise the mesh as a thermal resistance network. In Figure 2.12 the solid control volumes are axially coupled to adjacent solid control volumes in a 1D mesh. The 1D mesh of solid control volumes is then laterally coupled to the 1D mesh of fluid control volumes via thermal resistances. The relevant thermal resistances or conductances are then calculated based on geometry, thermal conductivity and Nusselt number correlations. The solid control volumes are also laterally coupled to ambient air boundary conditions of constant temperature. The relevant thermal resistances are based on the pipe dimensions and the heat transfer coefficient to air. For the heat transfer coefficient to air, a value of typically 6 to 20 $W/(m^2 \cdot K)$ has been used [De Wet and Per F Peterson, 2020].

Solution Procedure for a Heat Generating Pipe Figure 2.12 can be used to describe a Conjugate Heat Transfer problem in a heat generating pipe. This is important to consider because CIET's Heater geometry bears some resemblance to a heat generating pipe [Nicolas Zweibaum, 2015]. In this case, the fluid control volumes are centred at the radial coordinate $r = 0$, whereas the solid control volumes are centred at the radial coordinate $r = \frac{OD+ID}{4}$ where ID is the internal diameter of the pipe, and OD is the outer diameter of the pipe.

If I were to start solving for transient heat transfer in this geometry, I would need to set m_{CV} first. For simplicity, I will just divide the inner fluid cylinder and the outer solid ring uniformly over i nodes. In so doing:

$$m_{CV,solid} = \frac{m_{solid}}{i}$$

$$m_{CV,fluid} = \frac{m_{fluid}}{i}$$

As mentioned in this section, I also assume that the mass flow rate $\dot{m}$ has already been obtained because of operator splitting. Next, I set the initial conditions of the control volumes. For simplicity, I set all the control volumes to some uniform temperature $T_{initial}$. I also set the temperature of the pipe inlet to some inlet temperature T_{inlet} so that the enthalpy entering the pipe from the left is h_{inlet}. The boundary condition at the pipe exit is adiabatic so that no heat flux flows through the solid at either end of the solid control volumes. Likewise, I also set the heat flux for the fluid control volumes at both ends of the pipe to be zero. The only means of transferring heat along the path of fluid flow is advection.

With this setup, I would then take the following steps in each time step to solve for the temperature profile for the fluid and solid:

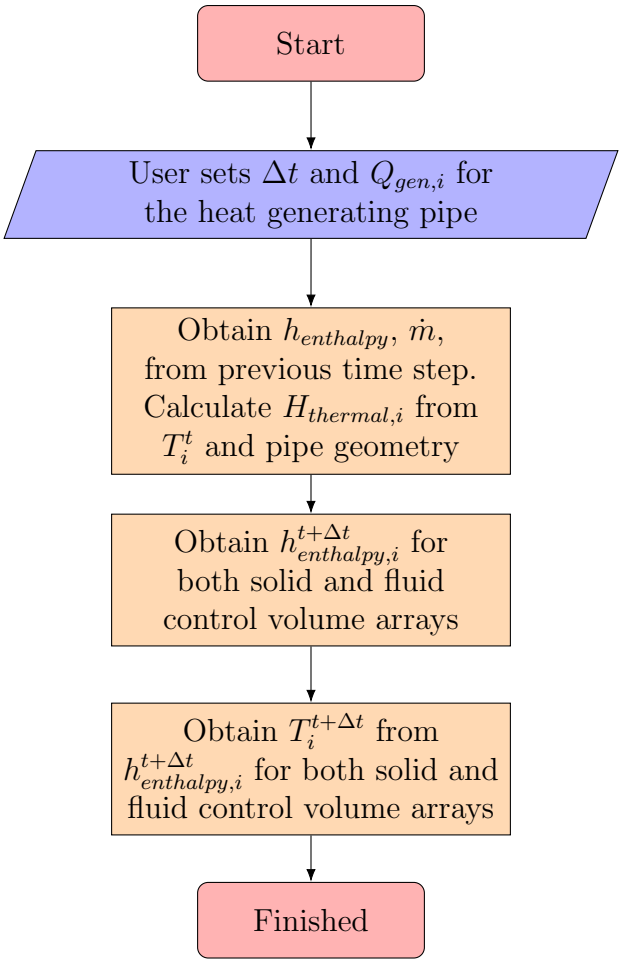


Figure 2.13: Conjugate Heat Transfer Solution Procedure for Two Laterally Coupled 1D Arrays of Control Volumes using Explicit Time Scheme at each Time Step

When obtaining $T_i^{t+\Delta t}$ from $h_{enthalpy,i}^{t+\Delta t}$, we may use some solver to obtain the temperature iteratively as we may not always know how enthalpy varies with temperature. In my master's thesis, I used the Brent-Dekker method to solve for $\dot{m}$ given a user set pressure change over a series of pipes [Ong, 2023]. For this dissertation, I use the same Brent-Dekker method to obtain the $T_i^{t+\Delta t}$ from $h_{enthalpy,i}^{t+\Delta t}$.

Generalising Conjugate Heat Transfer (CHT) Equations to Include more Radial Nodes and Piping Insulation While we have discussed solving a simple CHT problem for a heat generating pipe involving two 1D arrays of control volumes, meshes presented in literature are usually more complex. We may have more than one set of radial nodes in the solid mesh modelling the pipe [De Wet and Per F Peterson, 2020]. Moreover, we may also have insulation surrounding the pipe to minimise parasitic heat loss as found in many of the pipes in CIET [Nicolas Zweibaum, 2015]. In such a case, I can simply couple more 1D arrays of solid control volumes laterally.

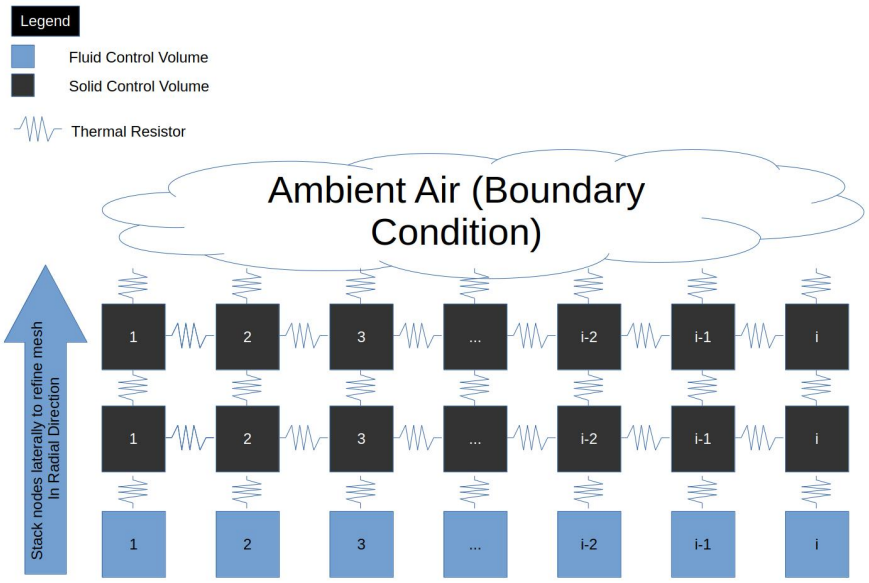


Figure 2.14: Control volume arrangement for 1D conjugate heat transfer with extra stacks of laterally coupled arrays of solid control volumes

Given these new stacks of control volumes, we will need to add more terms to account for the extra laterally coupled control volumes added in the radial direction. In Figure 2.14, there are two arrays of solid control volumes. The array connected to the ambient air boundary condition is the "outer stack". The other array is the "inner stack". The energy balance over the inner stack can be written as:

$$
\begin{aligned}
m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} &= H^{t}_{thermal,solid\ inner\ stack \leftrightarrow fluid,i}(-T^{t}_{i} + T^{t}_{fluid,i}) \\
&+ H^{t}_{thermal,solid\ inner\ stack \leftrightarrow solid\ outer\ stack,i}(-T^{t}_{i} + T^{t}_{solid\ outer\ stack,i}) \\
&+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i-1}) \\
&+ H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i+1})
\end{aligned}
$$

We can also change the energy balance for the outer stack based on the thermal resistance diagram in Figure 2.14. I will skip this for the sake of brevity. Once the equations are set up, then we repeat the steps in Figure 2.13 except that we include this extra stack of laterally coupled control volumes.

If we wish to add insulation, we need only add yet another stack of solid control volumes to represent the pipe insulation. Or, if we'd like, the outer stack shown in Figure 2.14 could also represent solid insulation added to the pipe. In either case, the form of the equations to be solved remains the same. Only the thermal conductance and thermal masses need to differ depending on how many stacks the user wishes to add.

Generalised form of Discretised Energy Balance for 1D Arrays of Control Volumes In general, a 1D array of control volumes can be laterally coupled to any number of other 1D control volume arrays. We can generalise this by adding a summation term for this lateral coupling:

$$
\begin{aligned}
m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} &= \sum_{j}^{N} H^{t}_{thermal,self\ \leftrightarrow\ j,i}(-T^{t}_{i} + T^{t}_{j,i}) \\
&+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i-1}) \\
&+ H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i+1}) \\
&+ \sum Q^{t}_{gen,i} + \sum Q^{t}_{boundary\ conditions,i}
\end{aligned}
$$

Where $H_{thermal,self\ \leftrightarrow\ j,i}$ represents the thermal conductance from the current control volume (called "self") to other control volumes (the other array is labelled j) at array index i. $T^{t}_{j,i}$ is the temperature of the other laterally coupled control volume at array index i.

For fluid control volumes, we can add the convection term back. The resulting energy balance can be expressed as Equation 2.25:

$$
\begin{aligned}
m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} &= \sum_{j}^{N} H^{t}_{thermal,self \leftrightarrow j,i}(-T^{t}_{i} + T^{t}_{j,i}) \\
&+ \dot{m}^{t}(-h^{t}_{enthalpy,i} + h^{t}_{enthalpy,i-1}) \\
&+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i-1}) \\
&+ H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i+1}) \\
&+ \sum Q^{t}_{gen,i} + \sum Q^{t}_{boundary\ conditions,i}
\end{aligned}
\tag{2.25}
$$

Where $Q_{boundary\ conditions,i}$ represents the heat added or removed due to user set boundary conditions at array index i. Equation 2.25 is the discretised form of the differential energy balance over any fluid control volume provided fluid flow traverses in the positive x direction. Of course, we may neglect the conduction terms in direction of the fluid flow if Pe is high enough. We can also apply Equation 2.25 to arrays of control volumes by setting $\dot{m} = 0$ kg/s.

Equation 2.25 needs to be applied to specified geometries in order to be useful. As I intend to model CIET's Heater, we shall first explore the geometries and meshes relevant for CIET's Heater in the following subsections. Additionally, Equation 2.25 is also uses an explicit time marching scheme. This can prove problematic for both numerical stability and accuracy should Δt get too large. We discuss this more in detail in the next section after we discuss CIET's Heater where we review solver instability issues.

CIET Heater

Let us now discuss the development of the heat transfer library with special focus on CIET's Heater. CIET's heater basically consists of an electrically heated pipe with some insert at the center of the pipe which aids heat transfer but does not generate heat. This heated section of the pipe was connected to two electrodes with copper cables. To aid the reader in understanding CIET a photograph of CIET's heater as of 2023 is provided in Figure 2.15:

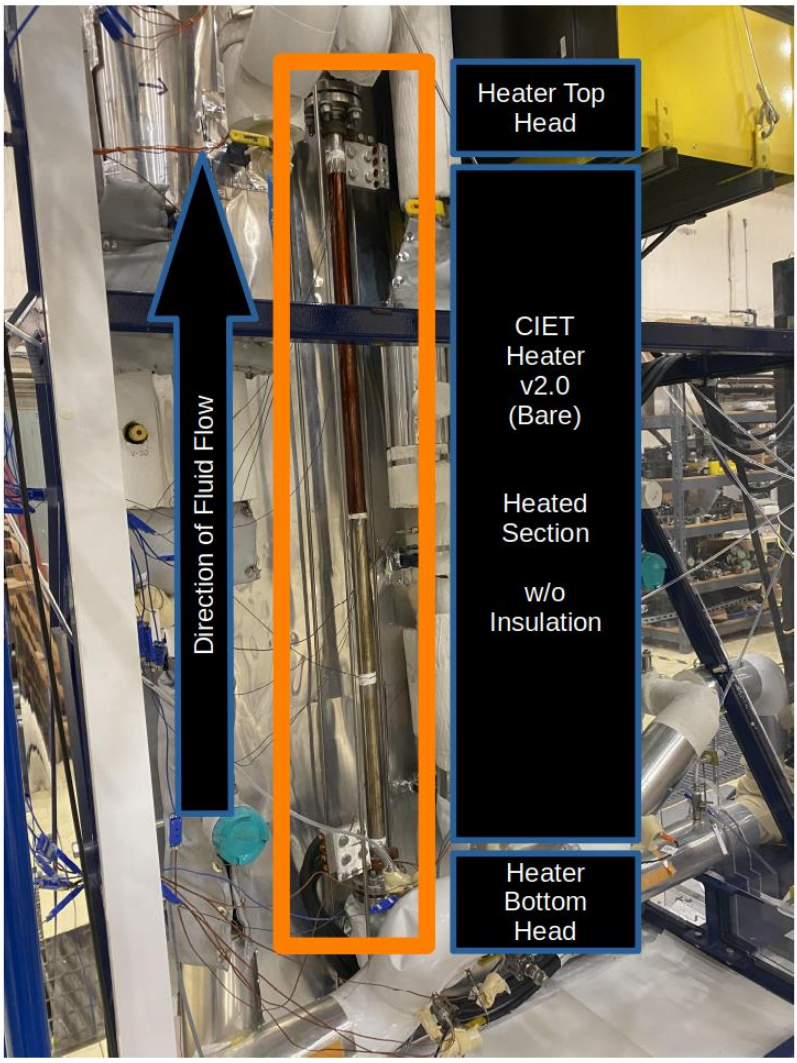


Figure 2.15: CIET Heater v2.0 Photograph with Labels of Heater Top and Bottom Head, the brown material on the heated section is Kapton tape to allow Infra Red Imaging

A simplified schematic of Figure 2.15 showing the r-z plane cross section is shown in Figure 2.16:

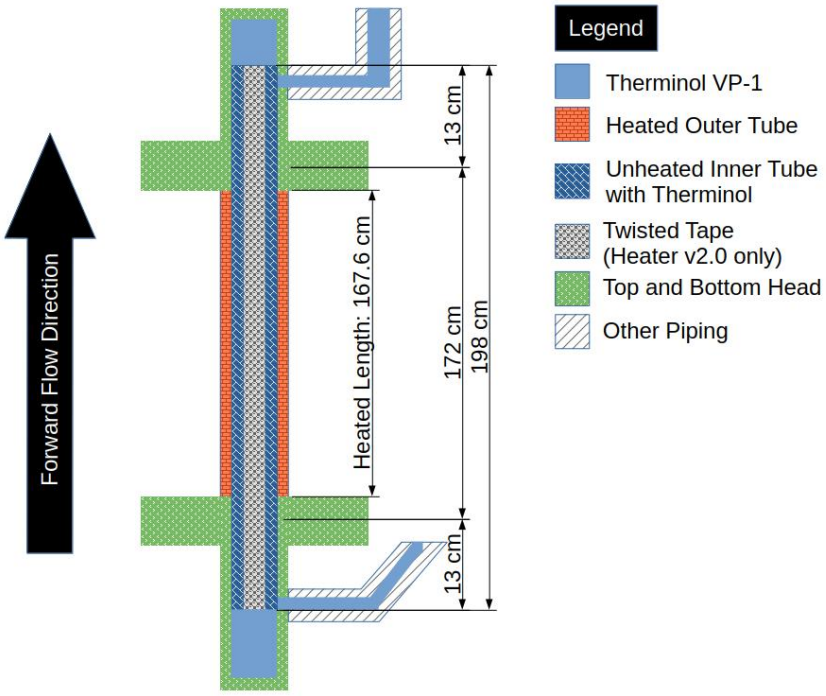


Figure 2.16: CIET Heater v2.0 Cross Section Simplified Schematic in the r-z Plane, Dimensions from De Wet's Dissertation [De Wet and Per F Peterson, 2020], (Not to Scale)

In Figure 2.16 [1], r is the radial (horizontal) coordinate, and z is the axial (vertical) coordinate. In CIET's Heater, fluid flows from its bottom to its top as shown in Figure 2.15. To prevent current from the electrically heated section from traversing the rest of the loop, the heated section is electrically insulated from the rest of CIET. This insulating material (PolyTetraFluoroEthane or PTFE) is sandwiched between flanges made of SS 304L at the heater's top and bottom [De Wet and Per F Peterson, 2020]. In literature, the top and bottom assemblies with the gaskets, tees and various components shown in Figure 2.15 are known as the heater top and bottom heads respectively [Nicolas Zweibaum, 2015; De Wet and Per F Peterson, 2020]. A close up of the heater bottom head is shown in Figure 2.17:

[1] For any such drawing or figure in this dissertation, as long as I have produced it and have not taken it from another person's work, you may reproduce it with citation, as in copy and paste the figure as a whole.

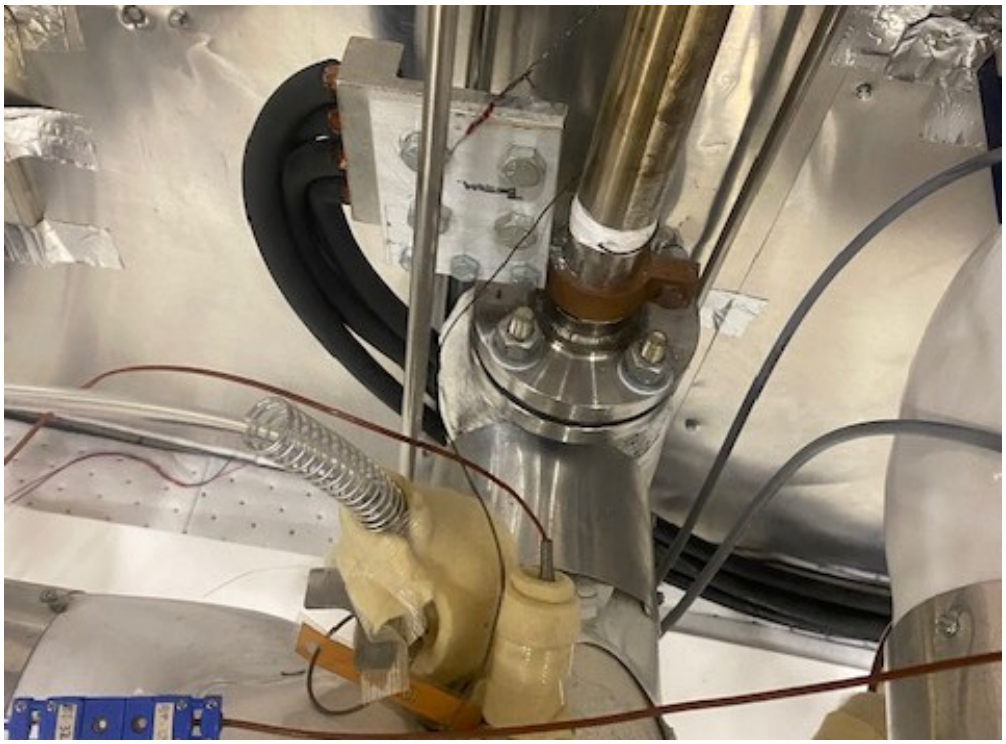


Figure 2.17: CIET Heater v2.0 Bottom Head

While I could not find detailed schematics for the CIET's top and bottom head in literature, I found some of its engineering drawings done using SOLIDWORKS and some of the pdf files in the shared archives of the Thermal Hydraulics Laboratory. These drawings were done mainly by AJ Gubser in 2012. I tried simplifying some of these drawings of the heater top and bottom assemblies by removing schematics of most of the washers and nuts present in the drawings to give the reader a rough idea of its geometry. I did this because these components are not explicitly modelled in numerical simulations of CIET present in literature [Nicolas Zweibaum, 2015; De Wet and Per F Peterson, 2020; Zou, R. Hu, and Charpentier, 2019]. I also included the addition of the twisted tape and perforated inner tube to illustrate that both of these inserts extended into the bottom head and top heads of the heater as mentioned in literature [De Wet and Per F Peterson, 2020]. The resulting drawing is shown in Figure 2.18:

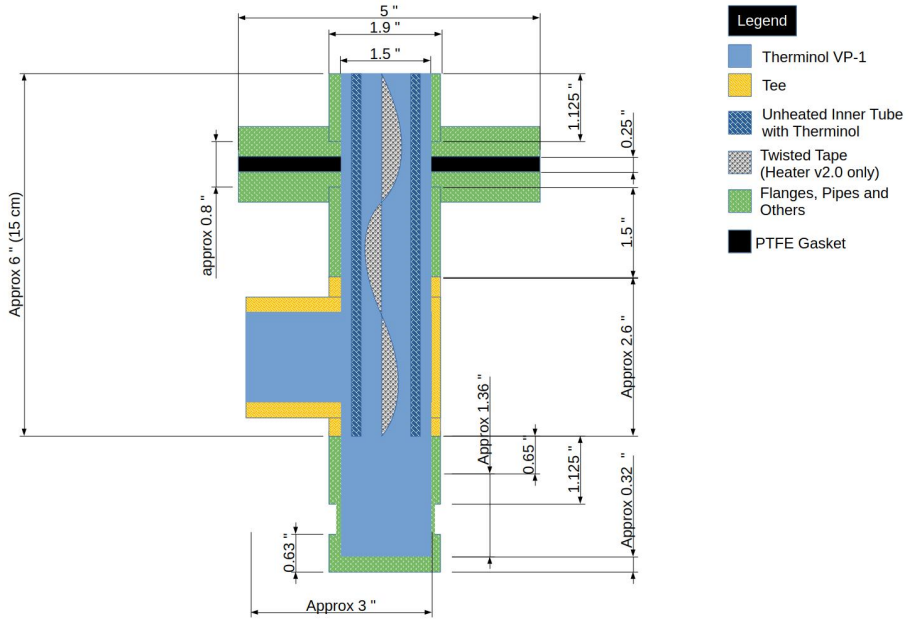


Figure 2.18: CIET Heater v2.0 Heater Top and Bottom Head Simplified Schematic Drawings, Measurements are in Inches unless otherwise Stated (Not to Scale)

Figure 2.18 was produced using some of the known dimensions from these drawings by Gubser. Now, Figure 2.18 highlights that some of the entrained fluid resides within the heater top and bottom heads. This was important for consideration detailed transient modelling of the heater [De Wet and Per F Peterson, 2020]. In constructing Figure 2.18, I found that the lengths of the entrained fluids were not explicitly labelled. While the Computer Aided Design (CAD) models were in the archive, they were SOLIDWORKS files. I was not familiar with SOLIDWORKS at the time of writing to read and manipulate the SOLIDWORKS files. To expedite the writing process, I used a ruler to measure out the drawings produced in the pdf files. I used this approximate approach to construct Figure 2.18 because Figure 2.18 was only meant to help the reader understand how components inside the heater top and bottom head are roughly arranged. Therefore, I approximated the lengths of some of these dimensions in Figure 2.18. These lengths are labelled in Figure 2.18 as approximate values.

To verify that these approximate length measurements are reasonably accurate for modelling the heater, I consider that the total length of the twisted tape and perforated tubing as recorded in literature is 198 cm and that the heated section is 167.6 cm [De Wet and Per F Peterson, 2020]. Since the twisted tape and perforated tubing extends through the heated section as well as both heater top and bottom heads, the total length of the twisted tape and perforated tubing inserts is 30.4 cm. In Figure 2.18, the length of the twisted tape and perforated tubing in the top or bottom heads is about 6 inches (15.24 cm). If we sum up the length in both top and bottom heads, we arrive at 30.5 cm. This is within measure-

ment error of 30.4 cm where the measurement error is ±0.1 cm for a ruler. Therefore, the approximate lengths are reasonably accurate when considering the length of the perforated tube and twisted tape.

Based on the design schematics found in the Thermal Hydraulics Lab acrhives, most of the components in the heater top and bottom head are identical, including the PTFE Flange Gasket, the McMaster-Carr 4335T56 1-1/2“ pipe size tee, and the McMaster-Carr 44685K55 Flanges. Thus, Figure 2.18 can describe most of the components present in both the heater top and bottom heads. However, only the heater bottom head contains a valve (V-10) the bottom of the heater bottom head assembly that one can use for draining and filling CIET. Valve V-10 and the extra appendages used for draining and filling procedures are not shown in Figure 2.18. However, if these appendages were likely to be put in Figure 2.18, Valve V-10 and some of its associated piping would be connected to the bottom of the heater bottom head in Figure 2.18. These specific valves and components are not modelled explicitly in literature. However, one will often find that when modelling the heater top and bottom heads numerically, different models with different dimensions are used for the heater top and bottom heads [Zou, R. Hu, and Charpentier, 2019; De Wet and Per F Peterson, 2020; Nicolas Zweibaum, 2015]. This is something we shall explore more in detail later in this subsection.

It is evident from Figure 2.18 that detailed modelling of CIET's Heater can be quite complicated if we consider all its constituent components. It is still complicated even if we do not consider its washers, nuts and individual flanges. Therefore, we may want to use simplified models to model the CIET Heater as a whole. To model CIET's Heater using simplified models, it is important to review literature about previous models of CIET's Heater. Now, there are several models of the Heater available based on different design iterations of the heater. To help the reader make sense of the models in literature, we shall first review the several versions of CIET's Heater as seen in literature.

CIET's Heater has gone through a few design iterations and changes since its initial construction. The main changes the reader should note are in Figure 2.19:

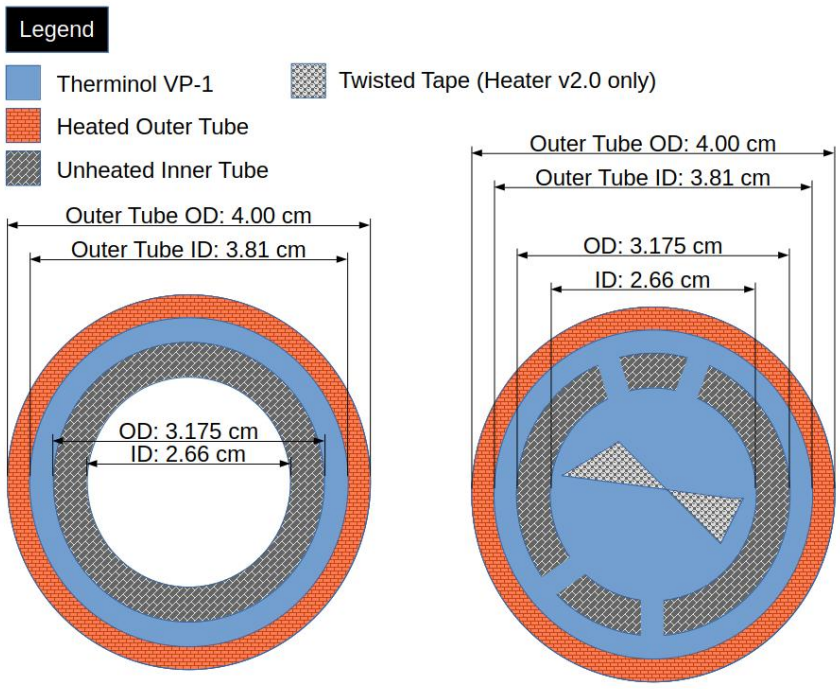


Figure 2.19: CIET Heater x-y Cross Section of the Heated Tube without Insulation (Not to Scale), Dimensions are from De Wet's Dissertation [De Wet and Per F Peterson, 2020]

Figure 2.19 is an x-y cross section of the heated tube, where the x-y plane has its normal parallel to the direction of flow in CIET's Heater. The initial heater was called CIET Heater v1.0. Its cross section is shown on the left of Figure 2.19. CIET Heater v1.0 was a thermally insulated electrically heated pipe which contained an inner pipe such that Dowtherm A (also known as Therminol-VP1) flowed in the annular region between the inner and outer pipe [Poresky, 2017], whereas no fluid flowed within the inner tube [De Wet and Per F Peterson, 2020]. CIET Heater v1.0 was also covered in 5cm thick fibreglass insulation to lower parasitic heat loss [Poresky, 2017]. While CIET Heater v1.0 has fibreglass insulation, I present cross sections of CIET Heater v1.0 without insulation in Figure 2.19 so that the figure is less cluttered and the main changes can be emphasised. After some time with CIET Heater v1.0, the inner pipe was replaced with perforated tube and metallic twisted tape to improve heat transfer characteristics [Lukas, Kendrick, and P. Peterson, 2017]. This change allowed fluid to flow in the centre of the heated pipe. This version is known as CIET Heater v2.0 shown on the right of Figure 2.19.

After some time, however, the fibreglass insulation on CIET Heater v2.0 was damaged and subsequently removed [De Wet and Per F Peterson, 2020]. Without the insulation, the heater was essentially "bare". Hence, I call this version of the Heater "CIET Heater v2.0 Bare". This version of CIET's Heater is the version shown in Figure 2.15. More details

of its construction are found in literature [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014; Lukas, Kendrick, and P. Peterson, 2017] and detailed schematic drawings are presented in De Wet's dissertation [De Wet and Per F Peterson, 2020]. These drawings will not be reproduced here as the emphasis is on heater modelling rather than its design and construction. Based on these CIET Heater design iterations, CIET's Heater was modelled in RELAP [Nicolas Zweibaum, 2015], the System Analysis Module (SAM) [Zou, R. Hu, and Charpentier, 2019] and Transform [De wet, Per F. Peterson, and Greenwood, 2019]. The RELAP version used CIET Heater v1.0 as their reference model. Two different SAM models of CIET's two heater designs were developed based on the frequency response experiments by Poresky on both CIET Heater v1.0 and CIET Heater v2.0 [Poresky, 2017]. De Wet's model in Transform was mainly based on CIET v2.0 Bare [De Wet and Per F Peterson, 2020].

In this subsection, it is important to study these models in more detail for the sake of constructing a real-time thermal hydraulics simulation of the CIET Heater. Moreover, it is also important to review some of the experimental data or experimentally derived correlations available which is important in model validation. For this purpose, we shall first consider the information in literature regarding the heated sections in CIET Heater v1.0 and CIET Heater v2.0 for both the insulated and bare versions. Thereafter, we also consider methods to model the heater top and bottom heads as shown in Figure 2.15.

CIET Heater v1.0 Heated Section

CIET Heater v1.0 is important to consider because it was used in initial natural circulation and forced circulation experiments [Nicolas Zweibaum, 2015] as well as frequency response tests [Poresky, 2017]. These datasets was used to validate RELAP5 models [Nicolas Zweibaum, 2015] and SAM models [Zou, R. Hu, and Charpentier, 2019]. For validating transient models of the heater, the forced circulation transient dataset [Nicolas Zweibaum, 2015; Zou, R. Hu, and Charpentier, 2019] used for RELAP5 and SAM modelling is most useful. This particular dataset is in the time domain and shows how the outlet temperature of CIET Heater v1.0 responds to step changes in heater power. However, this dataset is published in literature in the form of a graph rather than some easy to use correlation. This graph has to be read manually. Poresky's frequency response data was also published in the form of a graph. While Poresky also provided a theoretical derivation of a transfer function in his work [Poresky, 2017], it was not presented in a form which allowed me to easily validate my models with the experimental data provided. In contrast to these, De Wet's dissertation presents the transient response data of CIET Heater v2.0 in the form of a transfer function [De Wet and Per F Peterson, 2020]. To obtain the step response of CIET Heater v2.0, I can simply subject the Transfer Function to a step input and plot its output. Therefore, validating CIET Heater v2.0 is somewhat easier since the transfer functions were available. Nevertheless, The RELAP and SAM models are still useful to study because they are relatively simple to implement and they can serve as a reference for which I can use to design my heater model in this dissertation.

Mesh for CIET Heater v1.0 Heated Section First, we want to study the mesh layout and boundary conditions for SAM and RELAP models of CIET Heater v1.0. In these models, the heater is typically modelled in these codes as an array of fluid control volumes or nodes thermally coupled to the heated outer tube and unheated inner tube. A typical model one might find in SAM or CIET is found in Figure 2.20:

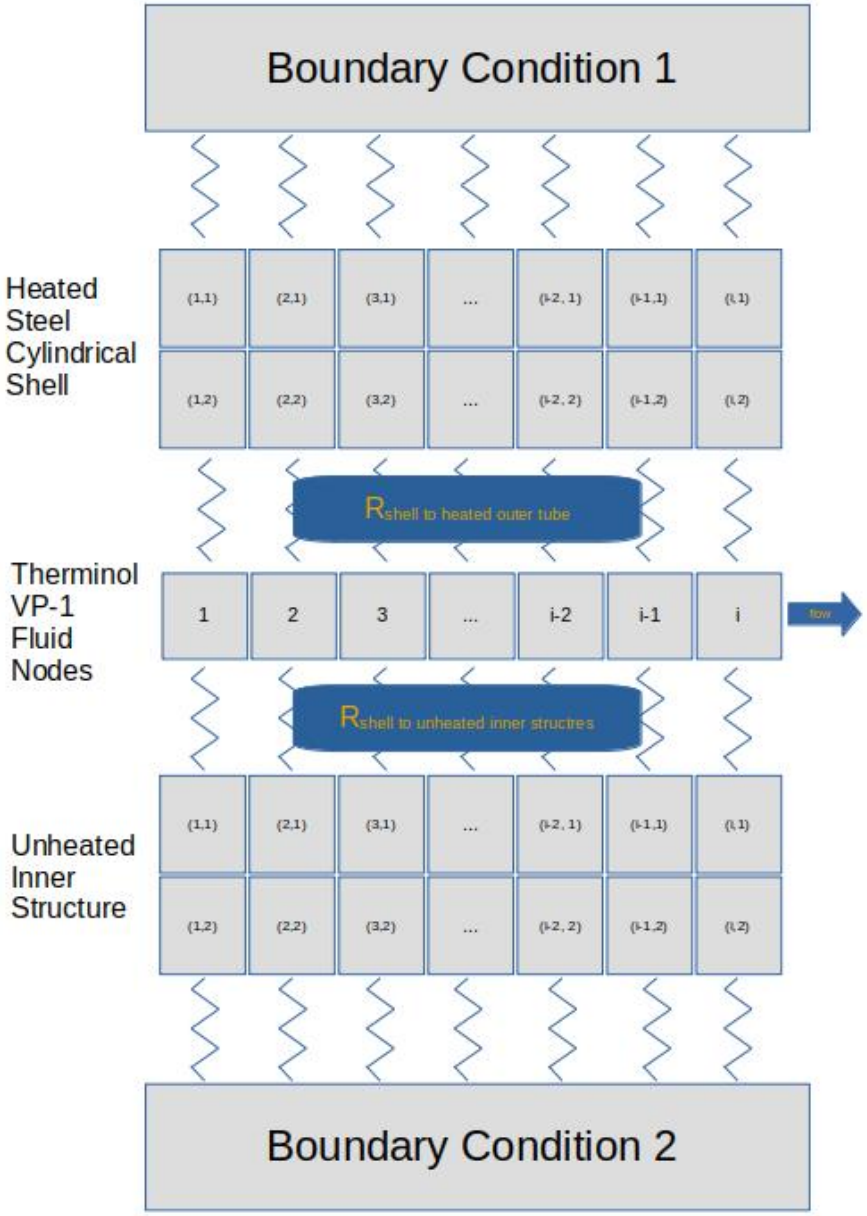


Figure 2.20: CIET Heater (Heated Section) Typical Mesh Layout

Figure 2.20 shows a heater mesh setup for i axial nodes specific to the heated section. I have used a thermal resistor network as this is a convenient way to visualise how the control

volumes are laterally coupled to each other. However, unlike in Figure 2.14, I do not draw the thermal resistors for all control volumes due to space constraints. In Figure 2.20, the Therminol fluid nodes are coupled with a convective thermal resistance to the heated cylindrical shell and unheated inner structure. Each of these solid heat structures are modelled with a set of axial nodes and radial nodes. Likewise, the Therminol VP-1 region, modelled as an annular fluid cylinder, is represented as an array of axial nodes. The heated outer tube and unheated inner tube are then individually coupled laterally to their respective boundary conditions.

In Figure 2.20, a nodalisation scheme with i axial nodes and two radial nodes for the heated outer cylindrical shell and unheated inner structure is shown as an example. Typically, the number of radial nodes is usually on the order of two or three, while the number of axial nodes used is on the order of 10. For example, the RELAP model used three radial nodes for each heat structure [Nicolas Zweibaum, 2015] and 15 axial nodes [Nicolas Zweibaum, 2015]. Likewise, the SAM model also used 15 axial nodes [Zou, R. Hu, and Charpentier, 2019]. In my best estimates, these 15 nodes were evenly spaced from each other. I also found it relatively difficult to find exactly how the nodes were radially spaced as these codes are closed source. Publications for this specific detail pertaining to these specific codes were also relatively difficult to find. To expedite code development, I just assumed an even radial (r) spacing. That is if a pipe wall thickness was 2 inches, and I discretised this into two radial nodes, the inner radial node would be one inch thick and the outer radial node would also be one inch thick. I can test whether this works well enough when I perform verification and validation tests later on.

It is also worthwhile noting that in the case of steady state heat transfer, the number of axial nodes and radial nodes does not heavily impact heat transfer phenomena provided they stay within certain ranges. In the case of RELAP5 models used for natural circulation, the steady state mass flowrate varies by less than 1% provided that the number of axial nodes is in the range of 5 to 55 and the number of radial wall nodes is in the range of 3 to 20 as the temperature profiles in the axial direction is approximately linear [Nicolas Zweibaum, 2015]. This suggests that for steady state heat transfer, increasing the number of nodes axially would not change the heater outlet temperature significantly.

Boundary Conditions for CIET Heater v1.0 Heated Section For these heater models, flow would then go through the fluid nodes or control volumes as shown in Figure 2.20 before traversing to other components. The inner heat structure, an inner tube in the case of CIET Heater v1.0, would usually be connected to an adiabatic boundary condition, this is boundary condition 2 in Figure 2.20. In contrast, the outer heated tube is usually connected to a user set constant heat flux wall boundary condition as it was done in SAM [Zou, R. Hu, and Charpentier, 2019]. The wall heat flux was dynamically calculated and usually accounted for parasitic heat losses from the heater [Zou, R. Hu, and Charpentier, 2019]. The exact values of the heat flux are set by the user and differ between simulation runs. Examples of such simulation runs can be found in Zweibaum's and Zou's publications

[Nicolas Zweibaum, 2015; Zou, R. Hu, and Charpentier, 2019].

I assume from these experiments that the wall heat flux is modelled as uniform throughout the heated tube. This is indeed the simplest approach. However, for CIET Heater v1.0, the published steady state forced and natural circulation data used to validated SAM and RELAP models does not include information about heater surface temperature profiles. The same can be said for the transient heat transfer simulations with step inputs of heater power. Only the bulk heater outlet and inlet temperatures are recorded as shown in SAM [Zou, R. Hu, and Charpentier, 2019]. Given the lack of validation data for CIET Heater v1.0, a viable approach is indeed to use a uniform heat flux.

In reality however, the heat flux across the heater tube may not truly be uniform. The resistive heater outer tube made of 304L stainless steel has a temperature dependent resistivity $\rho_{\mu\Omega-cm}$ given as [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014]:

$$\rho_{\mu\Omega-cm}(T) = 0.0612 \cdot T(^{\circ}C) + 73.109$$

Here, I differentiate the resistivity $\rho_{\mu\Omega-cm}$, from density ρ using the units of resistivity $\mu\Omega - cm$ to prevent confusion. Given this state of affairs, the we need to consider the effect of resistivity in the surface bulk temperature profile. Since the heater is electrically heated, we can consider the differential power produced in a small section of the heated tube:

$$dP_{watts} = I^2_{heater} dR_{ohms\ heater}$$

Where I_{heater} is heater current and P_{watts} is power in watts.

$$dP_{watts} = I^2_{heater} \frac{\rho_{\mu\Omega-cm}(T)dx}{A_{XS}}$$

We can find a general expression for the amount of heat generated in a section of length Δx between lengths x_1 and x_2 using:

$$P_{watts\ \Delta x} = \frac{I^2_{heater}}{A_{XS}} \int_{x_1}^{x_2} \rho_{\mu\Omega-cm}(T)dx$$

If we wish to find the power distribution, we can first integrate this expression over the whole of the heater length L_{heater}.

$$P_{watts\ total} = \frac{I^2_{heater}}{A_{XS}} \int_{0}^{L_{heater}} \rho_{\mu\Omega-cm}(T)dx$$

$$\frac{P_{watts\ \Delta x}}{P_{watts\ total}} = \frac{\int_{x_1}^{x_2} \rho_{\mu\Omega-cm}(T)dx}{\int_{0}^{L_{heater}} \rho_{\mu\Omega-cm}(T)dx}$$

Evidently, to find the heater power distribution, we shall need to find the heater temperature profile first $T(x)$. For early models of CIET Heater v1.0, models were not validated against the heater surface temperature profile. We may likewise use the same approach and assume a roughly uniform heat generation along the axial direction of the heated section for simplicity.

Nusselt Number and Thermal Inertia for CIET Heater v1.0 Heated Section Now, let us review some important heat transfer parameters for CIET Heater v1.0. Again, while I am not modelling CIET Heater v1.0, it was good to review some of this information to check if any of it was useful for my heater model.

CIET Heater v1.0 had its heat transfer coefficient modelled with equation 2.26 [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014; De Wet and Per F Peterson, 2020]:

$$\mathrm{Nu} = \begin{cases} 8 & \text{for Re } > 2300 \\ 5.44 + 0.034 * \mathrm{Re}^{0.082} & \text{for Re } < 2300 \end{cases} \tag{2.26}$$

For thermal inertia of the heater, as well as the conduction thermal resistances, calculations can be performed using the following design and performance parameters in table 2.3 [De Wet and Per F Peterson, 2020]:

Name	Value
Outer Tube Outer Diameter	4.0 cm
Outer Tube Inner Diameter	3.81 cm
Outer Tube Heated Length	167.6 cm
Outer Tube Heat Transfer Area	2007 cm^2
Inner Tube Outer Diameter	3.175 cm
Inner Tube Inner Diameter	2.66 cm
Flow Area	3.48 cm^2
Fluid Volume Height	198 cm
Metal Material	304L Stainless Steel
Heat Transfer Fluid	Dowtherm-A or Therminol VP-1
Main Fluid Volume	690.2 cm^3

Table 2.3: Heater 1.0 Design and Performance Parameters [De Wet and Per F Peterson, 2020]

One should note that in Table 2.3, the heated section length is 167.6 cm. This is based on De Wet's Transform models [De Wet and Per F Peterson, 2020]. In contrast, earlier models of CIET Heater v1.0 show a heated section length of 163.83 cm [Nicolas Zweibaum, 2015], this is a difference of 1.5 inches. This same heated section length was used for the SAM model [Zou, R. Hu, and Charpentier, 2019]. To unravel why this discrepancy exits, we can investigate some engineering drawings in De Wet's dissertation [De Wet and Per F Peterson, 2020]. One computer aided design (CAD) engineering drawing describes the lengths between

the electrodes. These electrodes, shown in Figure 2.17, are connected to the heated tube near the top and bottom heads. The next drawing of interest is a CAD drawing of elevations in the electric heater line, including the top and bottom heads. This CAD drawing was meant to emphasise the relative hydrostatic heights z relative to the pump centreline [De Wet and Per F Peterson, 2020]. Figure 2.21 overlays information from these CAD drawings:

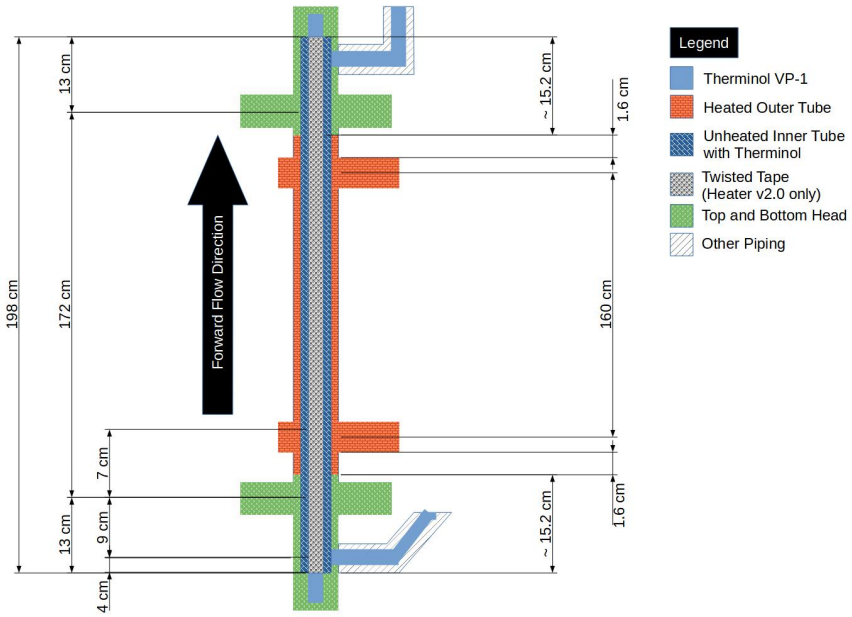


Figure 2.21: Heated Length CAD Diagram Comparison Drawings [De Wet and Per F Peterson, 2020] (Not to Scale)

On Figure 2.21, I also overlay estimates for the length of the twisted tape within the top and bottom head shown in Figure 2.18, which is itself based on Gubser's CAD engineering drawings of the heater top and bottom head. This allows me to make sense of the several length scales presented for the CIET v1.0 Heater.

Figure 2.21 shows length measurements from the which shows a centre to centre distance of 160 cm (62.992 inches [De Wet and Per F Peterson, 2020]) between the two electrodes connected to the heated tube. The distance from the electrodes to the their respective top and bottom heads is 1.6 cm (0.63 inches [De Wet and Per F Peterson, 2020]). If we add the length between the electrodes and the two lengths from the electrodes to their respective top and bottom head, we obtain a length of 163.2 cm. This is relatively close to 163.83 cm used in the RELAP model. Of course, the widths of the electrodes themselves is difficult to find in literature. For the widths, I looked into some of AJ Gubser's other drawings in 2012 for the heater electrodes and found that they are 0.875 inches or 2.22 cm, which means that the centre to edge distance of the electrode is 1.11 cm. Based on these numbers, however, I could

only reason that the RELAP5 163.2 cm length measurement, plus an estimated addition of the electrode lengths. I myself was puzzled as to why this was the case, as the lengths do not add up to 163.83 cm. Hence I ommitted this data from Figure 2.21. Nevertheless, this model worked well for RELAP5 and it was validated against experimental data. I saw no reason to continue worrying about a discrepancy of a few centimetres.

For De Wet's model, the length scale of 167.6 cm was likely based on CAD drawings of the heater's elevations on the left of Figure 2.21. From the CAD drawing, the length of the heated section plus top and bottom heads was shown directly as 198 cm. The top and bottom head lengths exclude some appendages shown in Figure 2.18. As I based Figure 2.18 on Gubser's same drawings of the heater top and bottom head, I concluded that the length from the bottom of the tee to the heated section was approximately 6 inches or 15.24 cm. One can sum up these lengths and verify that:

$$167.6cm + 15.2cm + 15.2cm = 198cm$$

Therefore, De Wet likely arrived at his measurement of the heater length not by measuring the length of the heater tube directly, but by subtracting the lengths of the heater top and bottom head from the combined length of 198 cm. This is called the fluid height in Table 2.3. De Wet's work was focused more on CIET Heater v2.0 rather than CIET Heater v1.0. His models in Transform were also based on CIET Heater v2.0 data [De Wet and Per F Peterson, 2020]. In those models, he used a heated length of 167.6 cm and also validated those against experimental data [De wet, Per F. Peterson, and Greenwood, 2019; De Wet and Per F Peterson, 2020]. Since heater models for v1.0 and v2.0 have been validated against experimental data in CIET using both heated lengths 163.83 cm and 167.6 cm, both of them worked well for their respective purposes.

For this dissertation, I could choose a build a CIET Heater v2.0 based on the heated section length of 163.83 cm and validate this model against CIET Heater v2.0 data. If the model matches experimental data, it would be valid as well. Given the choice of RELAP5 heater dimensions and Transform heater dimensions, I chose the RELAP5 version where the heater length was 163.83 cm [Nicolas Zweibaum, 2015] as a starting point due to its simplicity relative to the Transform model. This is because the Transform model also accounts for additional thermal inertia in the top and bottom heads as well as structural supports in CIET [De wet, Per F. Peterson, and Greenwood, 2019; De Wet and Per F Peterson, 2020]. I left this more accurate modelling of thermal inertia to future work. For now, all I needed was a validated model of CIET Heater v2.0 which could reproduce the experimentally determined Heater v2.0 bulk outlet temperature. Secondly, I also needed the model to reasonably match the transient response. The validation process will be discussed later. For now, let us continue reviewing important properties of CIET Heater v1.0.

If we wish to model CIET Heater v1.0, we can use heater length dimensions in the RELAP model [Nicolas Zweibaum, 2015] and SAM model [Zou, R. Hu, and Charpentier, 2019] which are simpler in comparison to the Transform model. The heater dimensions, along with the thermophysical property data for stainless steel can be used to model the

heater. These thermophysical properties were provided in tabulated form for the RELAP5 and SAM models [Nicolas Zweibaum, 2015; Zou, R. Hu, and Charpentier, 2019]. These values are shown in Table 2.4:

Temperature (K)	k ($W\ m^{-1}\ K^{-1}$)	c_p ($J\ kg^{-1}\ K^{-1}$)
250	14.31	443.3375
300	14.94	457.0361
350	15.58	469.4894
400	16.21	480.6974
450	16.85	490.66
500	17.48	500.6227
700	20.02	526.7746
1000	23.83	551.6812

Table 2.4: Thermophysical Properties of Steel [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015]

The density ρ used was 8030 $kg\ m^{-3}$ [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015]. These properties are also relevant to CIET Heater v2.0 since the heated tube is that same as that in CIET Heater v1.0.

Now, during the course of test driven development, I also found it convenient and useful to have the thermophysical properties of steel in the form of correlations rather than tabulated data. With tabulated data, I would need to program an interpolation scheme or construct some form of spline to obtain the thermophyiscal properties at the desired temperatures. For correlations, there is no need to do this. For 304L stainless steel, these correlations can be used to determine thermal conductivity and heat capacity [Graves et al., 1991]:

$$k\left(\frac{W}{m\cdot K}\right) = 7.9318 + 0.023051\ T(K) - 6.4166 * 10^{-6}\ T(K)^2 \tag{2.27}$$

$$c_p\left(\frac{J}{g\cdot K}\right) = 0.4267 + 1.700 * 10^{-4}\ T(K) + 5.200 * 10^{-8}\ T(K)^2 \tag{2.28}$$

For steady state tests, the thermal inertia of CIET Heater v1.0 was not considered. However, for transient tests, thermal inertia becomes important. However, while experimental data for thermal inertia exists [Zou, R. Hu, and Charpentier, 2019], the experimental data was presented in the form of a graph rather than a convenient to use transfer function. Therefore, I decided not to use CIET Heater v1.0 experimental data in the initial testing and development phase for this dissertation. Instead, I focused more on CIET Heater v2.0

and CIET Heater v2.0 Bare. CIET Heater v1.0 can be considered in future work. Nevertheless, several of the thermophysical property data for steel and Therminol VP-1, as well as the dimensions of the heated tube remain the same between CIET Heater v1.0 and CIET Heater v2.0. These can be used in the CIET Heater v2.0 model.

CIET Heater v2.0 Heated Section

CIET Heater v2.0 was bare shown in earlier parts of this section in Figure 2.15 as well as Figure 2.16.

Again, my goal is to develop an approximate model of CIET's Heater v2.0 to test my simulated neutronics feedback controller. I consider endeavour successful if I can match the simulation data to experimentally derived correlations and data. We shall first review the validation data available, with more emphasis on the steady state and transient data for the heater outlet temperatures. Next, we shall consider how to model CIET Heater v2.0 by exploring how previous models were constructed, as well as the thermophysical data used in modelling CIET Heater v2.0.

CIET Heater v2.0 Validation Data

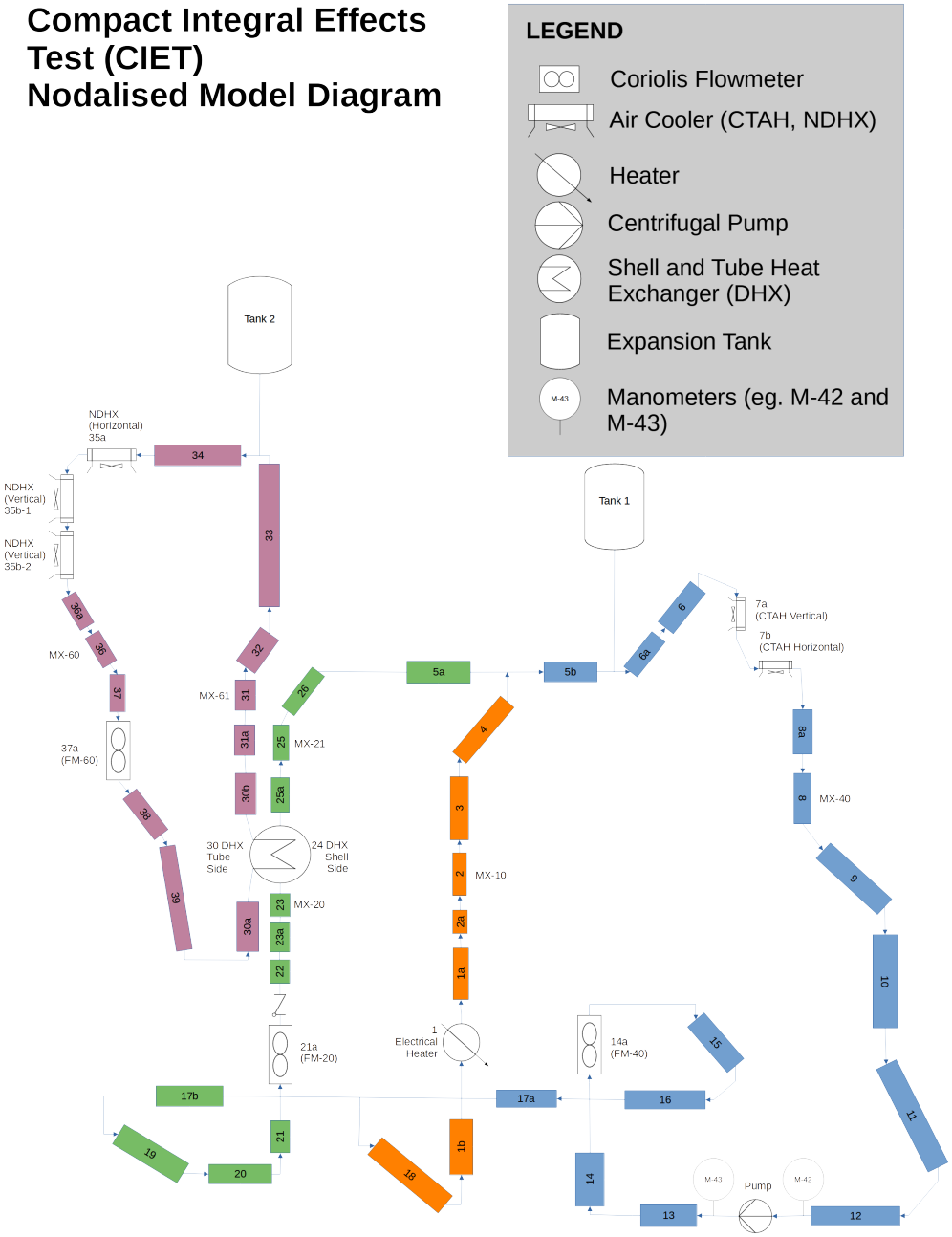
Empirical Transfer Function I chose to model CIET Heater v2.0 because its transient data was used to construct an empirical transfer function. This transfer function can then be used to reconstruct the transient experimental data obtained during those tests. The transfer function for the heater power to heater outlet temperature for CIET Heater v2.0 Bare is provided by De Wet [De Wet and Per F Peterson, 2020]:

$$G(s) = e^{-4s}\frac{3.217 * 10^{-5}s^3 + 6.675 * 10^{-7}s^2 + 1.139 * 10^{-8}s + 2.423 * 10^{-11}}{s^5 + 0.2251s^4 + 0.01688s^3 + 0.0003548s^2 + 3.057 * 10^{-6}s + 1.632 * 10^{-9}} \quad (2.29)$$

Equation 2.29 was obtained using a pseudorandom binary sequence (PRBS) frequency response test with 500 watts of heater amplitude and 10 seconds per bit [De Wet and Per F Peterson, 2020] in the PRBS. The test procedure was such that the mass flow rate was constant at 0.18 kg/s, the heater steady state power was 8 kW, and that the outlet temperature of the CTAH, essentially the cooling fan shown in Figure 2.4, was 80 $°C$. At these conditions, the steady state inlet temperature of CIET Heaver v2.0 bare was 79.12 $°C$ [De Wet and Per F Peterson, 2020]. The heater was then perturbed with a PRBS sequence where the power oscillated between 7.5 kW and 8.5 kW according to a set 63 bit PRBS sequence at 10 seconds per bit while keeping pump and fan speeds constant [De Wet and Per F Peterson, 2020].

Due to the way the frequency response test was conducted, Equation 2.29 captures the transient response of the whole of CIET. This means that thermal pulses travelling from the heater to the CTAH would travel from the CTAH back to the heater again. The path of

the fluid in CIET is best understood with reference to Figure 2.4. I produce Figure 2.4 here again for the reader's convenience:



Flow exits the heater into the heater top head (1a) in Figure 2.4. It moves upwards

to component 5b, enters the Coiled Tube Air Heater (CTAH) before exiting and moving down into the pump. The fluid then exits the pump and traverses to component 17a before entering component 18 and re-entering the heater. For forced circulation experiments used to obtain Equation 2.29, the DHX branch, which is component 17b to component 5a in Figure 2.4, as well as the natural circulation loop, component 30a to 39, are both valved off. They do not participate in heat transfer and there is no flow going through them. They are to be ignored in the context of Equation 2.29.

Let us consider how a thermal pulse forms in this context. Suppose the heater was operating at a steady state of 8 kW. Fluid enters the heater from the bottom head, component 1b, in Figure 2.4 at 79.12 $°C$. It gets heated by the heater at operating at 8 kW and normally exits at 102.2$°C$. This reading is taken by the thermocouple "BT-12". The BT likely stands for bulk temperature. Thermocouple BT-12 is placed between component 2 and 3 in Figure 2.4. This is because the static mixer MX-10 is meant to thoroughly mix the fluid so that BT-12 can get an accurate reading of the bulk average temperature. From components 1a to 2 in Figure 2.4, the heat loss can be assumed to be small due to the fibreglass insulation. In reality, the temperature directly at the heater exit prior to entering the heater top head, 1b, may be higher than BT-12. However, we cannot know for sure as there were no thermocouples measuring bulk temperature in that region. Regardless, let us continue the discussion on thermal pulses. Now, Therminol VP-1 traverses the to reach the CTAH as explained earlier. It loses heat to approximately 80$°C$ and re-enters the heater after losing more heat in the rest of the loop before re-entering the heater at 79.12 $°C$.

Suppose now that we introduce a step input to increase power of the heater from 8 kW to 8.5 kW. After some time, the same Therminol VP-1 now exits the heater hotter than 102.2$°C$. Let's say for the sake of explaining, somewhere in the region of 105 $°C$ for example. This fluid now enters the CTAH and leaves somewhat hotter than 80 $°C$ because the fan speeds are constant. Let's say 82 $°C$ for example. After traversing the loop, this hotter fluid now enters the heater such that the heater inlet temperature is hotter than 79.12 $°C$. We now say that this thermal pulse has traversed the loop and re-entered the heater. This happens because the CTAH's fan speed was kept constant in De Wet's frequency response experiments which were used to derive the transfer function [De Wet and Per F Peterson, 2020].

From previous frequency response experiments, the time the time it takes for a thermal pulse to traverse the loop is at least 58 seconds [De Wet and Per F Peterson, 2020]. Therefore, should I subject Equation 2.29 to a step change, this thermal pulse should not appear until after about 1 minute. After 1 minute, I should expect the thermal pulse behaviour to start impacting heater outlet temperatures.

In other transient experiments, the CTAH fan speed is varied such that the CTAH outlet temperature is kept constant such as those earlier experiments used to validate models in RELAP5 [Nicolas Zweibaum, 2015] and SAM [Zou, R. Hu, and Charpentier, 2019]. These would have eliminated any chance of the thermal pulses traversing the loop and entering back into the heater. In those experiments, transient response data is available for CIET Heater v1.0's outlet temperature. While transient response data for CIET Heater v1.0 was

available, it was published in a graphical form and had to be read off a graph manually. It was more convenient to use a transfer function, and therefore I used Equation 2.29. Given this behaviour, we would expect a short term transient response and a longer term response especially when the thermal pulses arrive back from the CTAH to the heater. The thermal pulses could then go through the loop several times before the loop arrives at some new steady state. Replicating the transient response Equation 2.29 with a full forced circulation loop model of CIET is outside the scope of this dissertation. Nevertheless, the short term response of Equation 2.29 is still quite useful for validating the transient response of the heater model. This is because the time scales of the conjugate heat transfer (CHT) of the heater are much shorter than the time scales for thermal pulses traversing the loop.

To validate my model using the empirical transfer function, I can subject my heater model to the same boundary conditions and initial conditions. The boundary conditions are 79.12 $^{\circ}C$ inlet temperature, a heat transfer coefficient from steel heater surface to air of 20 $W/(m \cdot K)$ where ambient air temperature is 21.76 $^{\circ}C$ [De Wet and Per F Peterson, 2020]. The thermal resistance diagram from fluid to air can be visualised using Figure 2.22:

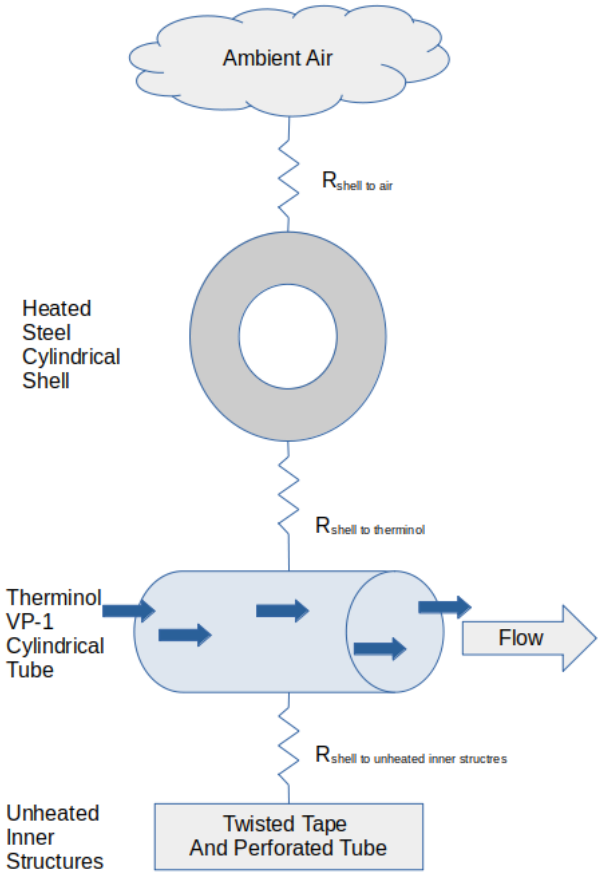


Figure 2.22: Heater v2.0 Bare Simplified Thermal Resistance Diagram

I consider all other surfaces to be adiabatic as I neglect axial heat flux in comparison to convection and radial heat flux. This was mentioned when discussing Pe. My initial conditions for the heater are at $79.12°C$. However, I am to bring the heater power to 8 kW and wait for the system to reach to steady state before starting the transient as done in De Wet's experiments [De Wet and Per F Peterson, 2020]. Therefore, the initial conditions matter less as compared to bringing the heater to this steady state.

Next, I subject my both my heater and the transfer function to the same step input. While I could use frequency response methods, a step input test is shorter and more intuitive. Hence, I chose this method. For step input tests, I need to consider the amplitude of the step input. To do so, we need to consider the conditions from which the transfer functions were obtained. De Wet has performed these frequency response tests with step inputs of 500 watts, at 10 seconds per bit using a 63 bits [De Wet and Per F Peterson, 2020]. This

PRBS signal can be thought of as a series of step inputs causing the heater power to oscillate between 7.5 kW to 8.5 kW. Thus, a step input of ± 500 watts can be used. That means that the heater, at a steady state of 8 kW, was brought up to 8.5 kW at the transient start time for one test. For the other test, it will be brought from 8 kW to 7.5 kW. I chose ± 500 watts specifically because the empirical transfer function was tested based on heater powers ranging between 7.5 kW to 8.5 kW. In doing so, I validate my model using the full range for which the empirical transfer function applies. Taking inspiration from the original PRBS tests, I can then subject my heater model to a step input of ± 500 watts, subject the transfer function to the same step input, and compare the bulk outlet temperatures to confirm if they match within thermocouple measurement uncertainty. If they do, I consider the model sufficiently validated for the purposes of this dissertation.

Of course, we could use step inputs of other magnitudes, let's say of 1 kW, for transient step response validation purposes. However, in De Wet's work with CIET, he mentioned that using larger amplitudes for perturbations tended to cause the Heater to show non-linear behaviour [De Wet and Per F Peterson, 2020]. In using larger step inputs for heater power, the transient response of CIET Heater v2.0 may have to be described by a different transfer function due to the non-linearities. To avoid this issue, I stuck to using a 500 watt step input to perform the step response tests.

The caveat to this step response method is that I can only use the short term transient data from the transfer function. This is because my heater model is not meant to replicate the thermal pulse that traverse the loop. Therefore, I cannot rely on data from the empirical transfer function past roughly one minute from the start time of the transient. Nevertheless, I still need a way of validating the heater behaviour after a long time. After the transient progresses for a long time, the heater outlet temperature will reach some new steady state. While we cannot fully validate the time scales for which the heater reaches the new steady state, we can at least verify if the heater reaches the correct steady state. For this purpose, we can validate the CIET Heater v2.0 bare model against steady state data. For the purposes of this dissertation, and for showcasing the design process of the simulated neutronics feedback controller, this will suffice. Further model improvements and validation efforts the heater and CIET are left to future work.

Steady State Data As mentioned, I also wanted to validate my Heater v2.0 model against steady state data. This data is presented in Table 2.5:

Heater Power (Watts)	Heater Inlet Temperature ($^{\circ}C$) BT-11	Heater Outlet Temperature ($^{\circ}C$) BT-12
3000	78.75	86.93
4000	79	90.25
6000	79.4	96.5
8000	79.12	102.2
10000	78.9	107.75

Table 2.5: Heater v2.0 Bare Steady State Data [De Wet and Per F Peterson, 2020], Prevailing mass flow rate $\dot{m} = 0.18$ kg/s

Table 2.5 contains heater inlet and outlet temperatures at steady state with measurement uncertainty of $\pm$ 0.5 K. The $\pm$ 0.5 K Type T thermocouple uncertainty is used here because Type T thermocouples were used within CIET [Zweibaum, Guo, et al., 2016]. The heater inlet temperature and heater outlet temperature are measured using the thermocouples in CIET BT-11 and BT-12. (BT should stand for bulk temperatures.) With reference to Figure 2.4, BT-11 lies in between component 18 and component 1b, whereas BT-12 lies between component 2 and component 3. Where component 1a is the heater bottom head and component 1b is the heater top head. BT-12, the thermocouple measuring the heater outlet temperature, is placed after a static mixer MX-10 because MX-10 ensures that the fluid is mixed thoroughly. This static mixer is represented by component 2 and component 2a on Figure 2.4. Placing the thermocouple after MX-10 ensures that the fluid temperature is more or less uniform and the local temperature measured by BT-12 is representative of the bulk fluid temperature.

At steady state, the temperature at BT-12 should be essentially the same as the bulk temperature of the fluid exiting the heater if we neglect parasitic heat loss through the heater top head and MX-10. However, when it comes to transient calculations, the fluid residence time of the fluid flowing through the top and bottom head, as well as the MX-10 need to be accounted for. So we will need to model them as well in order to compare the model to experimental data.

To consider the validation successful, the simulated bulk heater outlet temperature should match that of the experimental data in Table 2.5 to within thermocouple measurement error.

De Wet's experiments also yielded steady heater surface temperature profiles given the conditions in Table 2.5. However, if I wanted to reproduce these temperature profiles, I would likely have to model the power distribution within the resistance heater, and take into account the resistivity of steel mentioned when discussing CIET Heater v1.0:

$$\rho_{\mu\Omega-cm}(T) = 0.0612 \cdot T(^{\circ}C) + 73.109$$

$$\frac{P_{watts\ \Delta x}}{P_{watts\ total}} = \frac{\int_{x_1}^{x_2} \rho_{\mu\Omega-cm}(T)dx}{\int_0^{L_{heater}} \rho_{\mu\Omega-cm}(T)dx}$$

For the purposes of designing a simulated neutronics feedback controller, I am more concerned with the heater outlet temperature, the temperature BT-12, rather than the heater surface temperature profile. Therefore, I will not attempt to reproduce the heater surface temperature profile. Reproducing the heater surface temperature profile can be relegated to future work.

With validation details now discussed, let us move on to reviewing important parameters for model construction.

Heater v2.0 Heated Section Nodalisation First, we consider nodalisation. For CIET Heater v2.0 Bare nodalisation, there are several schemes available. In Transform, the heater was split into a heated section, and unheated inlet section and unheated outlet section [De Wet and Per F Peterson, 2020]. The heated steel cylindrical shell in Figure 2.22 was split into 8, presumably even, axial nodes along the heated section and two radial nodes [De Wet and Per F Peterson, 2020]. This is much coarser than SAM's heater nodalisation which split heated section into 15 axial nodes [Zou, R. Hu, and Charpentier, 2019], while the heater top and bottom heads are represented using two nodes each. Nevertheless, from sensitivity analysis in Zweibaum's work, 8 nodes performed as well as 15 nodes for steady state natural circulation [Nicolas Zweibaum, 2015]. Refining the mesh up to 55 nodes also provided little benefit [Nicolas Zweibaum, 2015]. Therefore, using 8 nodes instead of 15 nodes should not incur much of an accuracy penalty at least for steady state heat transfer for reproducing outlet temperatures BT-12. Moreover, the 8 node model was also validated with experimental data [De Wet and Per F Peterson, 2020]. Therefore, using 8 nodes should also be acceptable. For heater nodalisation, especially for the heated section, I aim to choose the simplest nodalisation schemes possible out of all the models because they take are easier to program and test, and also because they take less computational power.

For CIET Heater v2.0 model construction in this dissertation, I will utilise 8 evenly spaced nodes in the axial direction for the fluid nodes and solid nodes. That means that for a heated length L_{heated}, the length of each control volume will be $\frac{L_{heated}}{8}$. For steel shell nodes the radial direction, I would ideally use two nodes as well, evenly spaced in the r direction. For n radial nodes, each radial node will have a uniform thickness of $\frac{OD-ID}{2n}$ where OD is the heated shell outer diameter, and ID is the heated shell inner diameter. This was used due to ease of implementation. However, as we shall explore in the following chapter, the real-time computation requirements constrained me to using one radial node for the SS 304L heated pipe. We shall revisit nodalisation when we discuss this matter. For the unheated inner structures shown in Figure 2.22, I will also use 8 axial nodes. However, the heat transfer characteristics of inner unheated metallic structures are not well documented. The Nusselt number for the convective thermal resistance from the Therminol VP-1 to the inner metallic

structures is unavailable in literature to my best knowledge. Therefore, for the unheated SS 304L internal structures in the radial direction, I will just use one node.

Boundary Conditions for Heater v2.0 Tests For the Transform model of the heated section, suitable boundary conditions can be taken from Table 2.5. For example, at 8 kW, the boundary conditions are 79.12 $^{\circ}C$ inlet temperature. For parasitic heat losses to the environment, a heat transfer coefficient from steel heater surface to air of 20 $W/(m^2 \cdot K)$ where ambient air temperature is 21.76 $^{\circ}C$ [De Wet and Per F Peterson, 2020]. These are the same for the heater operating at any power. As mentioned before, the thermal resistance diagram from fluid to air can be visualised using Figure 2.22. Every other surface is considered adiabatic with regards to conduction heat transfer. However, fluid will enter the inlet at $\dot{m}$ of 0.18 kg/s and will exit the heater outlet at the same mass flow rate. Fluid shall exit the outlet at the same temperature as the last fluid control volume. For eight nodes, control volume 8 shall be at the exit. The fluid shall leave the system from control volume 8 at the temperature of control volume 8 and carry away the enthalpy based on the temperature of control volume 8. For the digital twin of CIET's Heater constructed in this dissertation, the boundary conditions and initial conditions will be revisited and discussed in the next chapter with the use of diagrams.

To construct the thermal resistances and to determine the thermal inertia of these control volumes, let us now consider some important heat transfer properties relevant to modelling CIET Heater v2.0 Bare.

Nusselt Number and Thermal Inertia for CIET Heater v2.0 Now, Heater v2.0 can have its solid-fluid heat transfer modelled with the empirical equation 2.30 [De wet, Per F. Peterson, and Greenwood, 2019; Lukas, Kendrick, and P. Peterson, 2017]:

$$\mathrm{Nu} = 0.0391 \ \mathrm{Re}_{D-heated}^{0.812} \mathrm{Pr}^{0.408} \left(\frac{\mu_{bulk}}{\mu_{wall}} \right)^{0.14} \tag{2.30}$$

In Equation 2.30, the last coefficient $\left(\frac{\mu_{bulk}}{\mu_{wall}}\right)^{0.14}$ represents a viscosity correction factor meant to account for temperature differences between wall and fluid. This practice of adding this viscosity correlation factor is used in other Nusselt number correlations such as Hausen's correlation [Gnielinski, 2013]. Sometimes, however, the correction factor would be $\left(\frac{\mathrm{Pr}}{\mathrm{Pr}_{surf}}\right)^{0.11}$ as it is based on Petuhov's correlation [Gnielinski, 2013] rather than the using $\left(\frac{\mu_{bulk}}{\mu_{wall}}\right)^{0.14}$. Now, in later publications of regarding the Transform model, there was no documented use of the wall viscosity or wall Pr correction factor for the heater Nusselt number [De Wet and Per F Peterson, 2020]. Thus, there is some discrepancy in literature as to whether the correction factor was used in the Transform model. One can verify that based on Therminol VP-1 properties, Pr is about 16 at the low bound fluid bulk temperature of

$80°C$ and about 9 at a typical heater surface temperature of $180°C$ [De Wet and Per F Peterson, 2020]. $\left(\frac{\mathrm{Pr}}{\mathrm{Pr}_{surf}}\right)^{0.11}$ is about 1.07 while $\left(\frac{\mu_{bulk}}{\mu_{wall}}\right)^{0.14}$ is about 1.13 at these temperatures. Thus, these correction factors account for about an additional 10% of Nu.

While the Nu correction factors are important to consider, the experimentally determined heater surface temperature was about $185°C$ at 10 kW and 0.18 kg/s [De Wet and Per F Peterson, 2020]. This is already outside the temperature range of applicability for the Therminol VP-1 correlations (20-180 $°C$) [Nicolas Zweibaum, 2015]. At lower mass flow rates, and at 10 kW heater power, the heater surface temperatures are bound to be even higher. Thus, I would have had to find separate correlations in order to predict μ and Pr at these higher wall temperatures. For the sake of expediency, I did not do so for my current iterations of the heater model presented in this work. I found out later that excluding this correction factor did not appreciably cause the simulated bulk heater outlet temperatures to deviate from experimental data. Therefore, I decided to ignore the wall correction factors for the time being and relegate its implementation to future work. Again, we can ascertain if these simplifications made significantly impact the accuracy of the model in the model validation section in the next chapter.

Now, let's go back to Equation 2.30. The Nusselt Number in Equation 2.30 was correlated using averaged temperatures rather than a temperature profile. The original definition was written as [Lukas, Kendrick, and P. Peterson, 2017]:

$$\mathrm{Nu}_{D-heated} = \frac{\dot{m} c_p (T_{out} - T_{in})}{A_{heater}\left(\tilde{T}_w - \frac{T_{out}-T_{in}}{2}\right)} \frac{D_{heated}}{k} \tag{2.31}$$

For Equation 2.31, $\dot{m}$ is mass flow rate in units similar to kg/s, T_{out} and T_{in} are the measured bulk outlet and inlet temperatures respectively, c_p is the fluid specific heat capacity in units similar to $J/(kg \cdot K)$. D_{heated} is heated diameter and k is thermal conductivity in units similar to $W/(m \cdot K)$. $\tilde{T}_W$ is mean wall temperature measured at half the length of the heater [Lukas, Kendrick, and P. Peterson, 2017]. Equation 2.30 was developed using heated diameter of 0.0381 m as the basis for the Reynold's number $\mathrm{Re}_{D-heated}$ [Lukas, Kendrick, and P. Peterson, 2017]. In case there is any confusion, one may refer to Figure 2.19 to make sense of the relevant dimensions. This heated diameter is the inner diameter of the outer tube, which is heated electrically. However, in subsequent models of CIET, Equation 2.30 was converted to use the hydraulic diameter ($D_{hydraulic}$) instead, using $D_{hydraulic} = 0.01467m$ as it was considered best practice to use hydraulic diameter for annular geometries [De wet, Per F. Peterson, and Greenwood, 2019]. There was little published justification as to why the heated diameter was originally used instead of hydraulic diameter. Nevertheless, the correlation still proved useful as an empirical correlation to predict friction factor using Nu. The hydraulic diameter is $4A_{XS}/P_w$ where A_{XS} is the cross sectional area, and P_w is the wetted perimeter. Therefore, correlations based on hydraulic diameter were used in later models of CIET Heater v2.0 such as those built in Transform [De wet, Per F. Peterson, and Greenwood, 2019]:

$$\mathrm{Nu}_{D-hydraulic} = 0.04179\ \mathrm{Re}^{0.836}_{D-hydraulic}\mathrm{Pr}^{1/3}\left(\frac{\mu_{bulk}}{\mu_{wall}}\right)^{0.14} \tag{2.32}$$

For modelling CIET's Heater, we shall stick to this practice and use a hydraulic diameter of 0.01467 m [De wet, Per F. Peterson, and Greenwood, 2019] to calculate $\mathrm{Re}_{D_{hydraulic}}$ and substitute this into Equation 2.32. To make sense of these two diameters, one can first observe that hydraulic diameter is smaller than the heated diameter of 3.81 cm. Figure 2.19 shows that this 3.81 cm is also the hydraulic diameter of the heater if the heater was completely filled with fluid as in a pipe. Since we fill the this cylindrical pipe with unheated structures, A_{XS} of the fluid decreases while P_w increases. Therefore, $4A_{XS}/P_w$ decreases relative to 3.81 cm and $D_{hydraulic} < 3.81\ cm$.

For most of these correlations, fluid properties are calculated at the mean fluid temperature [Lukas, Kendrick, and P. Peterson, 2017]. However, the exact definition of mean fluid temperature was not explicitly stated in the original conference paper [Lukas, Kendrick, and P. Peterson, 2017]. This could mean the average bulk fluid temperature, which is $\frac{T_{out}-T_{in}}{2}$, or the mean film temperature, which is the average of the bulk temperature and surface temperature. While it is common practice to evaluate fluid properties at film temperature [Bejan, 2013], the exact definitions of mean temperature for this Nusselt correlation were difficult to find.

When applying this to control volumes, I noted that the fluid temperature within each control volume is uniform. Knowing this, I then used the temperature of the fluid control volume to calculate a local Nu and convective thermal resistance. When using eight control volumes to represent the heated tube, I would then have eight values of Nu with which to calculate a localised convective thermal resistance. This was the simplest approach, and the simulations of heater outlet temperature using this Nu were then validated with existing experimental data. We shall revisit this in the next chapter.

Now, Equation 2.30 and Equation 2.32 applies for mass flow rates through the CIET Heater v2.0 from 0.1 kg/s to 0.24 kg/s at heater power values of 0 to 6 kW [Lukas, Kendrick, and P. Peterson, 2017]. This corresponded to $350 < \mathrm{Re}_{D-heated} < 3978$ [Lukas, Kendrick, and P. Peterson, 2017]. However, we want to use $\mathrm{Re}_{D-hydraulic}$ rather than $\mathrm{Re}_{D-heated}$. For this, let us now perform conversion between $\mathrm{Re}_{D-heated}$ and $\mathrm{Re}_{D-hydraulic}$:

$$\mathrm{Re}_{D-heated} = k\mathrm{Re}_{D-hydraulic}$$

Where k is some constant to be determined using the dimensions of CIET Heater v2.0 Now, for both Re:

$$\mathrm{Re}_{D-heated} = \frac{\rho u D_{heated}}{\mu} = \frac{\dot{m} D_{heated}}{A_{XS}\mu}$$

$$\mathrm{Re}_{D-hydraulic} = \frac{\rho u D_{hydraulic}}{\mu} = \frac{\dot{m} D_{hydraulic}}{A_{XS}\mu}$$

Where Lukas defined D_{heated} as 3.81 cm, and u is the average velocity based on the mass flow rate $\dot{m}$ and flow area A_{XS} [Lukas, Kendrick, and P. Peterson, 2017]. Some of these relevant dimensions are provided in Table 2.6 [De Wet and Per F Peterson, 2020]:

Name	Value
Outer Tube Outer Diameter	4.0 cm
Outer Tube Inner Diameter	3.81 cm
Outer Tube Heated Length	167.6 cm
Outer Tube Heat Transfer Area	2007 cm^2
Inner Tube Outer Diameter	3.175 cm
Inner Tube Inner Diameter	2.66 cm
Inner Tube with Tape Heat Transfer Area	4639 cm^2
Flow Area	10.52 cm^2
Fluid Volume Height	198 cm
Metal Material	304L Stainless Steel
Heat Transfer Fluid	Dowtherm-A or Therminol VP-1
Twisted Tape Diameter	2.54 cm
Twisted Tape Thickness	0.122 cm
Twisted Tape Height in Fluid	198 cm
Assembly Mass	3.12 kg
Main Fluid Volume	2083.5 cm^3
Hydrualic Diameter	1.467 cm

Table 2.6: Heater 2.0 Design and Performance Parameters [De Wet and Per F Peterson, 2020]

From Table 2.6, D_{heated} is also the inner diameter of the outer pipe, which is what heats the fluid. A_{XS} can also be found in Table 2.6 as 10.52 cm^2. We can now substitute the expressions for Re based on mass flow rate for both Reynold's numbers to obtain:

$$\frac{\dot{m}D_{heated}}{A_{XS}\mu} = k\frac{\dot{m}D_{hydraulic}}{A_{XS}\mu}$$

Since $\dot{m}$, A_{XS} and μ are the same for the same pipe:

$$D_{heated} = kD_{hydraulic}$$

Using the hydraulic diameter from Table 2.6, we get us to $k = 2.5971$ (5 significant figures are used for intermediate calculation).

$$\mathrm{Re}_{D-heated} = 2.5971\mathrm{Re}_{D-hydraulic}$$

Let us consider changing the Re range from heated to hydraulic diameter:

$$350 < \mathrm{Re}_{D-heated} < 3978$$

Divide throughout by 2.5971:

$$\frac{350}{2.5971} < \frac{\mathrm{Re}_{D-heated}}{2.5971} < \frac{3978}{2.5971}$$

$$135 < \mathrm{Re}_{D-hydraulic} < 1531$$

While $\mathrm{Re}_{D-hydraulic} < 2300$, this should not necessarily imply a laminar regime as the unheated twisted tapes and perforated tubes are meant to improve mixing and heat transfer. In contrast, laminar flow occurs when there are layers of fluid sliding past each other with little or no mixing between the layers. Another piece of information which may suggest a non-laminar flow is the form of the empirical friction factor correlation developed for CIET Heater v2.0. We can compare this friction factor to the laminar friction factor in pipe flows [Perry and Green, 2015]:

$$f_{darcy} = \frac{64}{Re_{D-hydraulic}}$$

In contrast, the friction factor originally developed for heater v2.0 is [De Wet and Per F Peterson, 2020; Lukas, Kendrick, and P. Peterson, 2017]:

$$f = 17.9 * \mathrm{Re}_{D-heated}^{-0.34} \tag{2.33}$$

Again, Equation 2.33 uses a heated perimeter of 0.0381 m rather than the hydraulic diameter of 0.01467 m in its original formulation [Lukas, Kendrick, and P. Peterson, 2017] when published in a conference paper. Similar to before, let us convert Equation 2.33 So that it uses hydraulic diameter. When we do so, we obtain Equation 2.34:

$$\begin{aligned} f &= 17.9 * \mathrm{Re}_{D-heated}^{-0.34} * D_{hydraulic}^{0.34} D_{hydraulic}^{-0.34} \\ &= 17.9 * \mathrm{Re}_{D-hydraulic}^{-0.34} * D_{hydraulic}^{0.34} D_{heated}^{-0.34} \\ &= 17.9 * \mathrm{Re}_{D-hydraulic}^{-0.34} * (0.01467)^{0.34} * 0.0381^{-0.34} \\ &= 12.94 * \mathrm{Re}_{D-hydraulic}^{-0.34} \end{aligned} \tag{2.34}$$

Regardless of the type of diameter used for Re, we note that that f scales as $\mathrm{Re}^{-0.34}$ rather than Re^{-1} in laminar pipe flow friction factors which suggests where pressure drop scales linearly with Re [Perry and Green, 2015]. Even in complex geometries such as packed beds, Ergun's equation [Ergun and Orning, 1949] suggests that in the laminar regime (or creeping flow regime), the friction factor scales as Re^{-1} as it does for Darcy's law for porous media [Perry and Green, 2015]. We can infer from these observations that flow in the heated tube is quite unlikely to be laminar in these given Re values.

For lower mass flow rates and Re, I was unable to find any other correlation. Thus, the applicability of this correlation in those flow regimes is questionable. Nevertheless, in this work, the digital twin of the Heater is meant for development of a simulated neutronics feedback controller operating in CIET at a single fixed flow rate of 0.18 kg/s. Developing a simulated neutronics feedback controller at other flow rates requires more work with reactor physics that is beyond the scope of this dissertation. Therefore, I simply did not solve for pressure drop across the heated section. Implementation and validation of more fully featured thermal hydraulics model of CIET Heater v2.0 Bare is left to future work.

For thermal inertia of the heater, calculations can be performed using the following design and performance parameters also in table 2.6. The heater outer tube mass for the heated section can be calculated as:

$$m_{tube} = \rho_{steel} V_{heater} = \rho_{steel} \frac{\pi L}{4} \left(OD^2 - ID^2 \right)$$

If we use table 2.6, L_{heated} is 167.6 cm in ID is the 3.81 cm, OD is 4.0 cm. However, for this dissertation, I opted to use L_{heated} of 163.83 cm as mentioned earlier in the chapter. Now, ρ_{steel} and $c_{p,steel}$, the density and heat capacity for steel, can be interpolated using tabulated data for steel [Zou, R. Hu, and Charpentier, 2019] or using correlations previously discussed [Graves et al., 1991].

Besides the heated tube, CIET Heater v2.0 Bare also contains unheated heat structures as mentioned before. These are the twisted tape and perforated tube. Unfortunately, there is insufficient information about the these heat structures available in published literature. In fact, the perforated tube was not modelled in using codes such as SAM [Zou, R. Hu, and Charpentier, 2019]. To simplify and expedite the model development process, I also ignore the thermal inertia of the perforated tube. Improvements to the heater model fidelity by modelling this tube is relegated to future work.

However, the twisted tape is modelled in the SAM model [Zou, R. Hu, and Charpentier, 2019]. I wish to model the twisted tape as well, but again, the heat transfer information for the twisted tape is limited. Hence, both its thermal inertia and the thermal conductance between it and the fluid have to be guessed.

From Table 2.6, the twisted tape width within the inner tube is 2.54 cm (1 inch), 0.122 cm (0.048 inch) thickness and length equal to the length of the heated section. To model the thermal inertia, I estimated the volume of the twisted tape assuming the twisted tape was a straight, rectangular strip with the same dimensions:

$$V_{twisted\ tape} = w_{twisted\ tape} \times t_{twisted\ tape} \times L_{heated\ tube}$$

Where $w_{twisted\ tape}$ is the twisted tape width and $t_{twisted\ tape}$ is the twisted tape thickness. As for the heat transfer correlation between the twisted tape and the fluid, I did not have any prior model with which to refer to. Of the correlations I could have used, Wakao correlation [Wakao, Kaguei, and Funazkri, 1979] meant for porous media, seemed to be quite suitable since one could model the twisted tape and perforated tube as porous media for the fluid to flow through:

$$\mathrm{Nu}_{d_p} = 2 + 1.1\ \mathrm{Pr}^{1/3}\mathrm{Re}_{d_p}^{0.6}$$

Where d_p is the particle diameter. Wakao's correlation uses Re and Nu based on particle diameter. Hence, I would have had to calculate fluid and solid fractions and find an equivalent particle diameter for the Wakao correlation. I left this endeavour for future work. For I just used what was most convenient: the Gnielinski correlation [Gnielinski, 2013]. We shall revisit Gnielinski's correlation later in this chapter when we discuss models used for CIET's piping. This was because I had the hydraulic diameter on hand with which to calculate Re. Moreover, I could not find the heat transfer area of the twisted tape separate from the perforated tubing. One approach could be to calculate the twisted tape area assuming it was a rectangular strip as before. However, I decided against putting in effort to model the twisted tape for this current iteration. I just used the inner tube twisted tape plus inner tube heat transfer area in Table 2.6 as a placeholder value for the time being. The heat transfer area I used for the heated section was:

$$A_{twisted\ tape} = \frac{L_{heated\ tube}}{198cm} * 4639\ \mathrm{cm}^2$$

Needless to say, this expression will overestimate the thermal conductance between the Therminol VP-1 and the twisted tape. Nevertheless, for the purposes of this dissertation, I only needed an approximate model of the CIET Heater v2.0 with which to test my simulated neutronics feedback controller. As long as the model can be validated in the ways mentioned earlier, the model will suffice.

Heater Top and Bottom Heads

As for the heater top and bottom heads, the solid structures are usually modelled using two axial nodes and three radial nodes [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015]. These are presumably spaced evenly. Using this assumption, I likewise spaced the nodes evenly as discussed for the heated sections of CIET Heater v2.0 Bare. Likewise, two axial nodes and one radial node is used for the Therminol VP-1 within these structures. For CIET Heater v2.0, the perforated tubing is ignored [Zou, R. Hu, and Charpentier, 2019] in these top and bottom heads.

For the twisted tape specifically, I merely scale the volume of the twisted tape by its length in the heater top and bottom heads.

$$V_{twisted\ tape} = w_{twisted\ tape} \times t_{twisted\ tape} \times L_{heater\ bottom\ or\ top\ head}$$

Where $w_{twisted\ tape}$ is the twisted tape width and $t_{twisted\ tape}$ is the twisted tape thickness. Again, from Table 2.6, the inner tube with tape width is 2.54 cm (1 inch), 0.122 cm (0.048 inch) thickness and length equal to the length. The length of the heater top or bottom head $L_{heater\ bottom\ or\ top\ head}$ will be used to calculate the volumes in each. The tapes are made of SS 304L. Hence, those properties will be used to calculate its mass and thermal inertia. For thermal conductance, an approach similar to the heater was used where:

$$A_{twisted\ tape} = \frac{L_{heater\ bottom\ or\ top\ head}}{198cm} * 4639\ \text{cm}^2$$

And the heat transfer coefficient for the twisted tape was based again on Gnielinski's correlation [Gnielinski, 2013] as a placeholder since no explicit validation data was available. We shall explore Gnielinski's correlations more when reviewing literature for modelling piping within CIET later in this chapter.

As for the dimensions of the heater top and bottom heads, there is disparity even within models in literature. This is because work was done in later iterations of the heater top and bottom head models order to improve the modelling of thermal inertia within these components [De Wet and Per F Peterson, 2020]. In the earlier RELAP models [Nicolas Zweibaum, 2015; Zou, R. Hu, and Charpentier, 2019] the relevant parameters for the heater top and bottom head are presented in Table 2.7:

Property	Heated Section	Heater Bottom Head (Inlet)	Heater Top Head (Outlet)
Length	1.6383 m	0.19685 m	0.0889 m
Hydraulic Diameter	$6.60 * 10^{-3}m$	$6.60 * 10^{-3}m$	$6.60 * 10^{-3}m$
Flow Area	$3.64 * 10^{-4}m^2$	$3.64 * 10^{-4}m^2$	$3.64 * 10^{-4}m^2$
Wall Thickness	0.001905 m	0.001905 m	0.001905 m

Table 2.7: Heater v2.0 Design and Performance Parameters used in the RELAP Model [Nicolas Zweibaum, 2015]

In contrast, the later models in Transform model improves upon the RELAP model by modelling the heater top head (outlet) and bottom head (inlet) with additional fluid and steel masses [De Wet and Per F Peterson, 2020]. This was meant to help the CIET Transform model better match the experimental frequency response data. The heater top head, bottom head, and heated section parameters used in Transform for all version of the heater can be found in Table 2.8:

Property	Heated Section	Unheated Inlet	Unheated Outlet
Length	1.676 m	0.46783 m	0.35458 m
Hydraulic Diameter	0.01467 m	0.01467 m	0.01467 m
Flow Area	0.00105 m^2	0.00105 m^2	0.00105 m^2
Wall Thickness	0.001905 m	0.0083058 m	0.0083058 m

Table 2.8: Heater v2.0 Design and Performance Parameters used in the Transform Model [De Wet and Per F Peterson, 2020]

Table 2.8 utilised information from De Wet's work on Transform [De Wet and Per F Peterson, 2020] for the heated length, hydraulic diameter, flow area and wall thickness. These were modelled as "fluid volumes", which I had to assume were cylindrical pipes due to the lack of published information. This included the lack of a published nodalisation table similar to the ones found for SAM and RELAP. Therefore, I was uncertain if the "Inlet" and "Outlet" referred to in De Wet's data referred to the heater bottom and top head respectively, or if the outlet included MX-10 as well. I could of course run several tests to check if the "Inlet" and "Outlet" meant the top and bottom head. For the time being, I will assume that the "Inlet" and "Outlet" refer to the top and bottom heads respectively, each with their own coupled thermal masses. Additionally, the Nusselt correlation was not explicitly stated for the top and bottom heads. The notes for the Nusselt correlation meant that it used a two region, single phase Nusselt correlation from Transform [De Wet and Per F Peterson, 2020]. However, I had no access to Transform, and had to guess what this was. I also could not use correlations in the RELAP5 model because the inner geometries were different. For the SAM model, I also found it difficult to find the exact Nusselt correlation used for this.

Based on this, I could simply use either the Gnielinski correlation for pipes [Gnielinski, 2013] or the CIET heater Nusselt number correlation to estimate the Nusselt number in the heater top and bottom heads. For the purpose of deciding an appropriate correlation to use, I also considered the thermal resistance diagram in Figure 2.22. This diagram also happens to apply for the heater top and bottom heads as well. Based on Figure 2.22, Nu in the top and bottom heads is important in determining parasitic heat losses to the air. Another observation from Figure 2.22, is that this convective thermal resistance is in series with the thermal resistance from the heated steel tube to air. Since the heater design was meant to maximise Nu, it is reasonable to assume that the convective thermal resistance for steel to air is much greater than the convective thermal resistance for the steel tube to Therminol-VP1. Based on this assumption, the sensitivity of parasitic heat loss to deviations in Nu from Therminol VP-1 to the steel tube in the heater top and bottom heads may not be significant. Therefore, whether I use the Nu correlations for generic pipes or the heater

Nu correlation, it may not matter significantly. I eventually decided to use the Gnielinski correlation [Gnielinski, 2013] due to its simplicity. Of course, we shall need to check if the resulting model can reproduce the heater outlet temperature. We shall revisit this in the next chapter.

While I need more data to replicate the Transform model, the information provided in Table 2.8 based on De Wet's dissertation [De Wet and Per F Peterson, 2020] is still valuable and useful if one wanted to model the thermal inertia of the heater top and bottom heads. While the geometry is not explicitly provided, it may be reasonable to assume that these are shaped as cylinders where its dimensions are given in Table 2.8 and that the Inlet and Outlet refer to the bottom and top heads respectively. Under these assumptions, we can compare the relative geometries of the RELAP5 model to the Transform model. In Table 2.7, the wall thickness is modelled as 0.001905 m, whereas in Table 2.8, the wall thickness is modelled at 0.0083058 m. In other words, the Transform model had a wall thickness for the heater top and bottom head or about four times more than in the RELAP model. Moreover, the Transform model had the heater Inlet and Outlet modelled as longer pipes in comparison to the RELAP5 model. Thus, the Transform model would have accounted for more thermal inertia in the heater top and bottom heads as compared to the RELAP5 model. This is best visualised in Figure 2.23:

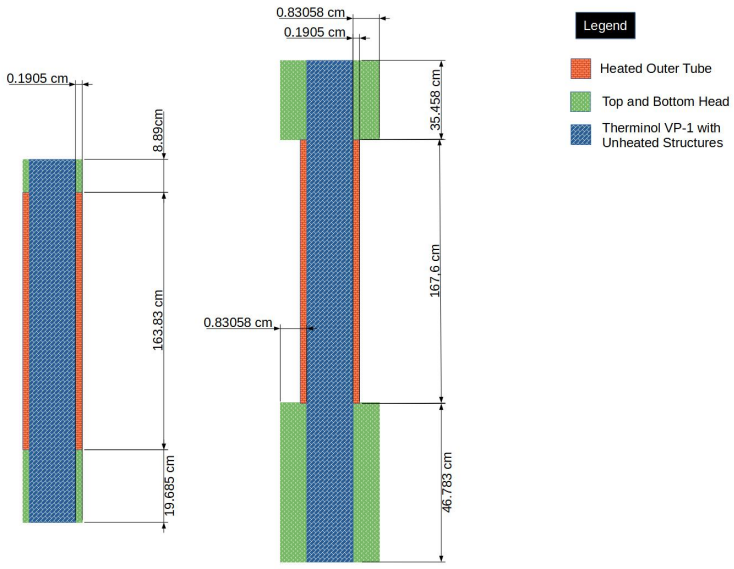


Figure 2.23: Comparison of the Modelling Approach used in RELAP5 [Nicolas Zweibaum, 2015] and Transform [De Wet and Per F Peterson, 2020]

It would be important to test the implications of this thermal inertia on the transient response of the heater. However, for the purposes of this work, I just need a reasonably working model of the heater with which to test my simulated neutronics feedback controller.

As mentioned before, I only need validate this using the transfer function and steady state data for the purposes of this dissertation. Therefore, I only validated the heater model constructed using the RELAP nodalisation for the purposes of this dissertation. As we will discuss in the next chapter, modelling of these extra couple thermal masses was not required to get the transient responses of the simulation and empirical transfer function to match within thermocouple measurement error. Modelling of the extra thermal masses is left to future work.

Proposed CIET Heater Model

For a first iteration of Heater v2.0, I would want to keep my heater model as simple as possible. Hence, I do not want to model the entrained fluid in the inlet and outlet portions. For now, I would adopt the RELAP method of modelling the heater top and bottom head without modelling fluid and steel masses at the inlet and outlet of the heater. This is done in this dissertation because it is a simpler to develop. The final CIET Heater v2.0 Bare model which I eventually validated with experimental data is shown in Figure 2.24:

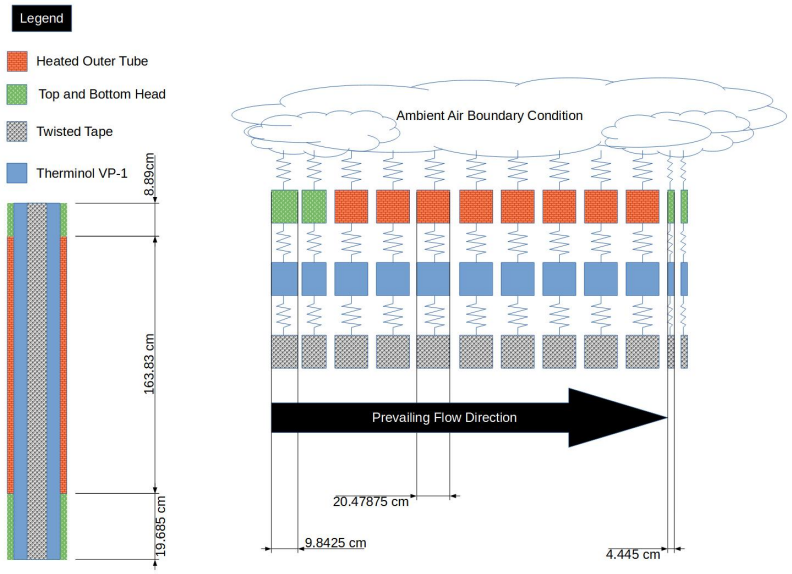


Figure 2.24: CIET Heater v2.0 Bare Proposed Model (Not to Scale)

Figure 2.24 shows the length dimensions for the CIET Heater v2.0 Bare. The internals are based of the dimensions of the twisted tape discussed earlier. The lengths will be the same as that of the heated section, the top and bottom heads respectively. The tape width is 2.54 cm (1 inch) and its thickness is 0.122 cm (0.048 inch). For the purposes of calculating its thermal resistance to the Therminol VP-1 control volumes, the placeholder value I use for the heat transfer area is a total of 4639 cm^2 spread over the heated section, top and bottom

heads scaled by the lengths relative to each other. The Nusselt number correlation is based on the generic Gnielinski correlation [Gnielinski, 2013] which we will review in the following subsection.

For the fluid volumes, the total fluid height is governed by the total lengths of the heated sections, as well as the top and bottom heads. The flow area is 10.52 cm^2, and the total volume of the fluids is the flow area multiplied by the height. For the heated outer steel tube for the heated section, the length is 163.83 cm. The inner diameter (ID) is 3.81 cm while the outer diameter (OD) is 4.0 cm. For the purposes of calculating thermal resistance, the heat transfer area to the Therminol VP-1 control volumes is calculated based on ID, and the Nusselt number correlation is based on:

$$\mathrm{Nu}_{D-hydraulic} = 0.04179\ \mathrm{Re}^{0.836}_{D_{hydraulic}}\mathrm{Pr}^{1/3}$$

For simplicity, no wall Pr or μ correction factor was used for the time being. Moreover, the axial thermal conductances between the heated section and the heater top and heater bottom head are set to zero for the purposes of model validation tests. The second component of thermal resistance from tube centre to Therminol VP-1 is based on the half thickness of the tube. Likewise, the thermal resistance from tube centre to air is governed by the half thickness of the tube, OD, and a h of 20 $W/(m^2 \cdot K)$ as used in the validated Transform model [De Wet and Per F Peterson, 2020].

Figure 2.24 shows the final nodalisation scheme I eventually used for the CIET Heater Bare v2.0. While the Transform model utilised two radial nodes for the heated steel shell, I eventually decided to use one radial node. This was to ensure calculations for the CIET Heater could keep up with real-time requirements. We will discuss the nodalisation scheme with the use of diagrams in the next chapter.

The coarser nodalisation scheme, simplified modelling of the twisted tape and heater internals, as well as the neglect of a significant portion of thermal inertia within the top and bottom heads may adversely impact accuracy of the model. Therefore, this model needs to be validated using experimental data. As mentioned previously, I decided to test this first iteration using De Wet's heater power to heater outlet temperature transfer function to ascertain how well the heater reasonably matches the transient response of the outlet temperature and if these simplifications caused significant deviation from the experimental correlation. This is discussed in the next chapter in the validation section of CIET Heater v2.0 Bare simulation. We could include more detailed models of the heater inlet and outlet, but this is reserved for future work.

CIET Piping

While we are not modelling the whole of CIET, we still need to implement models for CIET's piping. As discussed before, Thermocouple BT-12 is placed between component 2 and 3 in Figure 2.4. This is because the static mixer MX-10 is meant to thoroughly mix the fluid so that BT-12 can get an accurate reading of the bulk average temperature. Component

2 and 2a represent static mixer MX-10. Therefore, the outlet temperature measured in De Wet's experimental data is not the bulk temperature of the fluid exiting the heater top head [De Wet and Per F Peterson, 2020], but rather the fluid temperature after MX-10. In the RELAP5 model, static mixer MX-10 was modelled as an insulated pipe. Therefore, we need to review some literature relevant to pipe flow used for modelling MX-10 within CIET.

When modelling pipe flow, we restrict our discussion mostly to heat transfer and Nusselt number correlations. In my master's thesis [Ong, 2023], I already mentioned that I used the Churchill correlation [Perry and Green, 2015] to predict f_{darcy} in the laminar and turbulent regimes, and I did not explicitly model hydrodynamic entrance effects in the pipes. This model was already validated using experimental data as mentioned when I summarised my master's thesis. Hence, I will not repeat its full discussion here.

CIET Piping Models in Transform

For pipe modelling in CIET, a simple Nusselt number correlation were previously used to calculate heat transfer. In the laminar regime, the Transform model uses [De Wet and Per F Peterson, 2020] Equation 2.35:

$$\mathrm{Nu}_{D-hydrualic} = 4.36 \tag{2.35}$$

Equation 2.35 is applicable for $\mathrm{Re}_{D-hydraulic} < 2100$, and is also the same as the laminar Nusselt number derived for fully developed flow with a uniform heat flux boundary condition [Bejan, 2013]. In turbulent flow, the Transform model uses [De Wet and Per F Peterson, 2020] Equation 2.36:

$$\mathrm{Nu}_{D-hydraulic} = 0.023\mathrm{Re}^{0.8}_{D-hydraulic}\mathrm{Pr}^{0.4} \tag{2.36}$$

Equation 2.36 is essentially the Dittus Boelter correlation [Bejan, 2013] for fully developed flow where the pipe length to hydraulic diameter ratio $\frac{L}{D_{hydraulic}} \geq 60$ [Bejan, 2013]. The transform model applied a smoothing scheme for interpolation between the fully developed laminar and turbulent flow described in De Wet's dissertation [De Wet and Per F Peterson, 2020]. However, I am unclear as to what this smoothing scheme is like The smoothing scheme is:

$$\mathrm{Nu} = \mathrm{Nu}_{turbulent}\alpha + \mathrm{Nu}_{laminar}(1-\alpha)$$

Where $\alpha = (\tan\tan(x/\delta x) + 1)$. x is the input variable, and δx is half with of the transition region. Presumably, x is Re in this case as the correlations developed were generic to the Transform program. [De Wet and Per F Peterson, 2020]. The lack of brackets around the tangent term makes its meaning ambiguous, and I am not sure if it is an editorial error within the dissertation appendices. Nevertheless, I gather from this that the Nusselt number for turbulent and laminar schemes assume fully developed flow, perhaps due to its simplicity. Secondly, I also gather that that the Nusselt number in the transition regime between turbulent and laminar flow can be interpolated. Since Transform used models which

interpolate between laminar and turbulent flow regimes, I sought a similar Nusselt number correlation within literature that also interpolates between the two flow regimes.

Gnielinski's Correlations

For pipes, a suitable Nusselt number correlation is Gnielinski's correlation [Gnielinski, 2013] which provides correlations for each flow regime:

$$\mathrm{Nu} = \begin{cases} \mathrm{Nu}_{lam}(\mathrm{Re}) & \text{for Re } < 2300 \\ \mathrm{Nu}_{turb}(\mathrm{Re}) & \text{for Re} > 4000 \\ (1-\gamma)\mathrm{Nu}_{lam}(\mathrm{Re}=2300) + \gamma \mathrm{Nu}_{turb}(\mathrm{Re}=4000) & \text{for } 2300 < \mathrm{Re} < 4000 \end{cases} \tag{2.37}$$

Where:

$$\gamma = \frac{\mathrm{Re} - 2300}{4000 - 2300} \text{ for } 2300 < \mathrm{Re} < 4000 \tag{2.38}$$

These equations are usable for flow in laminar, transition and turbulent regimes. Additionally, they also account for developing and fully developed flow. To ascertain this, we can look at the provided forms of laminar and turbulent Nusselt Number (Nu). The expressions for laminar Nu, including the entrance region effects for uniform heat flux, are [Gnielinski, 2013]:

$$\mathrm{Nu}_{lam} = \left\{ 4.354^3 + 0.6^3 + \left(1.953 \sqrt[3]{\mathrm{Re\ Pr} \frac{d}{L}} - 0.6 \right)^3 + 0.924^3 \mathrm{Pr} \left(\mathrm{Re} \frac{d}{L} \right)^{1.5} \right\}^{1/3} \tag{2.39}$$

Equation 2.39 accounts for the entrance length using a d/L term, where d is the representative diameter, usually $D_{hydraulic}$ and L is the pipe length. As the flow develops, $L \to \infty$ and $d/L \to 0$ and these equations reduce to a form suitable for fully developed flow. Should the flow fully develop for Equation 2.39, we get:

$$\mathrm{Nu} = \left(4.354^3 + 0.6^3\right)^{1/3} = 4.3577 \approx 4.36$$

This approaches the Nusselt number for uniform heat flux of about 4.36 [Bejan, 2013]. Now, for laminar flow, we note that there is a difference between uniform temperature and uniform heat flux boundary conditions for the Nusselt number. For constant wall temperature, the laminar Nusselt for fully developed flow is 3.66. Gnielinski also provides a similar correlation where Nu approaches 3.66 [Gnielinski, 2013] as the flow fully develops. Though in reality, heat transfer hardly occurs at mathematically convenient boundary conditions. A more realistic boundary condition is perhaps where there is some uniform temperature on the outside of the pipe, some thermal resistance and thermal inertia within the pipe,

and perhaps a changing temperature difference axially along the pipe. Given this state of affairs, is difficult to define the Therminol VP-1 to steel boundary clearly as either uniform temperature or uniform heat flux. A simple approach for this work is to take the uniform heat flux Nu similar to previous CIET system simulations in Transform. Therefore, I just used the constant heat flux approach for my model.

Now, to evaluate Nu, we traditionally evaluate fluid properties at film temperature T_{film} [Bejan, 2013]:

$$T_{film} = \frac{T_{bulk} + T_{surf}}{2} \tag{2.40}$$

T_{bulk} is the bulk fluid temperature and T_{surf} is the temperature of the surface. Now, while it is best to use film temperature when evaluating fluid properties, it is sometimes more convenient in code to just use T_{bulk} as an approximation of T_{film}. Hence, for first iterations of code, I used T_{bulk} rather than T_{film}. For subsequent iterations of code, I would incorporate the use of T_{film} into the code.

Now that we have dealt with the laminar portions, we move on to the turbulent Nusselt number. This is presented in Equation 2.41 [Gnielinski, 2013] :

$$\mathrm{Nu}_{turb} = \frac{(f_{darcy}/8)(\mathrm{Re} - 1000)\,\mathrm{Pr}}{1 + 12.7\sqrt{(f_{darcy}/8)}(\mathrm{Pr}^{2/3} - 1)} \left[1 + \left(\frac{d}{L}\right)^{2/3}\right] \left(\frac{\mathrm{Pr}}{\mathrm{Pr}_{surf}}\right)^{0.11} \tag{2.41}$$

f_{darcy} is the Darcy friction factor which can be calculated using Filonenko's equation or Konakov's equation[Gnielinski, 2013]. However, other correlations such as Churchill's correlation could potentially be used as well since they pertain to friction factors in tubes [Perry and Green, 2015; Churchill, 1977]. As I intend the library to be flexible, I do not wish to tightly couple the calculation of Nu and f_{darcy}. I instead allow the user to choose how he or she likes to calculate f_{darcy}. Equations 2.41 and 2.39 account for the entrance length using a d/L term in their respective equations [Gnielinski, 2013], where d is the representative diameter, usually D_H and L is the pipe length. As the flow develops, $L \to \infty$ and $d/L \to 0$ and these equations reduce to a form suitable for fully developed flow shown in Equation 2.42:

$$\mathrm{Nu}_{turb} = \frac{(f_{darcy}/8)(\mathrm{Re} - 1000)\,\mathrm{Pr}}{1 + 12.7\sqrt{(f_{darcy}/8)}(\mathrm{Pr}^{2/3} - 1)} \left(\frac{\mathrm{Pr}}{\mathrm{Pr}_{surf}}\right)^{0.11} \tag{2.42}$$

Except for the Pr_{wall} term the form resembles the Gnielinski correlation in Bejan's textbook [Bejan, 2013]. Readers should note that the Fanning friction factor is used as opposed to Darcy friction factor in some pieces of literature such as Bejan's work [Bejan, 2013] and in Perry's Chemical Engineering handbook [Perry and Green, 2015]:

$$f_{darcy} = 4 f_{fanning}$$

Aside from this small detail, we have shown how Gnielinski's correlation accounts for flow in turbulent, transitional and laminar flow, as well as developing and fully developed

flow. This makes the equations suitable for general flow in straight pipes such as those in CIET.

On the Presence and Significance of Developing Flow in CIET

For the sake of determining a suitable entrance length, we must consider the flow paths of the fluid prior to entering a fluid component. If the flow traverses in one direction, this can be hard coded. However, if the flow is expected to traverse in both forward and reverse direction, implementing this in code can get much more complex. Given this, we want to ascertain if we can ignore entrance effects within CIET. This has implications on how we determine d/L for the fluid for the purposes of determining Nu. To see if we can ignore entrance length effects, we first need to check if these effects are present in CIET. Secondly, if these effects are present, we want to check if we can ignore them.

Presence of Entrance Length Effects We can investigate the presence of entrance effects first by finding a suitable length scale over which flow becomes fully developed. To do so, we can consider developing flows in circular ducts for both laminar and turbulent flow since many of the components in CIET are simply cylindrical pipes. We can gauge the lengths of these pipes prior to bends and see how these compare to the lengths required for fully developed flow.

For CIET, the longest pipe lengths are on the order of 2 m, while most pipes and components are on the order of 0.1 m. In between these piping components, there are bends, flow meters, static mixers, heat exchangers and even the heater as they contain objects such as spheres and twisted tapes [De Wet and Per F Peterson, 2020] intended to enhance heat transfer [Lukas, Kendrick, and P. Peterson, 2017] or ensure a well mixed flow. If we refer once more to Figure 2.4, we can see that there are several of these objects lie within the flow path of the fluid. Moreover, the pipes themselves may have thermocouples inserted in the flow to measure bulk fluid temperature, as well as valves. These objects may disrupt a fully developed boundary layer and may prevent the flow from ever developing fully.

A table of CIET's components is listed in previous literature [Ong, 2023; Nicolas Zweibaum, 2015; Zou, R. Hu, and Charpentier, 2019], including my master's thesis. I reproduce the piping schematics for the isothermal digital twin model from my master's thesis in Table 2.9:

Component Description	P&ID Label For Mixers And Flowmeters	Number on Diagram	Component Length (m)	Vertical Angle (°)	Hydraulic Diameter (m)	Flow area (m²)	Form Loss K (dimensionless)
Heater Branch (Bottom to Top)							
Pipe		18	0.1778	-40.0052	2.79E-02	6.11E-04	5.15
Heater bottom head		1b	0.19685	90	6.60E-03	3.64E-04	3.95
Heater		1	1.6383	90	6.60E-03	3.64E-04	0
Heater top head		1a	0.0889	90	6.60E-03	3.64E-04	3.75
Static mixer pipe		2a	0.149425	90	2.79E-02	6.11E-04	1.8
Static mixer	MX-10	2	0.33	90	2.79E-02	6.11E-04	*
Pipe		3	1.2827	90	2.79E-02	6.11E-04	3.15
Pipe		4	0.2413	49.743387	2.79E-02	6.11E-04	2.4
Branch		5	0.7493	0	2.79E-02	6.11E-04	0
CTAH Branch (Top to Bottom)							
Static mixer pipe		6a	0.1526	51.526384	2.79E-02	6.11E-04	5.05
Static mixer	MX-41	6	0.33	51.526384	2.79E-02	6.11E-04	*
CTAH (Vertical)		7a	0.3302	-90	1.19E-02	1.33E-03	3.9
CTAH (Horizontal)		7b	1.2342	0	1.19E-02	1.33E-03	***
Static mixer pipe		8a	0.22245	-90	2.79E-02	6.11E-04	3.75
Static mixer	MX-40	8	0.33	-90	2.79E-02	6.11E-04	*
Pipe		9	0.7112	-42.73211	2.79E-02	6.11E-04	0.8
Pipe		10	2.4511	-90	2.79E-02	6.11E-04	0.45
Pipe		11	0.4826	-63.47465	2.79E-02	6.11E-04	2.4
Pipe		12	0.333375	0	2.79E-02	6.11E-04	21.65
Pipe		13	1.273175	0	2.79E-02	6.11E-04	12.95
Pipe		14	0.6687	90	2.79E-02	6.11E-04	2.4
Flowmeter	FM-40	14a	0.36	90	2.79E-02	6.11E-04	**
Pipe		15	0.3556	-49.36983	2.79E-02	6.11E-04	0.8
Pipe		16	0.644525	-90	2.79E-02	6.11E-04	1.9
Branch		17	0.473075	0	2.79E-02	6.11E-04	0
DHX Branch (Bottom to Top)							
Pipe		19	0.219075	-31.44898	2.79E-02	6.11E-04	7.5
Pipe		20	0.33655	0	2.79E-02	6.11E-04	0
Pipe		21	0.487725	90	2.79E-02	6.11E-04	4.4
Flowmeter	FM-20	21a	0.36	90	2.79E-02	6.11E-04	**
Pipe		22	0.69215	90	2.79E-02	6.11E-04	9.95
Static mixer pipe		23a	0.0891	90	2.79E-02	6.11E-04	1.35
Static mixer	MX-20	23	0.33	90	2.79E-02	6.11E-04	*
DHX shell side		24	1.18745	90	5.65E-03	9.43E-04	23.9
Static mixer pipe		25a	0.22245	90	2.79E-02	6.11E-04	1.35
Static mixer	MX-21	25	0.33	90	2.79E-02	6.11E-04	*
Pipe		26	0.2159	52.571994	2.79E-02	6.11E-04	1.75

Table 2.9: Hydrodynamic Parameters used for the Rust Model [Ong, 2023]

*** For CTAH correlation, see Equation 2.43

** For Flowmeter correlation, see Equation 2.44

* For Static Mixer correlation, see Equation 2.45

For the Coiled Tube Air Heater (CTAH), the following correlation is used [Zweibaum, J E Bickel, et al., 2015]:

$$(f_{darcy}\frac{L}{D_{hydraulic}} + K) = 18 + \frac{93,000}{\mathrm{Re}_{D_{hydraulic}}^{1.35}} \tag{2.43}$$

Here in Equation 2.43, K represents form loss. For all Coriolis Flowmeters, the following correlation is used [Zweibaum, J E Bickel, et al., 2015]:

$$(f_{darcy}\frac{L}{D_{hydraulic}}+K)=400+\frac{52,000}{\mathrm{Re}_{D_{hydrualic}}} \tag{2.44}$$

For all static mixers, the following correlation is used [Zweibaum, J E Bickel, et al., 2015]:

$$(f_{darcy}\frac{L}{D_{hydraulic}}+K)=21+\frac{4000}{\mathrm{Re}_{D_{hydrualic}}} \tag{2.45}$$

Table 2.9 was produced using information from the RELAP model of CIET [Nicolas Zweibaum, 2015]. The component numbers are meant to mostly correlate to Figure 2.4 except for the branches 5 and 17, which are split into 5a and 5b, as well as 17a and 17b on Figure 2.4. The exact details are not discussed here for brevity. Some pipes such as pipe 4 are as short as 0.24 m, while some pipes such as pipe 10 are as long as 2.4 m. Fluid flowing in these pipes would often encounter bends. For example, a bend exists between pipe 3 and pipe 4, thus disrupting any fully developed flow if it is present. If we wish to ignore entrance effects, the entrance lengths must be short with respect to these pipe lengths. Let us consider some entrance length scales for this purpose.

For entrance lengths, we need to consider at least two bounding cases for pipes based on both laminar and turbulent flow regimes. We shall first consider laminar regimes before moving on to turbulent regimes. The hydrodynamic entrance X length for laminar flows can be estimated as [Bejan, 2013]:

$$\frac{X}{D_{hydrualic}}\approx 0.04\mathrm{Re}_{D-hydrualic} \tag{2.46}$$

For circular ducts, such as some pipes within CIET, literature recommends [Perry and Green, 2015]:

$$\frac{X}{D_{hydrualic}}\approx 0.05\mathrm{Re}_{D-hydrualic} \tag{2.47}$$

For laminar duct flow, the thermal entrance length (X_T) for fluids with Pr >> 1 scales linearly with Pr [Bejan, 2013; Perry and Green, 2015]:

$$\frac{X_T}{X}\sim \mathrm{Pr} \tag{2.48}$$

For fully developed hydrodynamic and thermal boundary layer flow in cylindrical ducts, we can use [Perry and Green, 2015]:

$$\frac{max(X,X_T)}{D_{hydrualic}}\approx 0.05\mathrm{Re}_{D-hydrualic}\mathrm{Pr} \tag{2.49}$$

For laminar flow of Therminol VP-1, we need Pr to calculate the ratio of the X to X_T. Pr is about 11.1 to 14. Therefore, the thermal entrance length is at least 10 times more than the hydrodynamic entrance length. We encounter this situation in laminar flow regimes,

usually when CIET operates in natural circulation mode. Here, the flow rates are on the order of 0.04 kg/s [Nicolas Zweibaum, 2015]. To estimate the relevant X in this regime, we can first calculate $Re_{D_{hydraulic}}$ for a typical pipe in CIET. We use $D_{hydraulic}$ of $2.79 \times 10^{-2} m$ [Nicolas Zweibaum, 2015], a flow area of 6.11×10^{-4} [Nicolas Zweibaum, 2015] and a mass flow rate of 0.04 kg/s. μ is taken at an average temperature of about $60°C$, which is a typical temperature occuring during natural circulation.

$$\begin{aligned} \mathrm{Re}_{D-hydrualic} &= \frac{\rho u D_{hydraulic}}{\mu} = \frac{\dot{m} D_{hydraulic}}{A_{XS}\mu} \\ &= \frac{0.04(kg/s) \times 2.79 \times 10^{-2} m}{6.11 \times 10^{-4} m^2 0.0008425(Pa \cdot s)} \approx 1132 \end{aligned} \tag{2.50}$$

Based on Equation 2.50 and 2.47, the ratio $X/D_{hydrualic}$ is about 56. For a hydraulic diameter of 2.79 cm, we obtain an entrance length scale of approximately 158 cm or 1.58 m. If we consider lengths required for thermally fully developed flows, the entrance length is about 16 m. Given that CIET's longest pipe is on the order of 2 to 3 m, we can conclude that in laminar regimes, the flow is almost never fully developed.

Next, let us consider turbulent flow regimes. CIET experiences some turbulent flow during forced circulation within its pipes. We can verify this by calculating $\mathrm{Re}_{D_{hydraulic}}$. To do so, we can use the same pipe diameters and cross sectional areas as before, but the mass flow rate $\dot{m}$ is about 0.25 kg/s. This is one of the higher flow rates recorded in literature [Lukas, Kendrick, and P. Peterson, 2017]. For μ, we consider that a typical fluid temperature of CIET within forced circulation is approximately 110 $°C$ [De Wet and Per F Peterson, 2020]. At this temperature, μ is about 0.000843 $Pa \cdot s$. We can substitute these values in to find a typical value of $Re_{D_{hydraulic}}$ in Equation 2.51:

$$\begin{aligned} \mathrm{Re}_{D-hydrualic} &= \frac{\rho u D_{hydraulic}}{\mu} = \frac{\dot{m} D_{hydraulic}}{A_{XS}\mu} \\ &= \frac{0.25(kg/s) \times 2.79 \times 10^{-2} m}{6.11 \times 10^{-4} m^2 0.0008425(Pa \cdot s)} \approx 13,550 \end{aligned} \tag{2.51}$$

Equation 2.51 shows that turbulent flow in the pipes would exist during forced circulation for CIET as Re > 4000. Therefore, we need to consider turbulent duct flow. Hydrodynamically fully developed flow in tubes occurs when [Bejan, 2013; Perry and Green, 2015]:

$$X \geq 10 D_{hydraulic} \tag{2.52}$$

$$X_T \geq 10 D_{hydraulic} \tag{2.53}$$

$$\tag{2.54}$$

Equation 2.52 is applicable for fluids with Pr ~ 1 [Bejan, 2013] as well as fluids with Pr of about 7 to 200 [Hartnett, 1955] with high Re. Therefore, it is applicable to the Therminol

VP-1 Pr range found within CIET. We should be careful though when using Equation 2.52. Based on Hartnett's experimental data with Freezene oil with Pr $\sim$ 100 [Hartnett, 1955], one should note that for lower Re turbulent flows such as Re $\sim$ 5500, $X_T/D_{hydrualic}$ increases to a value of 20 or 30. At Re of about 10,100, $X_T/D_{hydrualic}$ is about 12. Therefore, one should be more judicious with using Equation 2.52 especially for lower Re turbulent flows. This can plausibly occur for lower mass flow rates at lower temperatures with higher viscosities. For example, a flow of Therminol VP-1 at $80°C$ at 0.18 kg/s in the same pipes has Re $\approx$ 7000. This is typical of flow exiting the CTAH and entering the heater [De Wet and Per F Peterson, 2020]. In this flow regime, the $X_T/D_{hydrualic}$ is more than 10.

Now, let us consider the scale of these entrance lengths. For the purposes of determining a lower bound estimate of entrance length effects for turbulent flow, we can simply use $X_T/D_{hydrualic} \geq 10$ as a lower bound estimate for X_T. In this case, X_T is about 27.9 cm or 0.279 m, based on a hydraulic diameter of 2.79 cm. Given that a significant number of pipes shown in Table 2.9 are less than 1m, we can conclude that entrance effects may be important for many of these pipe. If we consider lower Re flows, the entrance lengths may be twice as long, or roughly 55 cm. In such regimes, we should consider entrance lengths and developing flow.

Significance of Entrance Length Effects Now that we have verified that entrance length effects are present in CIET regardless of flow regime, we need to consider the significance of entrance length effects as well. For flow in CIET's insulated pipes, entrance lengths impact the calculation of Nu. In turn, Nu is important for calculating parasitic heat loss. Parasitic heat loss is in turn important to account for in modelling CIET. In previous models of CIET such as those in RELAP5, the parasitic heat loss was often underestimated based by up to 75% [Nicolas Zweibaum, 2015]. This was attributed to losses through metallic components protruding from CIET's insulated pipes. Even when the metallic components were insulated based in infra-red camera data, the discrepancies persisted [Nicolas Zweibaum, 2015]. Hence, the approach taken then was to scale the heat transfer coefficients by a multiplication factor [Nicolas Zweibaum, 2015] in order to ensure that the model matched experimental data. This process was known as "model calibration".

Model calibration was also done with the Transform model [De Wet and Per F Peterson, 2020]. However, the Transform model accomplishes this by modelling metallic structural supports within CIET as heat structures within Transform. The structural supports were modelled according to measurements of structural supports within CIET. The supports were then connected to a thin wall with convective thermal resistance to ambient air boundary conditions [De Wet and Per F Peterson, 2020]. This was stated to have accounted for most of the parasitic heat loss. No mention was made about accounting for entrance length effects. Since the Transform did not account for entrance effects from the start, one could then infer that entrance lengths effects did not account for the bulk of parasitic heat losses within CIET. These were not important to account for as compared to the metallic support structures.

While I initially wanted to replicate the heat structure model, I was unable to due to

a lack of published data and dimensions for CIET's support structures. I also considered measuring these structural supports by hand. However, this was too far outside the scope of this dissertation as I only needed an approximate model of CIET's Heater. I decided to leave the study and modelling of parasitic heat losses within CIET to future work. As long as the model reproduced experimental data for CIET's Heater, this would suffice for this dissertation.

Given this state of affairs, it seemed that entrance effects may not be matter significantly in this context. Nevertheless, entrance length effects are already contained within Gnielinski's model [Gnielinski, 2013]. This is relatively convenient to put into my code. The capability to account for entrance lengths within the heat transfer library may have been useful to future users of the library. Hence, I included the capability to model entrance lengths in case they were needed in future.

To calculate d/L, I just chose the simplest method possible. I used the length of the fluid component to calculate d/L and the resulting Nu. This simplified model would be used in MX-10. The overall heater and MX-10 model would be validated against experimental data. We shall cover this in the next chapter.

Thermal Inertia and Thermal Resistance for Pipes

For thermal inertia and thermal resistance, we may refer to some piping design and performance parameters for CIET in Table 2.10 [De Wet and Per F Peterson, 2020]:

Name	Value
Heat Transfer Fluid	Dowtherm-A or Therminol VP-1
Wall Thickness	0.277 cm
Wall Inner Diameter	2.79 cm
Wall Roughness	$1.5 * 10^{-5} m$
Insulation Material	Fiberglass
Insulation Outer Diameter	5-inch
Insulation Inner Diameter	1-inch (approx)
Insulation Thickness	2-inch
Ambient Temperautre	$21.67°C$
Ambient Heat Transfer Coefficient	20 $W/(m \cdot K)$

Table 2.10: CIET Piping Design and Performance Parameters [De Wet and Per F Peterson, 2020]

I can use the same thermophysical properties for the piping since it is made of a similar steel to the heater. However, I also need to consider its fiberglass insulation. For fiberglass, the basis of the thermal conductivity data used in RELAP models is [Nicolas Zweibaum, 2015]:

$$k\left(\frac{W}{m\cdot K}\right) = 7.702 * 10^{-4} T(^{\circ}C) + 0.206 \tag{2.55}$$

However, in both RELAP the SAM model, this correlation was used to obtain thermal conductivity values and put into a table. I used this table for computation. Fibreglass density is 20 kg/m^3 and fiberglass heat capacity (c_p) is 844 $J/(kg\cdot K)$. The values of c_p and k are provided Table 2.11:

Temperature (K)	k ($W\ m^{-1}\ K^{-1}$)	c_p ($J\ kg^{-1}\ K^{-1}$)
250	0.028616	844
293.15	0.03306	844
350	0.038916	844
400	0.044066	844
500	0.054366	844
600	0.064666	844

Table 2.11: Thermophysical Properties of Fiberglass [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015]

Modelling Details for Static Mixer MX-10

Since we are interested in modelling MX-10, we also want to investigate the modelling details for MX-10. In the RELAP5 model, MX-10 was modelled using a static mixer pipe (component 2 and 2a) of of length 0.149425 m and a static mixer of length 0.33 m as seen in Table 2.9.

Let us discuss the thermal resistance, thermal inertia and nodalisation details for MX-10.

Thermal Resistance and Thermal Inertia for MX-10 For the purposes of heat transfer modelling, MX-10 is modelled as the components 2 and 2a in Table 2.9 and in Figure 2.4. These are modelled as cylinders of the same length and hydraulic diameter as seen in Table 2.9. For simplified modelling, the SS 304L wall thickness is assumed to be 0.277 cm, same as for any other pipe in Table 2.11. The fibreglass insulation is 2 inch thick. This fibreglass would supply the bulk of the thermal resistance to outside air. The fibreglass to air thermal resistance is based on the outer diameter of the fibreglass, 5 inches, and a heat transfer coefficient of 20 $W/(m^2\cdot K)$ to keep this consistent with h of the heated tube to air.

For MX-10, we also use the Gnielinski correlation for calculating the Nu. Pr and Re are evaluated at the control volume temperature. The friction factor is based on Table 2.9, which specifies an experimentally determined correlation for MX-10 based on isothermal pressure drop experiments [Nicolas Zweibaum, 2015]. This friction factor is then substituted into the Gnielinski correlation to estimate Nu. The entrance length value used for Nu for both component 2 and 2a are their respective component lengths. This is 0.149425 m and 0.33 m for component 2a and component 2 respectively.

For axial connections between component 2a and its adjacent components as seen in Figure 2.4, thermal conductance is zero, which means there is an adiabatic boundary. Only fluid advection is allowed to transfer heat between fluid control volumes. The adiabatic boundary also exists for adjacent solid control volumes. Likewise, for component 2, there is a similar adiabatic boundary.

For thermal resistance and thermal inertia, the fluid control volumes are modelled as cylinders with diameters of $2.79 \times 10^{-2}m$. The SS 304L wall surrounding the Therminol VP-1 has a thickness is assumed to be 0.277 cm. Finally, the same 2 inch thick insulation is used to calculate thermal inertia of the fibreglass node.

Nodalisation for MX-10 For MX-10 nodalisation, I took reference from the SAM model seeing that each of these components contains two axial nodes [Zou, R. Hu, and Charpentier, 2019]. Hence, in the simulations I validated in the next chapter, two control volumes are used for component 2 and 2a each in the axial direction. This is a total of four control volumes modelling MX-10. As before, these control volumes are equally sized for both solid and fluid control volumes. Radially, it is composed of an inner Therminol VP-1 fluid node, coupled to a SS 304L steel node. This steel node is coupled to an additional fibreglass insulation node via a cylindrical conductive thermal resistance. The centre of the fibreglass node is then connected via conductive thermal resistance and convective thermal resistance to the outside air. This is can be seen in Figure 2.25:

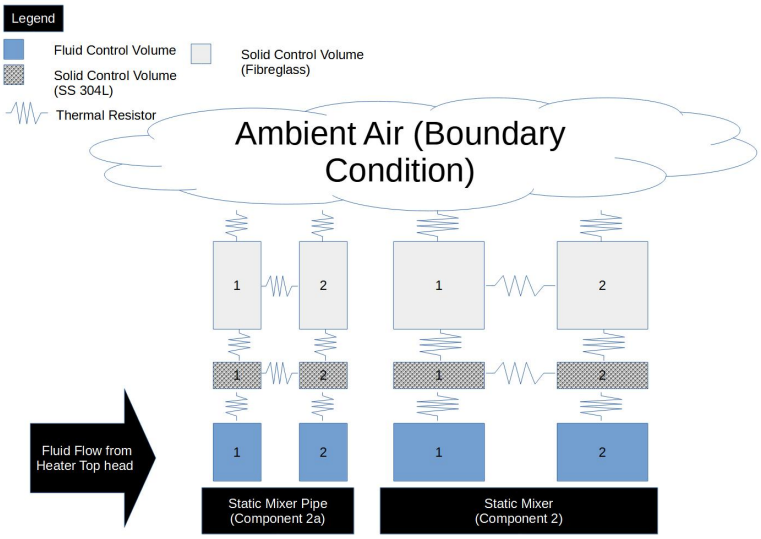


Figure 2.25: MX-10 Nodalisation Diagram (Not to Scale)

The number of radial nodes was not specified in the SAM model. However, I used one radial node only because I needed to reduce the computational burden to ensure calculations are run in real-time. Furthermore, I was not interested in the temperature profile of MX-10's steel shell or fibreglass insulation. Moreover, the main impact that the modelled MX-10 component has on the exit temperature is that of residence time, and a negligible heat loss. Based on early simulations, this was on the order of 0.01 K at steady state, or an order of magnitude less than the Type T thermocouple measurement uncertainty of $\pm$ 0.5 K [Zweibaum, Guo, et al., 2016]. Again, these measurement uncertainties are typical of Type T thermocouples used within CIET [Zweibaum, Guo, et al., 2016]. I found it extremely implausible that increasing the number of nodes would have any appreciable impact on parasitic heat loss. Therefore, I left the number of radial nodes for the steel and fibreglass at one each.

Heat Exchangers, Fans and Coolers

Now, heat exchangers are an integral part of CIET. They are mainly meant to remove heat from the primary loop to the DRACS loop, or to the surrounding air. From Figure 2.4, there are three main heat exchangers of interest that exchange heat between fluids. In the CTAH branch, we have the Coiled Tube Air Heater (CTAH), which removes heat from the primary loop during forced circulation. It models the Intermediate Heat Exchanger (IHX) which transfers heat into a working fluid for power production. In natural circulation mode , the DRACS Heat Exchanger (DHX) removes heat from the primary loop and moves it into the DRACS loop. The heat is removed from the DRACS loops via a Thermosiphon Cooled

Heat Exchanger (TCHX) [Nicolas Zweibaum, 2015]. This TCHX is also called the Natural Air Draft DRACS salt to air Heat Exchanger (NDHX) [Raluca Olga Scarlat, 2012].

For the purposes of this dissertation, we are more concerned with constructing a Digital Twin of the Heater rather than the whole loop. Therefore, work on heat exchangers is left for future work. However, a brief dossier containing important information for CTAH modelling is given in the Appendix. This is because we will eventually have to model the CTAH when we simulate forced circulation transients, such as ULOHS, within CIET.

Support Structures

The support structures responsible holding CIET in its place also happen to be responsible for parasitic heat losses in CIET [De Wet and Per F Peterson, 2020]. A figure in De Wet's Dissertation outlines where these support structures reside [2]. It is to be noted that the figure containing information on these support structures does not correspond directly to the nodalised models used in SAM or RELAP similar to Figure 2.4. We can only estimate where these support structures may reside. With reference to Figure 2.4, it seems that support structures are present on pipe 12, 13 and 14 in the pipe manifold area, as well as pipe 16 and 17a at the junction between the heater and CTAH branch. For the heater branch, pipe 3 and 4 seem to be modelled with connection to support structures. For the remainder of the CTAH branch, pipe 5b and pipe 10 seem to be connected to support structures as well.

For this dissertation, detailed investigation and calibration of support structures is not done yet for simplicity. However, the code responsible for conduction along the supports is implemented so that the computational burden of simulating such supports can be estimated. I do this so that I have assurance that the Digital Twin in future can run in real-time even with the added computational burden of simulating the structural supports.

For this, I added some dummy support structures to the heater top and bottom head. The support structures are crudely modelled as 1 ft long and 0.5 inch diameter SS 304L cylinders. These length scales are based on crude measurements I made with some support structures in CIET. A fellow thermal hydraulics laboratory member who has since graduated, Omar Ashraf Alzaabi, has taken these measurements in much more detail than I have. He has graciously allowed me to use some of his photographs detailing the measurements for this dissertation. Figure 2.26 shows one of his photographs, which I compressed, of how a typical support structure in CIET looks like:

[2] The appendix in De Wet's dissertation on approximately page 135 Appendix D provides a figure on where the support structures are [De Wet and Per F Peterson, 2020]. The copy I have, however, is a draft. But I think the information is still somewhat reliable.

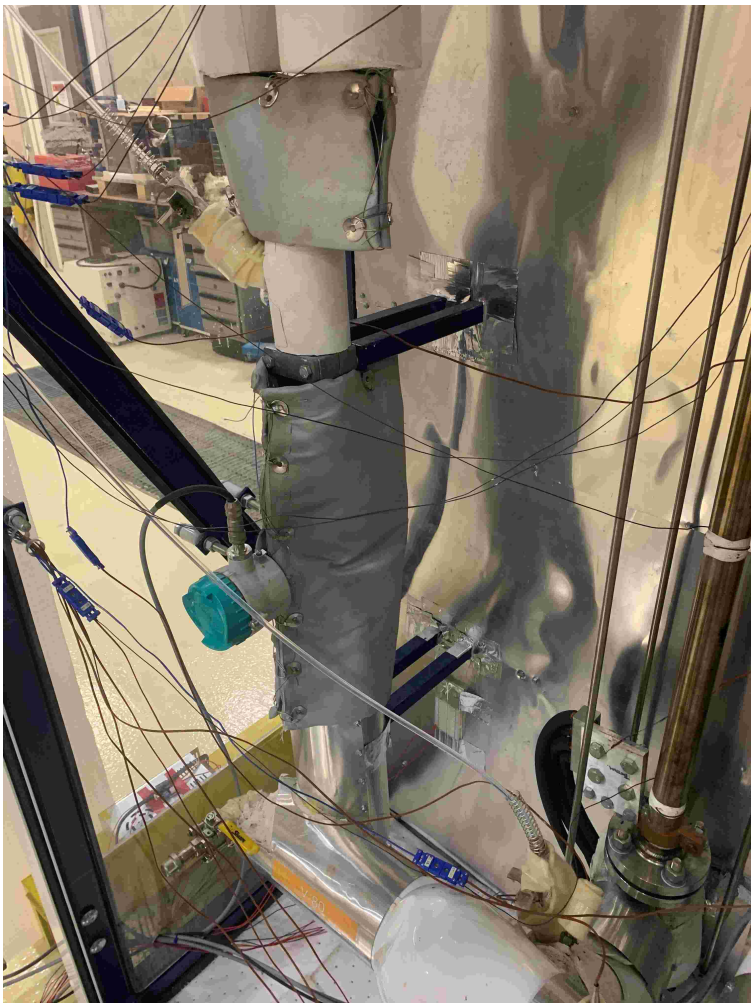


Figure 2.26: Photograph of DHX Support Structure, Credit Omar Ashraf Alzaabi

The support structures in Figure 2.26 are shown in blue. These support structures are typically L shaped as shown in Figure 2.27:

Figure 2.27: Photograph Emphasising L Shape of Support Structure near Value V81 in CIET, Credit Omar Ashraf Alzaabi

Most of these support structures have similar breadth and width of about 2.3 cm as shown in Figure 2.28:

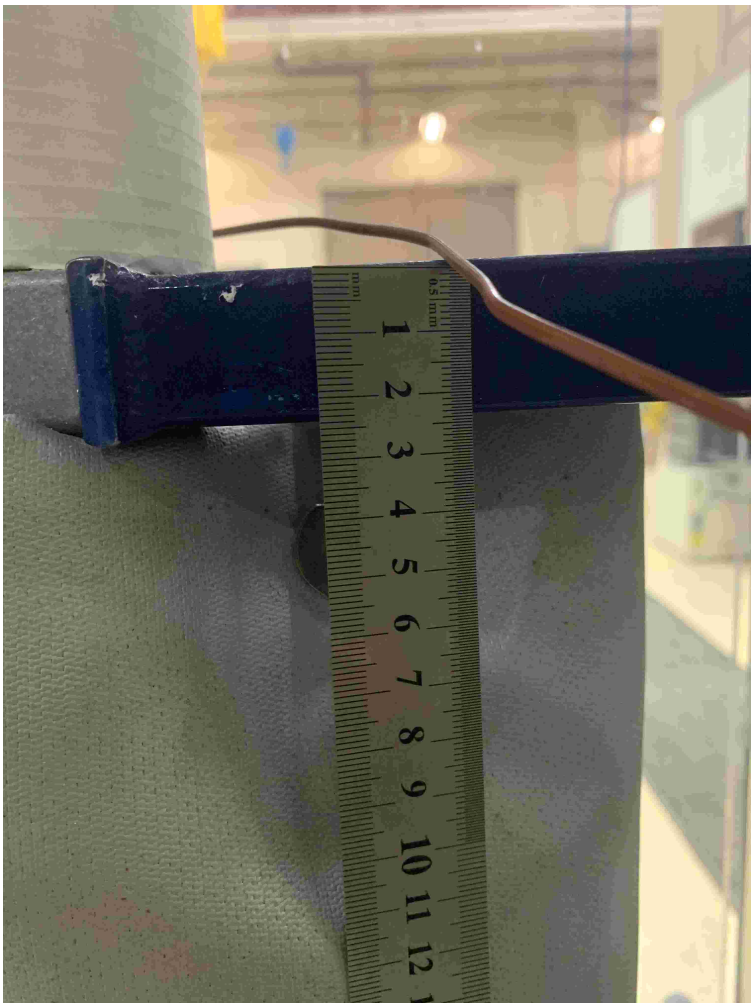


Figure 2.28: Photograph of showing Support Structure Width of about 2.3 to 2.4 cm in CIET near DHX Branch, Credit Omar Ashraf Alzaabi

The length of the blue L shaped structure is about 23 cm long as shown in Figure 2.29:

Figure 2.29: Photograph of showing Support Structure Length of about 23 cm in CIET near DHX Branch, Credit Omar Ashraf Alzaabi

However, the blue L shaped support itself is connected to another grey square like structure (I presume to be a bracket) holding the pipe as shown in Figure 2.26. The length of its side is shown in Figure 2.30:

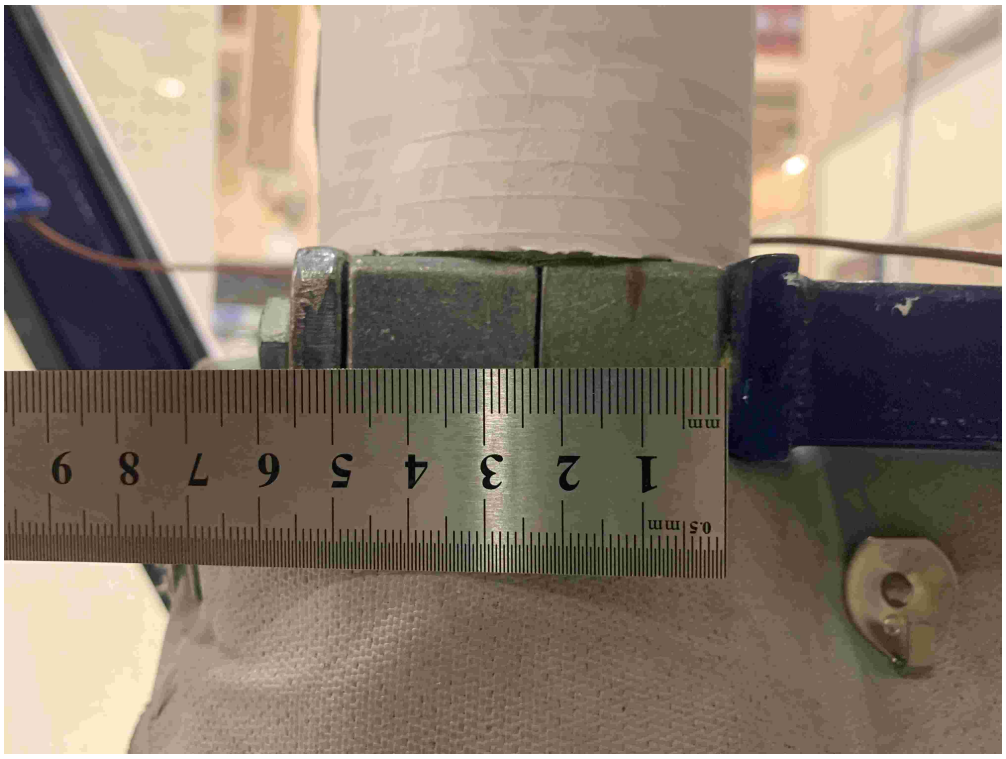


Figure 2.30: Photograph of showing Square like Support Structure (Presumably a Bracket) with Side Length of about 5.5 cm in CIET near DHX Branch, Credit Omar Ashraf Alzaabi

For now, detailed modelling of these support structures is left to future work. Nevertheless, I estimated a length scale of these support structures to be about 30 cm long or about 1 foot based on adding 5.5 cm to 23 cm. I also simplified the modelling of the L shaped support of about one inch long at the sides as a half inch diameter cylinder. This is how I arrived at the dimensions of a 1 foot long, half inch diameter cylinder to crudely model the support structures.

I then added coupled one of these dummy cylinders to the heater top head and one to the bottom head with their respective thermal resistance based on the length of the structure. I also added one more dummy cylinder support structure to the Static Mixer MX-10 mixer pipe, labelled 2a on Figure 2.4. This was for the purposes of simulating the computational burden incurred by these structures. With these additions, the final simulation still ran within an acceptable time frame. We will discuss this more in the next chapter.

2.5 Review of Solver Stability Issues for Heat Transfer Solvers

Now, in the course of performing numerical analysis for transients, stability is an important issue. This is especially the case if the solver has any degree of explicit coupling as mentioned

earlier in the chapter. To ensure that the simulation is stable, the methodology of choosing of the appropriate time step becomes very important. This is because we want to maximise accuracy and stability of the solver while minimising computational cost [Rossi et al., 2014].

This is a complex problem because different transients entail different time scales. For CIET, a frequency response test would entail at least three time scales. These would pertain to frequency, fluid residence time and conjugate heat transfer time scales [Zou, R. Hu, and Charpentier, 2019]. However, where conduction is concerned, we would then consider the mesh Fourier number [Salem, M. Errera, and Marty, 2019]. In conjugate heat transfer (CHT), we may run into multiple of such time scales. However, for stability, a smaller timestep usually implies better stability. Hence, the timestep should be determined by considering the smallest of these time scales. Since the timestep is directly dependent on the time scales for transients and heat transfer, it is important to review what some of these time scales are so that we may apply them to the thermal hydraulics library.

Discussion on Explicit and Implicit Coupling

Of course, in the context of system codes such as RELAP5, the Courant number limitations have been circumvented via the use of semi implicit or implicit coupling schemes [Aumiller, Tomlinson, and Bauer, 2001]. Implicit or semi implicit coupling schemes have generally worked well and are used in Computational Fluid Dynamics (CFD) codes and multiphysics codes such as OpenFOAM and GeN-Foam [Fiorina, I. Clifford, et al., 2015]. Implicit Coupling usually entails placing all components into a matrix where we need to solve $\mathbf{A}\vec{x} = \vec{b}$. However, should the matrix size grow too large, computational costs will add up. This is because computational costs for matrix multiplication scale as $\mathcal{O}(n^{2.37188})$ [Duan, H. Wu, and Zhou, 2022] where $\mathbf{A}$ is an $n \times n$ matrix and $\vec{b}$ has n elements. Hence, we generally would not want the matrix to grow too large, otherwise the benefits gained from allowing a larger time step would be offset by additional computational costs due to having a large matrix. These large matrices usually arise from coupling the entire system tightly using an implicit scheme. If we were to insist on writing a solver using fully implicit coupling schemes from scratch, would not only take more time to test and develop compared to an explicit (loose) coupling, it would also make the software architecture tightly coupled or monolithic [Fernández, 2011]. From a software engineering perspective, tightly coupled software architecture is undesirable in comparison to loose coupling [Pautasso and Wilde, 2009]. This is because tight coupling makes the program more difficult to modify and maintain in comparison to a loosely coupled codebase as it is less modular [Fernández, 2011]. In general, we want to avoid a fully implicit coupling scheme if the matrices necessitated by implicit coupling grow too large.

One other consideration when using a fully implicit time stepping approach is that we still have an upper limit for the time step we use. While we are not restricted by stability, we are still restricted by the time scales of the phenomena we need to simulate. For example, in frequency response transient simulation, we are limited by the time scales set by the perturbation frequencies, fluid residence time, and conjugate heat transfer (CHT) time scales

[Zou, R. Hu, and Charpentier, 2019]. Thus, we cannot increase the time step indefinitely even if we were to use the implicit time stepping schemes.

Non-linearities and Semi Implicit Coupling Example using 1D Conduction

Also, with tight (fully implicit) coupling, there is a good chance we will run into nonlinear equations in the solution procedure [Fernández, 2011]. One very simple example is an enthalpy change calculation. The enthalpy change of a control volume depends on the specific heat capacity of a volume (c_p). For this simple example, let us discuss what the control volume calculations over a typical control volume in a 1D array for solid conduction. The energy balance over a control volume in such an array is:

$$\frac{\partial[mc_p(T)T]}{\partial t} = \frac{\partial}{\partial x}\left(k(T)A_{XS}\frac{\partial T}{\partial x}\right)$$
$$\frac{\partial[\rho V c_p(T)T]}{\partial t} = \frac{\partial}{\partial x}\left(k(T)A_{XS}\frac{\partial T}{\partial x}\right) \tag{2.56}$$

In Equation 2.56, $k(T)$ is thermal conductivity, T is temperature, x is length, $m = \rho V$ is the mass of the differential volume dV, where ρ is density and V is volume. $c_p(T)$ is the temperature dependent heat capacity, and A_{XS} is the cross sectional area of the 1D array of control volumes. Now, c_p and $k(T)$ are both functions of temperature, and during the discretisation process, we would have non linear terms to deal with. To show this, let us discretise equation 2.56 using an implicit time scheme, but we assume $k(T)$ and ρ are either averaged or invariant with T for a simpler illustration:

$$\rho\Delta V\frac{c_p(T_x^{t+\Delta t})T_x^{t+\Delta t} - c_p(T_x^t)T_x^t}{\Delta t} = kA_{XS}\frac{T_{x+\Delta x}^{t+\Delta t} - 2T_x^{t+\Delta t} + T_{x-\Delta x}^{t+\Delta t}}{2\Delta x}$$

In these derivations, T_x^t is the temperature at position x and time t. The mesh is uniform with spacing Δx and the time step is Δt.

$$\Delta x A_{XS}\rho\frac{c_p(T_x^{t+\Delta t})T_x^{t+\Delta t} - c_p(T_x^t)T_x^t}{\Delta t} = kA_{XS}\frac{T_{x+\Delta x}^{t+\Delta t} - 2T_x^{t+\Delta t} + T_{x-\Delta x}^{t+\Delta t}}{2\Delta x}$$

$$\frac{2\Delta x^2}{k}\rho\frac{c_p(T_x^{t+\Delta t})T_x^{t+\Delta t} - c_p(T_x^t)T_x^t}{\Delta t} = T_{x+\Delta x}^{t+\Delta t} - 2T_x^{t+\Delta t} + T_{x-\Delta x}^{t+\Delta t}$$

c_p is in turn dependent on temperature and is often a polynomial function. If we assume that in the simplest case, c_p varies linearly with T:

$$c_p(T) = a_1 + a_2T$$

Then:

$$\frac{2\Delta x^2}{k}\rho\frac{[a_1 + a_2T_x^{t+\Delta t}]T_x^{t+\Delta t} - [a_1 + a_2T_x^t]T_x^t}{\Delta t} = T_{x+\Delta x}^{t+\Delta t} - 2T_x^{t+\Delta t} + T_{x-\Delta x}^{t+\Delta t}$$

$$\frac{2\Delta x^2}{k}\rho\frac{[a_1T_x^{t+\Delta t}+a_2(T_x^{t+\Delta t})^2]-[a_1T_x^t+a_2(T_x^t)^2]}{\Delta t}=T_{x+\Delta x}^{t+\Delta t}-2T_x^{t+\Delta t}+T_{x-\Delta x}^{t+\Delta t} \tag{2.57}$$

Now, Equation 2.57 has a nonlinear $(T_x^{t+\Delta t})^2$ term which makes the resulting matrix more difficult to solve as using a linear scheme. Regardless, this nonlinear problem would be challenging to solve in real-time. If we wish to linearise this problem, we could either assume c_p is constant with t, or else use the values of c_p at the last time step. This is because we need not worry about $(T_x^t)^2$ since the values at the time t are fixed. If we use the latter approach, we obtain Equation 2.58:

$$\frac{2\Delta x^2}{k}\rho\frac{[a_1T_x^{t+\Delta t}+a_2T_x^tT_x^{t+\Delta t}]-[a_1T_x^t+a_2(T_x^t)^2]}{\Delta t}=T_{x+\Delta x}^{t+\Delta t}-2T_x^{t+\Delta t}+T_{x-\Delta x}^{t+\Delta t} \tag{2.58}$$

Equation 2.58 allows us to solve a linear equation to obtain the temperature profile at the next step. It is also semi implicit because the c_p at the previous time step was used. While Equation 2.58 allows us to solve a linearised heat transport equation in a semi-implicit manner, we must be careful if the mesh grows too large such that we cannot solve the equations in real-time. This can happen if we tightly couple too many components such that we obtain a large matrix to solve.

Semi Implicit Coupling for 1D Fluid and Solid Control Volumes

Now for fluid control volumes, we need to simulate more than heat conduction. Suppose we add in the mass, energy and momentum balance equations. If we were to couple them explicitly, then we will have a multivariable nonlinear system of equations to solve. As I found out in my previous work, solving a nonlinear system of equations would not only slow down calculations, the solver itself may not even converge on a solution [Ong, 2023]. These were the main motivating factors for me to use an operator split method mentioned earlier in the chapter. In operator splitting, the mass, energy and momentum are coupled explicitly as shown earlier. This allows me to linearise some of the resulting equations. The resulting time marching scheme is semi-implicit. Therefore, some level of explicit solutions should be expected.

Now suppose that in the energy balance, we want to consider the discretisation of the energy equations. We already derived an explicitly discretised form of the energy equation in earlier Equation 2.25:

$$
\begin{aligned}
m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i}-h^{t}_{enthalpy,i}}{\Delta t} &= \sum_{j}^{N} H^{t}_{thermal,self\ \leftrightarrow\ j,i}(-T^{t}_{i}+T^{t}_{j,i}) \\
&+ \dot{m}^{t}(-h^{t}_{enthalpy,i}+h^{t}_{enthalpy,i-1}) \\
&+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t}_{i}+T^{t}_{i-1}) \\
&+ H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t}_{i}+T^{t}_{i+1}) \\
&+ \sum Q^{t}_{gen,i} + \sum Q^{t}_{boundary\ conditions,i}
\end{aligned}
\tag{2.25}
$$

Except for the mass flow rate, and equivalent energy balance discretised using an implicit time marching scheme would take the form:

$$
\begin{aligned}
m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i}-h^{t}_{enthalpy,i}}{\Delta t} &= \sum_{j}^{N} H^{t+\Delta t}_{thermal,self\ \leftrightarrow\ j,i}(-T^{t+\Delta t}_{i}+T^{t+\Delta t}_{j,i}) \\
&+ \dot{m}^{t}(-h^{t+\Delta t}_{enthalpy,i}+h^{t+\Delta t}_{enthalpy,i-1}) \\
&+ H^{t+\Delta t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t+\Delta t}_{i}+T^{t+\Delta t}_{j,i}) \\
&+ H^{t+\Delta t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t+\Delta t}_{i}+T^{t+\Delta t}_{j,i}) \\
&+ \sum Q^{t+\Delta t}_{gen,i} + \sum Q^{t+\Delta t}_{boundary\ conditions,i}
\end{aligned}
\tag{2.59}
$$

As mentioned before, we may not want to solve the (almost) fully implicit Equation 2.59 because the thermal conductances, enthalpies and even local heat generation terms (for resistive heating) are generally functions of T. To linearise this nonlinear problem, we can take evaluate the thermal conductances based on temperatures of the last time step. Thus, this becomes a semi-implicit form of the discretised equation. The boundary conditions themselves may be based on thermal conductances especially if the boundary condition is a user set constant temperature, if this is so, then the thermal conductances can also be evaluated at the last time step. We may also simplify the equation further by specifying that volumetric heat generation terms are evaluated at the previous time step as well in case these are nonlinear with temperature.

$$
\begin{aligned}
m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i}-h^{t}_{enthalpy,i}}{\Delta t} &= \sum_{j}^{N} H^{t}_{thermal,self\ \leftrightarrow\ j,i}(-T^{t+\Delta t}_{i}+T^{t+\Delta t}_{j,i}) \\
&+ \dot{m}^{t}(-h^{t+\Delta t}_{enthalpy,i}+h^{t+\Delta t}_{enthalpy,i-1}) \\
&+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t+\Delta t}_{i}+T^{t+\Delta t}_{j,i}) \\
&+ H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t+\Delta t}_{i}+T^{t+\Delta t}_{j,i}) \\
&+ \sum Q^{t}_{gen,i} + \sum Q^{t}_{boundary\ conditions,i}
\end{aligned}
$$

Next, we also need to evaluate enthalpies as the system of equations is now a function of $h^{t+\Delta t}_{enthalpy}$ and T. In general, we cannot always assume $h^{t+\Delta t}_{enthalpy,i}$ to be a linear function of temperature. For this, there are two methods of linearising this function. Firstly, we only consider enthalpies at the previous timestep for the sake of calculating the advection term.

$$\begin{aligned} m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} &= \sum_{j}^{N} H^{t}_{thermal,self \leftrightarrow j,i}(-T^{t+\Delta t}_{i} + T^{t+\Delta t}_{j,i}) \\ &+ \dot{m}^{t}(-h^{t}_{enthalpy,i} + h^{t}_{enthalpy,i-1}) \\ &+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t+\Delta t}_{i} + T^{t+\Delta t}_{j,i}) \\ &+ H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t+\Delta t}_{i} + T^{t+\Delta t}_{j,i}) \\ &+ \sum Q^{t}_{gen,i} + \sum Q^{t}_{boundary\ conditions,i} \end{aligned}$$

$h^{t}_{enthalpy,i}$ can then be calculated using any non-linear thermophysical property correlation. This would then account for the T dependence of c_p. For the enthalpy change with temperature, we can assume those changes to be small over each time step such that $c^{t}_{p} \approx c^{t+\Delta t}_{p}$. The enthalpy change term can then take the form:

$$h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i} = c_p(T^t)[T^{t+\Delta t} - T^t]$$

Thus, we have a semi-implicit, linearised form of the energy equation:

$$\begin{aligned} m_{cv}c_p(T^t)\frac{T^{t+\Delta t} - T^{t}}{\Delta t} &= \sum_{j}^{N} H^{t}_{thermal,self \leftrightarrow j,i}(-T^{t+\Delta t}_{i} + T^{t+\Delta t}_{j,i}) \\ &+ \dot{m}^{t}(-h^{t}_{enthalpy,i} + h^{t}_{enthalpy,i-1}) \\ &+ H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t+\Delta t}_{i} + T^{t+\Delta t}_{j,i}) \\ &+ H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t+\Delta t}_{i} + T^{t+\Delta t}_{j,i}) \\ &+ \sum Q^{t}_{gen,i} + \sum Q^{t}_{boundary\ conditions,i} \end{aligned}$$

The second approach is similar to the first approach, except that for the sake of the advection term, we assume a roughly constant c_p over the timestep as well as the temperature range of the problem. In that case:

$$-h^{t}_{enthalpy,i} + h^{t}_{enthalpy,i-1} = -c_p(T^t_i)[T^t_i - T_{ref}] + c_p(T^t_{i-1})[T^t_{i-1} - T_{ref}]$$

Where T_{ref} is some reference temperature. If we assume the temperatures between adjacent control volumes are similar enough such that $c_p(T^t_{i-1}) \approx c_p(T^t_i)$, then the reference enthalpies cancel out.

$$-h^t_{enthalpy,i} + h^t_{enthalpy,i-1} = -c_p(T^t_i)T^t_i + c_p(T^t_{i-1})T^t_{i-1}$$

With this, we arrive at another form of the semi-implicit equations:

$$\begin{aligned} m_{cv}c_p(T^t)\frac{T^{t+\Delta t} - T^t}{\Delta t} &= \sum_j^N H^t_{thermal,self \leftrightarrow j,i}(-T^{t+\Delta t}_i + T^{t+\Delta t}_{j,i}) \\ &+ \dot{m}^t(-c_p(T^t_i)T^t_i + c_p(T^t_{i-1})T^t_{i-1}) \\ &+ H^t_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t+\Delta t}_i + T^{t+\Delta t}_{j,i}) \\ &+ H^t_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t+\Delta t}_i + T^{t+\Delta t}_{j,i}) \\ &+ \sum Q^t_{gen,i} + \sum Q^t_{boundary\ conditions,i} \end{aligned}$$

Of these two options, I chose to evaluate the enthalpies at the last time step as this seemed to account for the temperature dependence of c_p better of these two schemes.

$$\begin{aligned} m_{cv}c_p(T^t)\frac{T^{t+\Delta t} - T^t}{\Delta t} &= \sum_j^N H^t_{thermal,self \leftrightarrow j,i}(-T^{t+\Delta t}_i + T^{t+\Delta t}_{j,i}) \\ &+ \dot{m}^t(-h^t_{enthalpy,i} + h^t_{enthalpy,i-1}) \\ &+ H^t_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t+\Delta t}_i + T^{t+\Delta t}_{j,i}) \\ &+ H^t_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t+\Delta t}_i + T^{t+\Delta t}_{j,i}) \\ &+ \sum Q^t_{gen,i} + \sum Q^t_{boundary\ conditions,i} \end{aligned}$$

Now, axial conduction, or the conduction term in the direction of the flow can be neglected for high Pe flows. To program this out in the thermal hydraulics library, I would need to calculate a localised Pe based on the geometry of the piping system since Pe = Re Pr. If Pe is below some threshold amount, then I would include axial conduction, and then I would have to write code for the axial conduction case. This means I would need to have two versions of the energy balance in my library and write code to check for Pe. This means more code to write and more code to debug, and an overall longer development time. Hence, I went to check if the computational burden for axial conduction was significant. I found out that in early tests simulating control volumes for CIET, the additional computational cost of calculating axial conduction was insignificant if the axial conduction calculation was carried out efficiently. This means calculating an average axial conduction only once for the whole time step for the whole 1D array of fluid nodes. This was carried out successfully. Therefore, I decided to include it without checking for Pe. Of course, for flows in CIET, Pe was still high enough such that including axial conduction did not visibly or significantly change any temperature simulation results. Due to this, I removed the code checking for Pe, and just calculated axial conduction in all cases when using the semi-implicit solver.

The average axial conduction over the whole set of control volumes was calculated based on the node length, the average temperature of the 1D array of fluid control volumes at the previous time step T^t_{avg} where:

$$T^t_{avg} = \sum_i \frac{V_i}{V_{total}} T^t_i$$

Where V_i is the volume of the control volume index i and,

$$V_{total} = \sum_i V_i$$

This would enable the solver to obtain a rough estimate of axial conduction for fluids at rest. For 1D arrays of solid control volumes, we can set the fluid flow rate to zero and the same equation would apply. For solid control volumes describing the heated steel pipes and piping, we may be able to neglect axial conduction under certain conditions [Vera and Quintero, 2018] where axial heat flows are small in comparison to radial heat flows. However, rather than develop code for whether to neglect axial conduction based on these conditions, I found it more convenient to re-use the same axial conduction approach and same code for the 1D fluid array for the 1D solid array. While using an averaged axial conductance for generalised 1D solid arrays in this case may not be ideal for accuracy, it turns out that this is crucial for getting the calculations to run in real-time as calculating individual conductances for each node can get computationally expensive and prevent the ability for these computations to run in real-time. Hence, I used this approximation.

Thus, we arrive at:

$$\begin{aligned} m_{cv} c_p(T^t) \frac{T^{t+\Delta t} - T^t}{\Delta t} = &\sum_j^N H^t_{thermal,self \ \leftrightarrow \ j,i}(-T^{t+\Delta t}_i + T^{t+\Delta t}_{j,i}) \\ &+ \dot{m}^t(-h^t_{enthalpy,i} + h^t_{enthalpy,i-1}) \\ &+ H^t_{avg,axial}(T^{t+\Delta t}_{j,i} - 2T^{t+\Delta t}_i + T^{t+\Delta t}_{j,i}) \\ &+ \sum Q^t_{gen,i} + \sum Q^t_{boundary \ conditions,i} \end{aligned}$$

Now, we have a general energy balance equation for our solid and fluid control volumes. Of course, the direction of flow can be accounted for by changing the advection term accordingly, and energy balance for solids can again be obtained by setting the advection term to zero. Since we have a suitable discretised energy balance to solve, we must consider how to construct the matrices in a practical manner.

Partitioned Solution Schemes

As mentioned earlier, the 1D fluid arrays and solid arrays are meant to be laterally coupled in order to account for the number of nodes in the radial direction. If we were to couple their

temperatures implicitly, we would have to construct several different matrices depending how many radial nodes we have for the insulation and steel piping. If we have a 1D array of fluid control volumes with a solid steel tube represented by three radial nodes, we would then have to program code to construct a matrix for the 1D array of fluid control volumes coupled to three 1D arrays of solid control volumes to form an essentially 2D mesh for conjugate heat transfer. Suppose for now that I want to add insulation with two nodes. I would have to rewrite code in order to add the control volumes representing the insulation. Let's also suppose that I want to add internal unheated steel structures such as the twisted tape for CIET Heater v2.0. I would again have to re-write code for this configuration. From a programming standpoint, having to write code for different conjugate heat transfer configurations is quite cumbersome. This is one problem I needed to consider.

Next, we also have to deal with axial coupling, this is usually done for components adjacent to each other. Again, If I couple the heated section control volumes to several components, I would have to write code to construct matrices to solve for the temperature profile. If the configuration somehow changes such I need to add heat exchangers in future, structural supports, then I would have to write custom code for each of those configurations, or at least write code to build those matrices using several nested for loops. This is also quite cumbersome. Moreover, it makes the code so tightly coupled that it is difficult to modify for other configurations. This limits the applicability of this developed heat transfer library to more general cases. Hence, I want to consider developing code in a more modular fashion.

One solution that helped me develop a more modular heat transfer library is the use of partitioned solution methods. Partitioned solution methods are another kind of semi-implicit method [Fernández, 2011]. This is where a strongly coupled implicit solution procedure is broken up into two or more parts. Each part is then implicitly solved on its own before coupling them together. Let us again use a simple 1D conduction problem in order to illustrate this point.

$$\frac{\partial T}{\partial t} = \alpha \frac{\partial^2 T}{\partial x^2} \tag{2.60}$$

In Equation 2.60, α is the thermal diffusivity where $\alpha = \frac{k}{\rho c_p}$. We can now discretise Equation 2.60 using an implicit time scheme with a mesh length of Δx. Suppose that this mesh has N nodes and we consider node i:

$$\frac{T_i^{t+\Delta t} - T_i^t}{\Delta t} = \alpha \frac{T_{i+1}^{t+\Delta t} - 2T_i^{t+\Delta t} + T_{i-1}^{t+\Delta t}}{\Delta x^2}$$

Suppose this mesh was to be partitioned at node i. This process is shown in Figure 2.31:

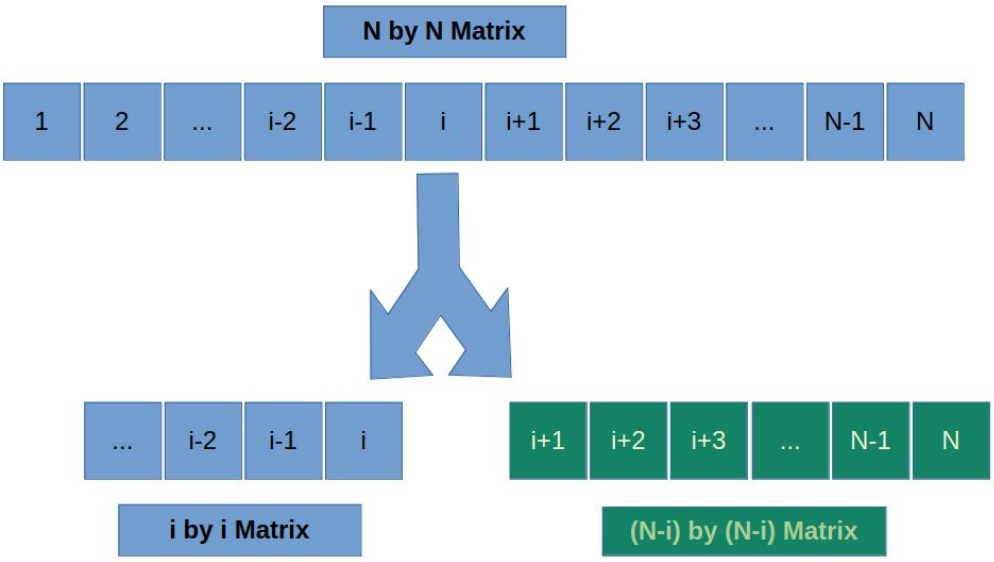


Figure 2.31: Mesh Partitioning for 1D Conduction Problem at node i

In Figure 2.31, we implicitly couple the two partitioned 1D array of control volumes. However, we explicitly couple these two smaller arrays. The main changes apply to control volume i and control volume $i+1$. At control volume i, the equation describing it is now:

$$\frac{T_i^{t+\Delta t} - T_i^t}{\Delta t} = \alpha \frac{T_{i+1}^t - 2T_i^{t+\Delta t} + T_{i-1}^{t+\Delta t}}{\Delta x^2}$$

For node $i+1$, the equation before the partitioning process is:

$$\frac{T_{i+1}^{t+\Delta t} - T_{i+1}^t}{\Delta t} = \alpha \frac{T_{i+2}^{t+\Delta t} - 2T_{i+1}^{t+\Delta t} + T_i^{t+\Delta t}}{\Delta x^2}$$

After the partitioning process, we obtain:

$$\frac{T_{i+1}^{t+\Delta t} - T_{i+1}^t}{\Delta t} = \alpha \frac{T_{i+2}^{t+\Delta t} - 2T_{i+1}^{t+\Delta t} + T_i^t}{\Delta x^2}$$

Let us now apply this to our 1D heat transfer equations for the fluid control volumes:

$$
\begin{aligned}
m_{cv}c_p(T^t)\frac{T^{t+\Delta t}-T^t}{\Delta t} &= \sum_j^N H^t_{thermal,self \leftrightarrow j,i}(-T_i^{t+\Delta t}+T_{j,i}^{t+\Delta t}) \\
&+ \dot{m}^t(-h^t_{enthalpy,i}+h^t_{enthalpy,i-1}) \\
&+ H^t_{avg,axial}(T^{t+\Delta t}_{i+1,i}-2T_i^{t+\Delta t}+T^{t+\Delta t}_{i-1,i}) \\
&+ \sum Q^t_{gen,i} + \sum Q^t_{boundary\ conditions,i}
\end{aligned}
$$

For lateral thermal conductances, the $\sum_j^N H^t_{thermal,self \leftrightarrow j,i}(-T_i^{t+\Delta t}+T_{j,i}^{t+\Delta t})$ term, I used partitioned solution methods to explicitly couple radially coupled control volumes. Hence, for the temperatures of some other laterally coupled control volume j, I use its temperature at the previous time step to calculate heat flux. I show this in Equation 2.61:

$$
\begin{aligned}
m_{cv}c_p(T^t)\frac{T^{t+\Delta t}-T^t}{\Delta t} &= \sum_j^N H^t_{thermal,self \leftrightarrow j,i}(-T_i^{t+\Delta t}+T_{j,i}^{t}) \\
&+ \dot{m}^t(-h^t_{enthalpy,i}+h^t_{enthalpy,i-1}) \\
&+ H^t_{avg,axial}(T^{t+\Delta t}_{i+1,i}-2T_i^{t+\Delta t}+T^{t+\Delta t}_{i-1,i}) \\
&+ \sum Q^t_{gen,i} + \sum Q^t_{boundary\ conditions,i}
\end{aligned}
\tag{2.61}
$$

Again, we can eliminate the advection term to apply Equation 2.61 to solid control volumes. If the control volume happens to be at the tail end, it can be coupled to any other control volume based on a constant temperature boundary condition at that time step with conductance $H^t_{avg,axial\ partitioned}$ and temperature $T^t_{i+1,i}$:

$$
\begin{aligned}
m_{cv}c_p(T^t)\frac{T^{t+\Delta t}-T^t}{\Delta t} &= \sum_j^N H^t_{thermal,self \leftrightarrow j,i}(-T_i^{t+\Delta t}+T_{j,i}^{t}) \\
&+ \dot{m}^t(-h^t_{enthalpy,i}+h^t_{enthalpy,i-1}) \\
&+ H^t_{avg,axial}(-T_i^{t+\Delta t}+T^{t+\Delta t}_{i-1,i}) \\
&+ H^t_{avg,axial\ partitioned}(-T_i^{t+\Delta t}+T^{t}_{i+1,i}) \\
&+ \sum Q^t_{gen,i} + \sum Q^t_{boundary\ conditions,i}
\end{aligned}
\tag{2.62}
$$

A partitioned so allows me the flexibility to connect control volumes to each other in a modular fashion so that I do not need to write new code to construct matrices for different heater configurations and piping configurations. Additionally, the computational cost of solving for the matrices is smaller because each partitioned matrix is small. While there will still be some cost associated with solving matrices, partitioned schemes help mitigate the cost

of computation by keeping the matrix size small. With fully implicit timestepping schemes, the allowable timestep for stability increases due to the removal of the Courant (or other) stability restraints. However, the computational costs also increase within that timestep [Duan, H. Wu, and Zhou, 2022]. This may jeopardise our ability to run the simulation in real-time. To ensure that the simulation runs in real-time, the computational time required for a simulated timestep must be faster than that said timestep. That means if the simulated timestep is 0.1 s, the actual computational time required must be less than 0.1 s in order for the simulation to be real-time. Thus, while allowable timesteps for stability do increase in an implicit scheme, the increase in computational time could more than offset any benefit afforded by the timestep increase, thus making it more difficult for the simulation to run in real-time. This is not desirable. However, when one breaks a problem into smaller explicitly coupled subdomains, we mitigate the computational costs of a growing matrix by keeping that matrix small. In fact, if we explicitly couple arrays of control volumes which are internally implicitly coupled, we can also parallelise the problem more readily when a problem is loosely coupled computationally. Thus, more options for speeding up the simulation open up when semi-implicit coupling is allowed. As shown in the discussions leading up to Equation 2.61, we can achieve this by coupling the control volumes within 1D solid mesh and 1D fluid mesh of a pipe component in a semi-implicit manner, and we could then couple the 1D solid and fluid meshes together explicitly. We could also explicitly couple interactions between pipes using an explicit scheme. Overall, the coupling scheme is semi-implicit. This would keep the matrices small and simple, and relatively easy to solve. While the explicit coupling of the separate partitions necessitate the consideration of a suitable timestep for stability, the advantages for explicitly coupling partitioned 1D meshes more than offset this problems caused by this issue.

It may be also advantageous from a code development standpoint to use semi-implicit solvers to some degree because it is common for existing open source codes to be written using an implicit or semi-implicit time marching scheme. I could then adapt existing code into my thermal hydraulics codebase using the solver so that testing and development time is shorter. One example in literature is the GeN-Foam thermal conductance model used to model heat transfer in pebbles [Robert et al., 2023] released under the open source GNU GPL v3 license. Thus, a partitioned solution scheme or semi-implicit coupling scheme would help speed up development time.

Given these considerations, I decided to use the semi-implicit coupling methodology where 1D control volumes arrays are internally coupled semi-implicitly, but when coupling with other 1D control volume arrays, they are explicitly coupled. To solve the resulting matrices, I used optimised Basic Linear Algebra Subprograms (BLAS) libraries such as OpenBLAS [Xianyi, Qian, and Yunquan, 2012] to perform matrix multiplications and solve the matrix quickly enough to ensure that the library can support real-time simulations using consumer gaming Laptops. These 1D control volume solvers were then validated using analytical solutions. More details will be presented in the following chapter. Before that, we still need to be mindful of certain timescales because the 1D control volume arrays are still explicitly coupled with each other.

Courant Number

Since we likely have to use at least some form of explicit coupling, we will have to consider how to choose the right timestep for stability. In convection heat transfer, one key stability criteria is determined using the Courant Friedrichs Lewy (CFL) number (also known the Courant Number for short [Y. Liu, 2020]). This pertains specifically to advection. In advection, the time step needs to be adjusted such that the Courant number (Co) is not above unity. For a simple control volume (CV) with one stream of fluid flowing in and out of the CV, the Co number is expressed as:

$$Co_{\max} = \frac{\Delta t_{\max} u}{\Delta x} \tag{2.63}$$

Where u is the prevailing bulk fluid velocity. We can simply rearrange this equation to obtain the timestep:

$$\Delta t_{\max} = \frac{\Delta x}{u} Co_{\max} \tag{2.64}$$

In the case of control volumes, we cannot always assume that there is only one stream of fluid entering and exiting the CV. Therefore, another technique for calculating Co is required. Fortunately, such generalised expressions for Co already exist in literature. For OpenFOAM, based on the documentation, the timestep based on the generalised Co is [OpenFOAM, 2023]:

$$\Delta t_{max} = \frac{Co_{max}}{2V_{cell}} \sum_{i=1}^{N} |u_i \bullet A_i| \tag{2.65}$$

Where V_{cell} is the volume of the CV (also known as "cell") in OpenFOAM, N is the number of faces for a particular control volume, u_i is the velocity vector at face i, A_i is the area of face i and the $\bullet$ represents a dot product. Interested readers can explore methodology for obtaining the generalised expressions for Courant Number in literature [Rauter et al., 2021].

Fourier Number

For convection heat transfer, at lower flow rates and where conduction heat transfer becomes dominant, then the Fourier number (Fo) becomes important for determining time step [Hensen and Nakhi, 1994; Thomas, Samarasekera, and Brimacombe, 1984]. Now, this is not just any kind of Fo, but it is one associated with mesh lengthscales. This Fo is important in conjugate heat transfer as well, where conduction heat transfer becomes important. Since Fo is important for the thermal hydraulics library, let us consider how Fo determines the maximum allowable timestep.

For illustration, we consider the example of heat conduction using explicit time step schemes. In this case, the following criterion must be satisfied [Hensen and Nakhi, 1994]:

$$\frac{\alpha \Delta t}{\Delta x^2} \leq 0.25 \tag{2.66}$$

The quantity in Equation 2.66 is the Fourier number (Fo). In this discussion about time stepping and stability, Δx refers to grid length and Δt refers to time step. Also, to distinguish this Fo from other Fo, I shall call this $Fo_{discretisation}$ because this Fourier number deals with mesh discretisation and time step discretisation. In literature, $Fo_{discretisation}$ is also known as the "mesh Fourier Number" [Salem, M. Errera, and Marty, 2019]:

$$Fo_{discretisation} \equiv \frac{\alpha \Delta t}{\Delta x^2} \tag{2.67}$$

As the time stepping scheme becomes more implicit, a larger $Fo_{discretisation}$ becomes permissible. The numerical scheme is stable when [Hensen and Nakhi, 1994]:

$$Fo_{discretisation} \leq \frac{1}{4(1-\gamma)} \tag{2.68}$$

γ is a weighting parameter to calculate temperature gradients and can be set by the user to determine the extent to which the time stepping scheme is explicit or implicit. We select γ such that $0 < \gamma < 1$. $\gamma = 0$ implies a fully explicit time stepping scheme, $\gamma = 0.5$ implies a Crank-Nicolson scheme and $\gamma = 1$ means a fully implicit scheme [Hensen and Nakhi, 1994].

To understand the significance of γ we may consider the appropriate the heat conduction equation [Hensen and Nakhi, 1994]:

$$\frac{\partial T(x,t)}{\partial t} = \alpha \frac{\partial^2 T(x,t)}{\partial x^2} \tag{2.69}$$

Again, $T(x,t)$ is temperature of the system at position x and time t. This can be discretised using a time step Δt and a mesh length Δx into [Hensen and Nakhi, 1994]:

$$\frac{T_x^{t+\Delta t} - T_x^t}{Fo_{discretisation}} = \gamma\big(T_{x+\Delta x}^{t+\Delta t} - 2T_x^{t+\Delta t} + T_{x-\Delta x}^{t+\Delta t}\big) + (1-\gamma)\big(T_{x+\Delta x}^{t} - 2T_x^{t} + T_{x-\Delta x}^{t}\big) \tag{2.70}$$

Where T_x^t is $T(x,t)$ but written in subscripts and superscripts for ease of reading.

While the phenomena determining the time scales in conduction is different from advection, the mesh Fo is quite similar in function and concept to the Courant number criterion. Interested readers may want to explore literature based on Von Neumann Stability Analysis [Wesseling, 1996] or Fourier Stability Analysis. One would find that such stability analysis methods often deals with how various system frequencies or modes respond to a typical error perturbation introduced into the system [Tadmor, 1987]. One could even draw analogies between this method and frequency response since system stability can be determined from a system's frequency response. Except that now, instead of actual system noise, the source of the disturbance to the system is numerical error. Interested readers may wish to refer to "Numerical integration of the barotropic vorticity equation" [Charney, Fjörtoft, and Neumann, 1950] for more information.

Application of Stability Criteria to Solid-Fluid Interactions

Now, Co and the mesh Fo deal with advection and conduction individually. However, at solid-fluid boundaries, a different time scale for stability is required. This time scale may be different from the conjugate heat transfer time scale t_{CHT} since t_{CHT} deals with the phenomena of a physical time scale while we wish to discuss a time scale suitable for numerical stability. In simplified analysis, assuming a constant surrounding temperature (T_∞), t_{CHT} can be expressed as [Zou, R. Hu, and Charpentier, 2019]:

$$t_{CHT} = \frac{\delta \rho c_p}{h} \tag{2.71}$$

In Equation 2.71, δ refers to the characteristic length scale. In the case of the heater wall exchanging heat with some fluid, δ refers to the wall thickness [Zou, R. Hu, and Charpentier, 2019]. In this context, ρ and c_p represents the heater wall density, and specific heat capacity. h represents the heat transfer coefficient between wall and fluid. The timescale was derived by letting the product Bi Fo = 1, where Bi is the Biot number and Fo is the Fourier number [Zou, R. Hu, and Charpentier, 2019]. One possible timescale is letting δ be the mesh length scale Δx. While this analysis is not nearly as rigourous as deriving such a timescale using Von Neumann analysis, it can still give us a clue as to how the timescale should scale with important system parameters. In effect, $t_{CHT\ \Delta x}$ should scale as $\frac{\Delta x \rho c_p}{h}$.

We can first look to literature, where several methods for determining such a time step exist. We can use a "mesh Biot number" [Salem, M. Errera, and Marty, 2019] and modified mesh Fourier number [M.-P. Errera and Chemin, 2013]. In most of these finite volume calculations at the mesh boundary, the effect of advection is often ignored at the first cell adjacent to the solid boundary [Salem, M. Errera, and Marty, 2019]. This is because at the first cell next to the wall, the fluid velocity is zero if we use a no slip condition. This makes it in essence a classic conduction case where we do not have to make any modification to the mesh Fourier number in order to calculate a suitable timestep. For a coarse mesh nodalised approach, we do not impose no slip conditions on the fluid. Therefore, we cannot directly apply the exact same methods used for high fidelity computational fluid dynamics (CHT). We may have to develop something different. We might use some other kind of analysis to obtain a suitable time step. For this, we could perhaps adapt the mesh Fourier number for the solid-fluid boundary. Traditionally, we use Fo to quantify a suitable time step between two solid control volumes. Using explicit time stepping, we can calculate the maximum time step:

$$Fo_{\max\ \ discretisation} = \frac{\alpha \Delta t_{\max}}{\Delta x^2} \tag{2.72}$$

$$\Delta t_{\max} = 0.25 \frac{\Delta x^2}{\alpha} \tag{2.73}$$

Δx represents the distance between the two centres of the control volumes. In other words, it is the typical mesh timescale. α is the thermal diffusivity usually in units similar

to m^2/s.

$$\alpha_{fluid} = \frac{k_{fluid}}{\rho_{fluid} c_{p,fluid}} \tag{2.74}$$

Within α, $\rho_{fluid} c_{p,fluid}$ represents the volumetric heat capacity of the material within the cell. This does not change whether the material in the control volume is solid or fluid. However, k governs the heat flux within the system for conduction heat transfer. In fact, the thermal conductance over each mesh unit is governed by $\frac{kA}{\Delta x}$ where A is some basis for heat transfer area.

For conjugate heat transfer (CHT) at solid-fluid boundaries, the heat transfer coefficient h becomes important for calculating thermal conductance and heat flux. The thermal conductance is simply hA. Now, suppose we try to obtain an equivalent thermal conductivity for the solid-fluid convection heat transfer $k_{fluid\ convection}$ based on equating the conductances or heat fluxes. The basis areas A cancel out, and so we obtain:

$$h(T_s - T_{fluid}) = k_{fluid\ convection} \frac{\partial T}{\partial x} \tag{2.75}$$

$$h(T_s - T_{fluid}) = k_{fluid\ convection} \frac{T_s - T_{fluid}}{\Delta x} \tag{2.76}$$

It is apparent that:

$$h\Delta x = k_{fluid\ convection} \tag{2.77}$$

And the ratio between k_{fluid} and $k_{fluid\ convection}$ is essentially a local Nusselt Number:

$$\mathrm{Nu}_{\Delta x} = \frac{h\Delta x}{k_{fluid}} \tag{2.78}$$

Now, to substitute these expressions into the Fourier number, we can obtain an equivalent thermal diffusivity based on convection heat transfer $\alpha_{fluid\ convection}$:

$$\alpha_{fluid\ convection} = \frac{k_{fluid\ convection}}{\rho_{fluid} c_{p,fluid}} = \frac{h\Delta x}{\rho_{fluid} c_{p,fluid}} = \alpha_{fluid} \mathrm{Nu}_{\Delta x} \tag{2.79}$$

Now, we defined Δx in the context of fluid cell length scales or control volume length scales. Therefore, Δx should be based upon cell geometry. For a cartesian like geometry, Δx could be based on cell length. However, for the thermal hydraulics library, we find ourselves dealing more with pipe like geometries. Therefore, a suitable length scale would be based on the radius of the pipe r_{pipe}. This is because r_{pipe}, would be the distance between the pipe centre and the surface of the wall. This is important to consider since most geometries in CIET would contain heat transfer from fluid in the pipe to the wall, or vice versa. Now, to obtain $\mathrm{Nu}_{\Delta x}$, we still need some sort of correlation to obtain it. In the context of pipes, Nu_D, the Nusselt number based on pipe diameter D, is often used for heat transfer correlations. We can be use Nu_D to determine $\mathrm{Nu}_{\Delta x}$. Alternatively, for convenience, we may wish to do

away with using $\mathrm{Nu}_{\Delta x}$ altogether and just use Nu_D. To do this, we may rewrite Equation 2.79 as:

$$\alpha_{fluid\ convection} = \alpha_{fluid}\mathrm{Nu}_D * \frac{\Delta x}{D} = \alpha_{fluid}\mathrm{Nu}_D * \frac{r_{pipe}}{D} = 0.5\alpha_{fluid}\mathrm{Nu}_D \tag{2.80}$$

We could then use Equation 2.80 to calculate the Fourier Number for the convection near the pipe boundary as:

$$\Delta t_{\max} = 0.25\frac{\Delta x^2}{0.5\alpha\mathrm{Nu}_D} = 0.5\frac{\Delta x^2}{\alpha\mathrm{Nu}_D} \tag{2.81}$$

It should be noted that Figure 2.81 is an approximation which depends on how Δx is selected for various geometries. However, it has the effect that if α or Nu_D increases, the $\Delta t_{\max}$ should decrease. Increasing either α or Nu_D should have the effect of increasing the apparent thermal conductivity of the system. This makes sense. Moreover, if we deal with solid control volumes, Nu_D would become Bi and we would arrive at a timescale consistent in form with t_{CHT} in Equation 2.71. Thus, it confirms that the analysis can scale time step scales with mesh size, α, k and h in a manner consistent with the scaling analysis in literature [Zou, R. Hu, and Charpentier, 2019]. This would become a starting point for us to perform design iterations of adaptive time stepping.

Error Considerations in Timestep Adjustment

Besides simulation stability, accuracy should also be maintained when solving transient heat transfer simulations numerically. If stability is not an issue, then time step adaptation strategies can focus more on controlling the time step truncation error [Maffulli et al., 2018]. One example this comes into play is in the calculation of heat transfer rates across each time step. For conduction and convection k and h are usually temperature dependent. However, we linearise the expression by assuming these quantities are constant during the time step advancement. If we were to compare heat transfer rates depending on whether or not we assume k and h are constant over the time step, we would obtain one form of the truncation error. It is important to keep such truncation methods small. Therefore, there have been many methods have been discussed in literature on how to control the time step truncation error.

One method that this is done is step doubling [Maffulli et al., 2018]. Suppose we can obtain a temperature solution using a time step Δt. This solution can be compared to a solution where two time steps of $\Delta t/2$ are used. The difference we see is known as the truncation error.

A second method we can use is where we use a lower order method with which to measure truncation error [Maffulli et al., 2018]. Suppose we are using a typical 4th order Runge-Kutta method typically used for numerical solution of initial value problems [Perry and Green, 2015], we may obtain a decent solution based on this time step. We can then compare this solution to a lower order Euler method to then estimate the truncation error. Should this

truncation error be small compared to a user set tolerance, then the Δt would increase. If it is larger than some user set tolerance, Δt would decrease. Such methods are used to accelerate the solution procedure so that, if the solution is already stable, we need not become excessively conservative by adjusting time step based on Co or $Fo_{discretisation}$ [Maffulli et al., 2018].

Most time stepping adaptation methods in literature are discussed in the context of finite volume methods (FVM), finite difference methods and finite element methods (FEM) [Mohan and Tamma, 1994]. While we are using a lower fidelity model for the thermal hydraulics library, it is still useful to borrow some concepts from FVM methods and see if these could be adapted for the Digital Twin constructed here.

For this purpose, let us consider a simplified version of the adaptive time stepping strategy adopted by Mohan [Mohan and Tamma, 1994]. Suppose that there for a time step advancement calculation, we wish to determine the temperature profile. This temperature profile has an exact solution $\mathbf{T(t)}$ and the numerically calculated solution at the current time step $\mathbf{T_n}$. The error ε is defined using the l_∞ norm or simply the maximum error between the exact and approximate solution:

$$\varepsilon = \max ||\mathbf{T(t)} - \mathbf{T_n}|| \tag{2.82}$$

Theoretically speaking, we want to adjust Δt such that this ε is small or below some user set tolerance ΔT_{tol}. In practice, we cannot obtain $\mathbf{T(t)}$ realistically, hence, we need an estimate for $\mathbf{T(t)}$. A decently conservative estimate perhaps for this ε is that the all changes in temperature between subsequent time steps are due to errors. This would most certainly be the case if the temperature profiles are meant to be constant with time. Hence, we might use temperatures calculated at the previous time step in order to estimate this error [Mohan and Tamma, 1994]:

$$\max ||\mathbf{T}_n - \mathbf{T}_{n-1}|| \leq \Delta T_{tol} \tag{2.83}$$

This of course assumes that we have a history of temperature with which to look up. However, when designing real time simulations, I would desire to free memory as far as possible to keep computational costs low. I could then keep more objects in the very limited cache memory and thus speed up calculations [Alsharef et al., 2021] such that it is useful for digital twin applications. As an alternative, we might use the temperature profile calculated at the next time step in order to estimate the potential error:

$$\max ||\mathbf{T}_{n+1\ estimated} - \mathbf{T}_n|| \leq \Delta T_{tol} \tag{2.84}$$

In turn, ΔT_{tol} can be set such that [Mohan and Tamma, 1994]:

$$\Delta T_{tol} = a|\mathbf{T}_{\max} - \mathbf{T}_{\min}| \tag{2.85}$$

Where $0 < a < 1$ based on user defined settings. a should be selected so that it is not too large such that instabilities occur, nor too small such that there is no marginal benefit

in having reduced the time step [Mohan and Tamma, 1994]. While the nodalised model as a whole may not be discretised into matrix equations as for the FVM, the principle of defining time steps based on temperature changes of an existing body can be taken from this. For a two body system, this condition can be written as:

$$\Delta T_{tol} = a|T_{hot} - T_{cold}| \tag{2.86}$$

Hence, we can scale time step tolerance to be some percentage of the temperature difference between the two bodies, or between the characteristic temperature scale of the system. The net effect is that higher temperature gradients would result in shorter time steps. In essence, we want to limit the change in temperature change during time step advancement to some threshold value.

2.6 Literature Review Summary

From the literature review, we have covered some of the important principles and information for programming the heat transfer libraries for CIET's Digital Twin, or at least the Digital Twin of CIET's Heater. This includes learnings and best practices for writing the Digital Twin, information and correlations for the heater and the piping as well as solver stability considerations. With these in mind, we are now ready to program and construct our heat transfer libraries.

Chapter 3

Thermal Hydraulics Library Construction

In this chapter, we discuss some principles on how to construct a thermal hydraulics library in the Rust prgramming language. Specifically, we focus on how to program the heat transfer aspects of the thermal hydraulics library. Since we have already elaborated on some of the heat transfer equations in the previous chapter, we shall emphasise the programming principles more in this chapter than the heat transfer aspects. We then validate the library against simple test cases and experimental data to check if the library functions correctly and produces the intended results. This is covered when we discuss Test Driven Development of the Thermal Hydraulics Library. Lastly, we also need to ensure that the library is able to execute calculations fast enough and accurate enough so that a real time digital twin of CIET can be constructed. This means that the calculation times for both the fluid mechanics simulation and the heat transfer simulation will need to be faster than real time. Moreover, we need to ensure that our Digital Twin of CIET Heater v2.0 Bare is validated. These will be covered in the last part of this chapter.

3.1 Principles

Development Strategy

For library development, we aim to use the "rapid iteration" or "rapid prototyping" approach used by Kairos Power LLC [Blandford et al., 2020]. This entails the use of several test design cycles in order to arrive at the final product expeditiously. In other words, we wish to develop small, imperfect, modular segments of code and test them rather than plan to develop a finished product from the outset. As such, the library as presented in this paper will cover only early iterations or merely the "alpha" testing of the Rust Thermal Hydraulics library used for the digital twin. This means that the Application Program Interface (API) presented here will cover only early iterations of the "thermal_hydraulics_rs" library.

A second principle is to follow Donald E. Knuth's advice that "premature optimization is the root of all evil" [Knuth, 1974], meaning to say that programs should not be optimised for speed gains too early on lest it makes debugging and maintenance excessively difficult for little gain [Knuth, 1974]. Therefore, using more complex Rust language features such as traits objects, lifetimes and dynamic dispatch should generally be avoided unless absolutely needed.

Simplifications

Given these principles, I would not develop thermal hydraulics capabilities for use cases outside CIET for the purposes of this dissertation. This means that I will limit the capability development of the thermal hydraulics library to single phase incompressible fluid flow as well as heat conduction. For now, we do not deal with multiphase phenomena as this is of less importance than the single phase heat transfer and fluid mechanics phenomena. Given that this is the case, I will often use the Boussinesq approximation. This means that density changes are negligible except in the case of buoyancy and natural convection. Consequently, for control volumes used to represent CIET, mass accumulation within the control volume is assumed to be zero.

User API

Early iterations of development showed that programming interactions between differing control volumes and boundary conditions in a concise and easy to read manner would prove challenging. Therefore, the library should be written with some kind of structure or interface which makes manipulating control volumes and boundary conditions intuitive as possible. Furthermore, advancing time step (Δt) should be done with a simple function call rather than with several procedural steps, which can be quite cumbersome if they were written in an "inline" fashion.

From a user's perspective, it would be more intuitive to conceive of control volumes as objects in object oriented programming (OOP). The user could then use a function to "connect" two control volumes in order to have them interact with each other or with some boundary condition. The underlying library should then take care of the rest. As discussed in the previous chapter, we solve for 1D arrays of control volumes using a semi-implicit time marching scheme. Between the 1D arrays, an explicit time coupling scheme is used. In the limit where the arrays contain only one control volume, then we shall only have explicit coupling between the control volumes. I programmed capability in the library to allow the user to decide if he or she wants to couple 1D arrays control volumes or single control volumes together. The desired user API workflow for this process is shown in Figure 3.1:

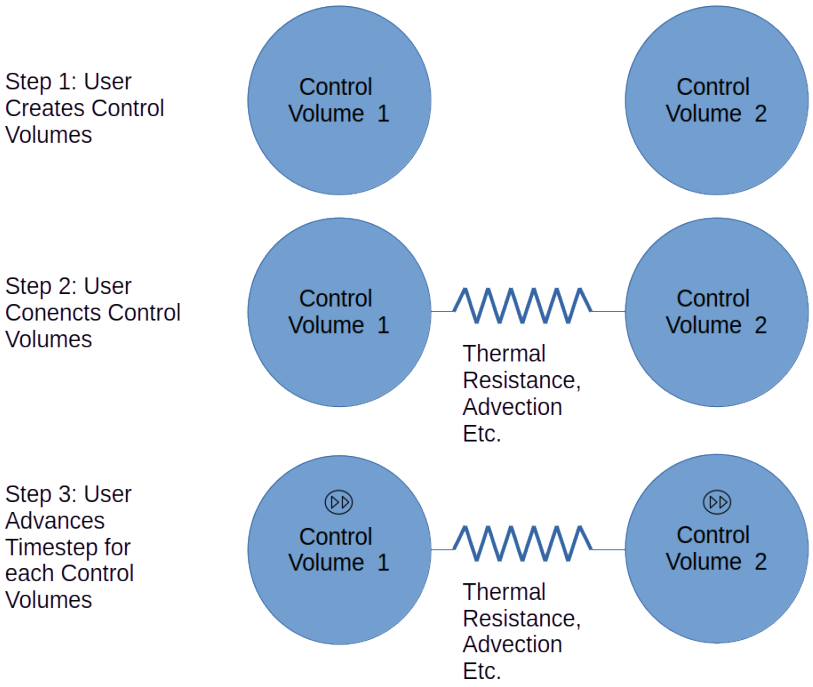


Figure 3.1: User API Progression Overview

This three step approach shown in Figure 3.1 should allow end users enough room for flexibility in how they may want to construct their simulation. At the same time, it reduces cognitive load sufficiently by abstracting away details on how interactions are calculated between control volumes and how the timestep is advanced so to speak. This framework will then guide the construction of the heat transfer portion of the thermal hydraulics library.

Overview for Control Volume and Boundary Condition Interactions

To develop a library which is able to to perform heat transfer calculations, we must first develop program structures that represent control volumes. Moreover, we must also program how control volumes interact with each other and with a variety of boundary conditions. These could be a user specified ambient temperature boundary condition, user specified heat flux, or user specified heat addition.

These control volumes and boundary conditions should be relatively intuitive to use. Moreover, these control volumes should be able to represent heat transfer components. However, representing heat transfer components with control volumes can prove to be quite cumbersome. For example, if one were to consider a pipe with fluid flowing through it, we may want to discretise it into a set number of nodes. For CIET's heater, this number may be as many as 8 to 15 [De Wet and Per F Peterson, 2020; Zou, R. Hu, and Charpentier, 2019]. If we were to consider one set of nodes representing the fluid, another set of nodes

representing the metallic pipe, and yet another set representing the insulation, then we may end up with as many as 24 to 45 nodes which the user needs to construct. Now, an IET such as CIET may contain as many as to 30 of these components. This means we may need to have as many as 500 or more control volumes. Programming these by hand will be quite impractical, especially when the user may want to change the level of discretisation for the simulation. To address this issue, I needed to introduce a second type of programming entity (otherwise known as a "class" or in Rust, "struct") besides singular control volumes. These are programming classes or structs that represent control volumes instantiated in arrays. Of course, control volumes can also be instantiated in more than just a one dimensional array. This, however, is more in the scope of higher fidelity Computational Fluid Dynamics (CFD) simulations which cannot be run in real time. Such capabilities are therefore outside the scope for this library. Therefore we will only consider singular control volumes and one dimensional control volume arrays.

Overall, we have several types of control volumes and several kinds of boundary conditions that should interact with each other so that heat is exchanged as seen in Figure 3.1. To implement this in code, we need to first discuss the working principles behind all these possible interactions.

Heat Transfer Interactions and Entities

In the context of heat transfer, two control volumes interact with each other when they transfer heat. For conduction and solid-fluid interactions, we can model heat transfer between two control volumes by using thermal resistor or thermal conductor between them. The heat flow will then be determined by the temperature difference and the thermal resistance or conductance H:

$$Q_{1\to2} = -H_{12}(T_2 - T_1) \tag{3.1}$$

In this case, the heat flow from control volume 1 (CV_1) to control volume 2 (CV_2), $Q_{1\to2}$ is determined by the temperature of CV_1 (T_1) and the temperature of CV_2 (T_2) as well as the thermal conductance between CV_1 and CV_2 (H_{12}). To calculate $Q_{1\to2}$, we shall need to know T_1 and T_2 beforehand and specify H_{12}. H_{12} would then be determined via the geometry between the control volumes, Nusselt Number correlations or thermophysical properties such as thermal conductivity. This simplifies the programming process since conduction, convection (between solid and fluid) and radiation can all be represented using a thermal resistance mode.

However, one should note that control volumes do not only interact with each other via a thermal conductance or thermal resistance, but also through advection. In the case of advection:

$$Q_{1\to2} = \dot{m}_{1\to2} h_{fluid\ enthalpy}(T_1) \tag{3.2}$$

In advection, the $Q_{1\to2}$ is determined only by T_1 and the specific enthalpy h_{fluid} in (J/kg)[1]. This is, of course, in the case that the mass flowrate from CV_1 to CV_2 $\dot{m}_{1\to2}$ is positive. In the case that $\dot{m}_{1\to2}$ is negative:

$$Q_{1\to2} = \dot{m}_{1\to2} h_{fluid\ enthalpy}(T_2) \tag{3.3}$$

The amount of heat transferred from CV_1 to CV_2 $Q_{1\to2}$ is negative, which means CV_1 gains heat from CV_2. This should make physical sense.

Overall, the control volumes interact with each other mainly through thermal conductance and advection. To keep things organised, I then generalise the way that control volumes interact with each other under a class or object in the library known as a "HeatTransferInteraction".

Of course, control volumes should be able to interact with boundary conditions as well. For example, a heater can be modelled as a constant heat addition boundary condition. From the end user perspective, connecting a boundary condition to a control volume should have a similar feel as connecting two control volumes. Therefore, I would group both boundary conditions and control volumes together in one generic class or object. Hence, both control volumes and boundary conditions are called "HeatTransferEntity" objects.

Timestep Advancement

Now, after the heat flows between "HeatTransferEntity" objects are computed, the temperatures of the heat transfer entities at the next timestep $T^{t+\Delta t}$ need to be calculated. Moreover, once we move on to the next timestep, the calculated temperatures of the next time step should then become the temperature of the current time step. These two processes are critical for transient simulations, and I will call this the "timestep advancement" step of the program. The end user should be able to just connect "HeatTransferEntity" objects via "HeatTransferInteraction" objects, and then call a function to execute code for timestep advancement.

For each control volume:

$$m_{cv} c_{p,cv} (T^{t+\Delta t} - T^t) = Q_{total} \Delta t \tag{3.4}$$

Underneath the hood, an appropriate time step Δt should be determined such that the simulation is accurate such that the thermal hydraulics phenomena of different timescales are captured. Additionally, should control volumes or control volume arrays be explicitly coupled to other control volumes or control volume arrays, then we will have to consider stability when determining the appropriate timestep. Since I chose a semi-implicit method where there are some explicit coupling schemes involved, the maximum allowable system time step would be determined by the parts which are explicitly coupled. While partitioning the system into different systems where separate timesteps are used in each system is possible, I will develop this capability for this dissertation. Such endeavours are left for future work.

[1]Do note that h can be used for both specific enthalpy and heat transfer coefficient, depending on context

In addition to an appropriate timestep, the relevant specific heat capacities and masses of each control volume ($c_{p,cv}$ and m_{cv}) should be obtained so that we can perform timestep advancement. The total heat rate added to the control volume Q_{total} should also be known for this calculation.

Therefore, each control volume should be able to map $c_{p,cv}$ to its temperature and vice versa. Furthermore, it should also have a list or vector of heat added or subtracted from it that is calculated from the various heat transfer interactions. Lastly, Δt should be optionally calculated by the user or calculated algorithmically based on Von Neumann analysis (Mesh Fourier Number or Courant number analysis) or some maximum temperature change.

Of course, $c_{p,cv}$ is not a constant with temperature. While $c_{p,cv}$ can be estimated at each timestep, the temperature at the next time step would then suffer from approximation error or truncation error. One way around this problem is to perform transient calculation in terms of material enthalpy rather than material temperature. This was done frequently in OpenFOAM solvers such as "chtMultiRegionFoam". A similar approach can be applied here:

$$m_{cv}(h^{t+\Delta t}_{enthalpy} - h^{t}_{enthalpy}) = Q_{total}\Delta t \tag{3.5}$$

The temperature of the control volume can be calculated by mapping material specific enthalpy to temperature and vice versa as needed using a thermophysical properties library.

Program Flow

Based on these requirements, one could calculate $Q_{A\to B}$ during a typical heat transfer interaction. Both control volumes A and B would have a vector containing the heat transferred in each interaction. For this interaction, $Q_{A\to B}$ would be added to the heat transfer vector in B and subtracted from the heat transfer vector in A. This process is illustrated in the heat transfer entity connection stage of Figure 3.2:

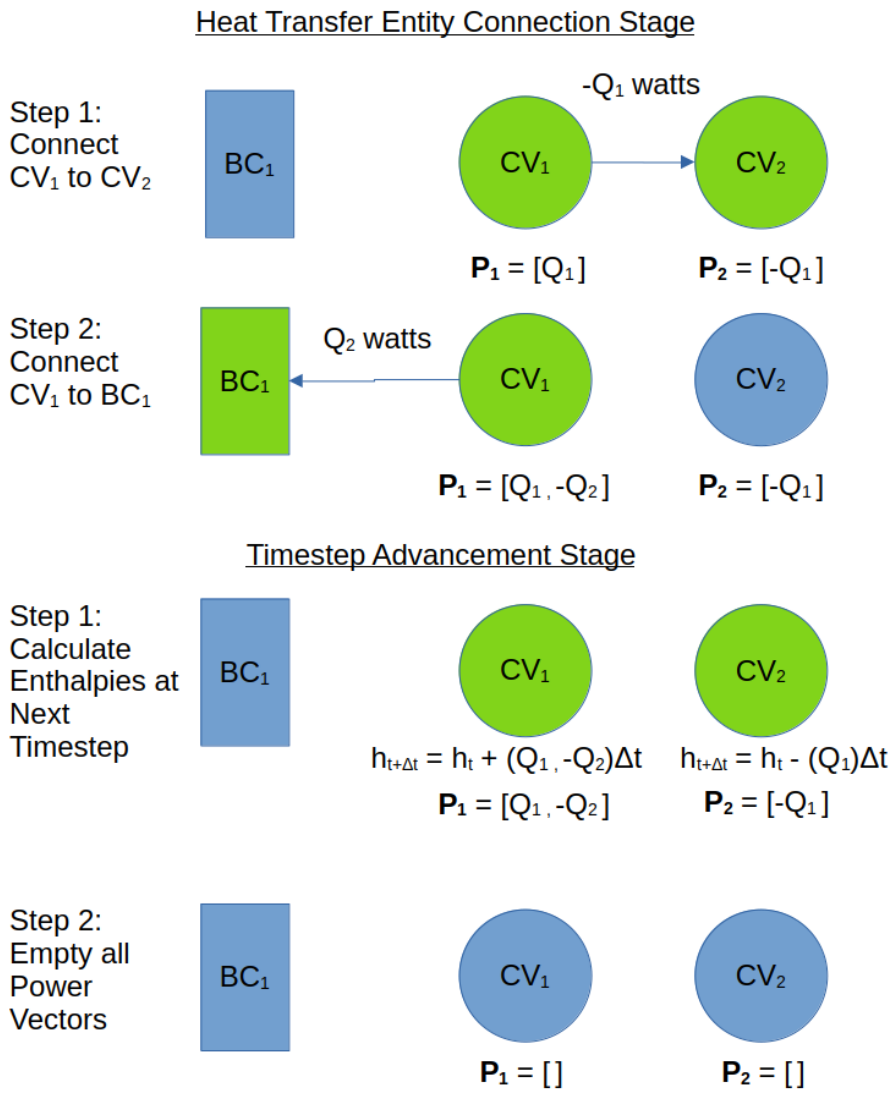


Figure 3.2: Program Flow for Thermal Hydraulics Library

The end user would be able to connect heat transfer entities whether they would be boundary conditions (BCs) or control volumes (CVs). This would happen at during every timestep calculation. These heat transfer calculations would, of course, require temperature and mass flowrate data. Temperature data should be supplied by mapping the control volume's current specific enthalpy h at the present timestep.

Once all heat transfer interactions are completed, the user can specify a time step or have

it be dynamically calculated according to the stability time scales described in the previous chapter, and then he or she would call the advance timestep function in the second part of Figure 3.2. Timestep advancement would simply entail calculating the enthalpy at the new timestep and updating the "HeatTransferEntity" objects to ensure that their respective specific enthalpies correspond to that of the next time step. It would also mean emptying the vector of power supplied to each control volume before starting the next timestep.

3.2 Implementation Methods for Program Flow

Now that we have outlined a rough way the program should work, we now outline the tools to be used to accomplish this mission.

Again, I iterate that this library is meant for building a digital twin of CIET.Therefore, it should be able to perform calculations for CIET in real-time or faster than real-time. Therefore, both accuracy and speed are of the essence. Furthermore, user experience is an important design consideration factor so that it will be somewhat intuitive to use the library.

In this section, we shall discuss how polymorphism and unit safe code is used to improve user experience, and how parallelism and other methods help the code achieve the required speed. I also discuss using linear algebra libraries and n-dimensional arrays to improve both the user experience and calculation speed of the program.

Using Polymorphism for Improved User Experience when Dealing with Control Volumes and Boundary Conditions

For user experience, it is important that the programmer setting up the simulation not need to deal with the total complexity of each of these calculation steps. Therefore, there should be a certain level of abstraction so that the tedious calculation details are mostly hidden from the user. However, the program should be flexible enough to cover all sorts of heat transfer interactions in CIET.

To do this, the thermal resistance method was used to model heat transfer in the digital twin library used for CIET as far as possible. This is because the thermal resistance method is an intuitive way to visualise heat transfer between several components. Having an intuitive user interface is important because we wish to model CIET, which has several interconnected components that can easily overload the user with information. Hence, modelling CIET as a network of control volumes connected with thermal resistors at least for conduction and solid-fluid interfaces would be most suitable. However, not all heat transfer entities transfer heat in the same way. For solid-fluid heat transfer, we mostly use empirical Nusselt number correlations and surface area for calculating thermal conductance, but for solid-solid interactions, the material's thermal conductivity and dimensions are used to determine the thermal conductance. We can see that each type of heat transfer interaction behaves differently depending on the underlying physics and geometry. The programming concept best suited to accommodate these different modes of heat transfer is polymorphism.

For the end user, this means that a "HeatTransferEntity" object could at times behave as a control volume or a boundary condition based on user preference. Likewise a "HeatTransferInteraction" could be based on thermal conductance, advection, or simply facilitate heat flow from a constant power heat source. In computer lingo, this means that the function should be able to accept different types of objects as function inputs [Cardelli and Wegner, 1985]. This implies that the function accepting inputs must be able to deal with the user giving it boundary conditions of various types and control volumes of various types. It must also be able to handle the various types of interaction between these entities to compute a suitable thermal resistance, thermal conductance or heat flow.

There are several ways to introduce polymorphism in Rust. However, I will discuss only two main ways. Here, we shall focus on the use of traits and enums for polymorphism in Rust.

Trait Polymorphism in Rust

For those familiar with object oriented language such as C++, C# and Java, traits are quite similar in concept to "interfaces" and "abstract classes". I have tried this approach for developing the fluid mechanics library in Rust in my master's thesis [Ong, 2023]. In earlier iterations of the fluid mechanics library, which has since been integrated into the "thermal_hydraulics_rs" library, I had wanted to create vectors or lists of fluid components. These components could be pipes, which utilise the churchill friction factor to calculate frictional losses, or custom components which have empirically derived form loss correlations. Regardless of the underlying mechanics, the user need only supply a mass flowrate and the fluid component would return a pressure loss. This can be enforced in the library by making both the pipe and the custom component implement a "FluidComponent" trait. Therefore, a "FluidComponent" object can behave at times as a pipe, or at times as a static mixer or some other component based on user preferences. This was one way of doing polymorphism. However, experience showed that using dynamic dispatch with trait objects in Rust could prove cumbersome when these trait objects nested within other object because there is a need to explicitly specify lifetimes for the Rust compiler. This became cumbersome specifically when "FluidComponent" objects are nested within other structs or objects. Here, one needs to trait object lifetimes for any object containing a "FluidComponent" trait object. Hence, I wanted to move away from using too many nested trait objects for polymorphism.

Enum Polymorphism in Rust

A second approach where one could avoid dealing with explicitly specifying lifetimes is the use of Rust enums for polymorphism. This makes code development fairly easier and more expeditious. Enums are a data type or data structure where a user can make a selection from a list of selectable items otherwise known as "variants". For instance, a boundary condition can be structured as an enum which allows the user to select from a hard coded list of boundary conditions. Using enums is advantageous in Rust because the Rust compiler

forces you to account for how to handle various variants at compile time. This reduces the number of runtime errors one would have to debug.

The disadvantage of using enums is the tendency to introduce boilerplate code. However, while boilerplate code can make the codebase much bigger, it can also make the code more readable because the reader does not have to jump between 20 source files in order to find out how a function works. Moreover, changing behaviour for one enum would not affect the behaviour of other enums as compared to using traits or interfaces. This loose coupling gives enum based polymorphism potentially more flexibility than trait based polymorphism. Hence, I chose to code the heat transfer libraries using enum based polymorphism.

Using Unit Safety for User Experience

Besides polymorphism, calculating quantities using unit safe code such as using the "uom" crate (a Rust package or module) in Rust has proven advantageous for the user experience. If the user wanted to use the thermal hydraulics library, the user would be almost guaranteed to eliminate bugs based on wrong units at compile time if the unit safe code were used. This proved useful when developing and using the fluid mechanics library previously because a whole category of bugs based on wrong units was eliminated due to using this crate. Using the unit safe crate is also useful because it forces the end user to specify units for every quantity at compile time. This made debugging runtime errors easier as the user would not have to consider unit errors. Of course, one would have to get used to performing calculations using this crate rather than floating point numbers. While using unit safe calculations is quite verbose, it proved useful in guaranteeing unit safety. Therefore, I continued using this crate for the heat transfer library to improve user and developer experience.

Using N-Dimensional Arrays and Linear Algebra Libraries for Improved User Experience and Speed

Another feature that helps improve the user experience is the use of n-dimensional arrays and linear algebra libraries. As briefly mentioned before, it would be quite impractical for the user to manually instantiate arrays of single control volumes to represent a discretised heat transfer component. After the user instantiates the control volumes, he or she must then manually code the timestep advancement step for each control volume. This can prove quite cumbersome to code. To get around this problem, we can learn some practices from computational fluid dynamics (CFD) code.

In CFD software such as OpenFOAM, discretisation and creation of the arrays of control volumes is usually automated with the use of meshing software. The creation of matrices to represent the control volumes is automated as well. Once the meshing and discretisation process is complete, finite volume codes such as OpenFOAM and GeN-Foam would then solve the underlying system of matrices using optimised linear algebra solvers. This process makes it practical for the user to simulate complex geometry without getting too involved in instantiating the control volumes one by one.

I take inspiration from the CFD approach in that I would use n-dimensional arrays to represent the heat transfer of control volumes within each component. For this case, $n = 1$ because we are not likely able solve higher order arrays in real-time. For the end user, he or she would only need to specify the number of nodes for a heat transfer component, and the library would take care of the discretisation of the component into several 1D arrays. By doing so, I hide the complexity behind an abstraction. The user can simply construct an array of control volumes or "array_cv" once and never need to look at the underlying matrix construction. Of course, I could have represented CIET as a matrix problem and used linear algebra libraries to solve them. However, experience from reading SpiceSharp libraries written in C# showed that it was quite difficult to read, understand and modify the source code for my own since it was structured as such. Hence, I choose to use matrices and arrays to solve for temperature profiles only within each component rather than the system as a whole. Therefore, an additional form of "HeatTransferEntity" known as array control volumes or ArrayCV objects. ArrayCV objects can be added to represent all such nodalised control volumes which use matrices to represent the energy balance equations. Moreover, the end user may find this easier to use rather than constructing one control volume for each node in each control volume.

ArrayCV objects contain control volumes which are coupled in a semi-implicit manner amongst themselves. However, ArrayCV objects are also meant to be explicitly coupled to other ArrayCV objects and SingelCV objects as I wanted to implement the mesh partitioning scheme as described in the last chapter. To facilitate the explicit coupling, I programmed the ArrayCV such that it contained two SingleCVs, one at each tail end of the array of control volumes. Since I had already developed code for SingleCVs to couple explicitly with each other, I could reuse this code to allow ArrayCV objects to couple explicitly amongst each other and with SingleCV objects. This is shown in Figure 3.3:

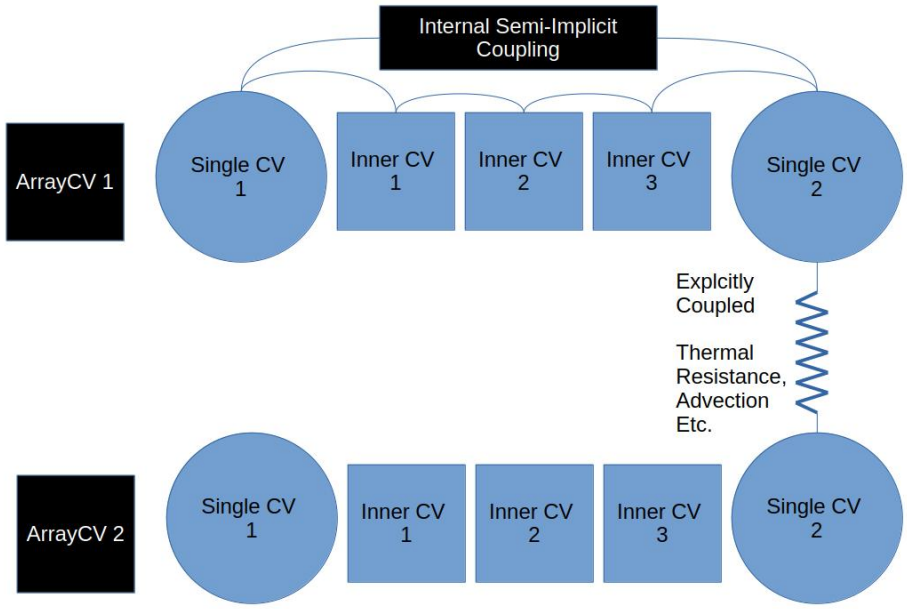


Figure 3.3: Semi Implicit Coupling within ArrayCVs and Explicit Coupling Between ArrayCVs using Embedded SingleCV objects within ArrayCVs

As shown in Figure 3.3, the ArrayCVs themselves contain two SingleCV objects within their each array at both tail ends of the 1D array. These can be coupled axially to other ArrayCVs or other SingleCVs using an explicit coupling scheme. Within each ArrayCV, the control volumes contained within it are coupled in a semi-implicit manner. To solve the array control volume matrices within the ArrayCV, a linear algebra library must be used that hides the complexity of solving the matrices representing the control volume. This allows the user to focus more on constructing the simulation rather than thinking about matrix construction. Additionally, using these linear algebra libraries is also extremely beneficial for real-time simulation because these libraries are optimised for speed. A suitable library for this purpose is the "ndarray-linalg" library written in Rust was used. It uses a linear algebra package (LAPACK) library to perform operations to solve $\mathbf{A}\vec{x} = \vec{b}$. The LAPACK library I used was "intel-mkl" or Intel's Math Kernel Library originally written in C++ and Fortran. I used this particular library because it was cross platform and could work on both Intel (tested on GNU/Linux Operating Systems (OS)) and AMD CPUs (tested on Windows OS). On Linux systems, I chose to program the library in such a way that the OpenBLAS LAPACK library was used instead of "intel-mkl". This is because it is licensed under the MIT license and Apache 2.0 license rather than the Intel Simplified Software License (ISSL).

Parallelism and Other Methods for Speed Increases

A critical mission of the Digital Twin is real-time calculation or faster than real-time calculation. However, initial test results showed that the program operating in serial mode was not going to be fast enough. For the first few trial runs of CIET's heater, test results showed that the average auto calculated time step needed for stability using an explicit coupling scheme was about 0.0023 s or 2.3 ms. This was done using eight fluid nodes axially, one fluid node radially, and two steel nodes radially. The 2.3 ms timestep requirement was mainly due to the steel nodes having very little thermal inertia compared to the thermal diffusivity, and due to the explicit coupling scheme axially and radially in initial runs. Of these two directions, the conduction in the radial direction necessitated the smallest timescales. Hence, time steps for these initial runs with explicit schemes needed to be kept small to ensure that the Fourier number was less than 0.25. For calculation at this interval, the heater needed roughly 600 ms of calculation time in debug compilation mode. This was about 260 times slower than real time calculations. This showed the need to speed up calculations so that they can happen faster than the required time step. Code optimisation was done in an iterative process. Usually, it was done only after an unoptimised version of the code is able to reasonably replicate experimental results.

To optimise the code, the most obvious way was to compile the library in "release" mode rather than "debug" mode. In this manner, the compiler performs optimisations on the eventual binary file so that the CPU can execute the instructions faster as compared to debug mode. The only drawback is that "release" mode compilation usually takes longer than "debug" mode. In this scenario, however, the tradeoffs were acceptable. A good rule of thumb is that Rust code compiled in "release" mode performs about 10 times faster that code compiled in "debug" mode.

The second method to speed up code was to understand which sections of the code were likely to be the bottlenecks. To do so, I used libraries, including Rust's SystemTime functions, to check the time taken and percentage of time required for four sections of code. Additionally, I have had friends recommend me code profiling tools such as "Callgrind" and "KCacheGrind". Such tools have been used in literature for sequential code performance analysis [Weidendorfer, 2008]. For this project, flamegraphs were used to help profile and optimise the code. Use of such tools was already established in literature [Lunnikivi, Jylkkä, and Hämäläinen, 2020]. This supports my decision in using these tools.

3.3 Test Driven Library Development

Now that we have outlined some principles and methods of how to construct the thermal hydraulics library, the next step would be to develop the library and validate it using simple validation cases especially for the SingleCV object. Only when the simple cases are done, then we move on to a more complex test case.

In this section, we use the lumped capacitance and semi infinite medium conduction test

cases to verify if the SingleCVs were developed for conduction cases and simple conjugate heat transfer cases. This is because the analytical solutions are relatively well known. We can therefore compare the analytical solution to the simulated values. We then explore a simplified one node model of the heater similar to CIET's heater and use its transfer function as a basis of comparison. This would help verify that the advection mode of heat transfer works correctly. Lastly, we attempt to iteratively construct a model based on ArrayCVs to model CIET's heater as described in the previous chapter.

Lumped Capacitance Model

In early stages of code development, I needed a simple test to verify if the API works correctly. This first test I used was the most simple for transient conduction: lumped heat capacitance. For lumped heat capactiance, we simulate a steel sphere cooling in air.

Geometry, Boundary Conditions and Initial Conditions

For this test case, the sphere is 2.0 cm in diameter, and made of 304L stainless steel. The steel starts at an initial temperature of 150 $°C$ and is cooled with ambient air at 25 $°C$. The heat transfer coefficient is 20 $W/(m\ K)$. We can compute an analytical solution given some approximations and compare that to the simulated values in the thermal_hydraulics_rs library.

Analytical Solution

Let's first state the lumped capacitance energy balance:

$$\rho V c_p \frac{\partial T_{ball}}{\partial t} = -hA_s(T_{ball} - T_{ambient}) \tag{3.6}$$

Where ρV is the mass of the steel ball (density ρ times volume V), T_{ball} is the temperature of the steel ball, t is time, h is the heat transfer coefficient, $T_{ambient}$ is the ambient temperature of air. For the sake of simplicity, we use a representative average specific c_p for this calculation. With this simplification, we can easily reduce the partial derivative to a total derivative.

$$\frac{d(T_{ball} - T_{ambient})}{dt} = -\frac{hA_s}{\rho V c_p}(T_{ball} - T_{ambient}) \tag{3.7}$$

This is a simple initial value problem. It is customary to nondimensionalise the coefficients using the Biot number Bi and Fourier Number Fo. Bi is defined in literature as [Xu, P.-W. Li, and C. L. Chan, 2012]:

$$Bi = \frac{hL_c}{k_{solid}} \tag{3.8}$$

Here k_{solid} is solid thermal conductivity usually in $W/(m\ K)$ and L_c is some characteristic system length. A suitable candidate here for L_c is:

$$L_c = \frac{V}{A_s} \tag{3.9}$$

Fo is defined and used in literature as [Hensen and Nakhi, 1994]:

$$Fo = \frac{\alpha t}{L_c^2} \tag{3.10}$$

Where α is thermal diffusivity usually in $m^2 s^{-1}$:

$$\alpha = \frac{k}{\rho c_p} \tag{3.11}$$

We can nondimensionalise equation 3.7 as:

$$\frac{d(T_{ball} - T_{ambient})}{dt} = -Bi\frac{\alpha}{L_c^2}(T_{ball} - T_{ambient}) \tag{3.12}$$

After integration, we can obtain an expression:

$$\frac{T_{ball}(t) - T_{ambient}}{T_{ball}(t=0) - T_{ambient}} = \exp(-BiFo) \tag{3.13}$$

It is customary to represent nondimensional temperature as $\theta(t)$:

$$\theta(t) = \frac{T_{ball}(t) - T_{ambient}}{T_{ball}(t=0) - T_{ambient}} \tag{3.14}$$

So that:

$$\theta(t) = \exp(-BiFo) \tag{3.15}$$

Let us calculate L_c for Bi:

$$L_c = \frac{4\pi r^3}{3 * 4\pi r^2} = \frac{r}{3} = \frac{D}{6} \tag{3.16}$$

Where r is ball radius and D is ball diameter. For 2 cm ball diameter, $L_c = 0.3333cm$.

Now, we need thermal conductivity and diffusivity is to be able to calculate Bi and Fo. For this, I'm going to take a rough representative average value of thermal conductivity and diffusivity at around 355 K or $82°C$. At $82°C$, the measured value of α is 0.0390 $cm^2 s^{-1} \pm 2\%$ at 355.2K measured using laser flash apparatus from Springfields [Graves et al., 1991] and k is 15.27 $W/(m\ K) \pm 1.5\%$ measured at 354.9K using high temperature longitudinal (HTL) apparatus Oak Ridge National Laboratory (ORNL) [Graves et al., 1991].

At these values, Bi can be calculated as:

$$Bi = \frac{20W/(m^2\ K) * 0.3333cm}{15.27W/(m\ K) * 100\ cm/m} = 0.004366 \pm 1.5\% \tag{3.17}$$

As $Bi < 0.1$ we can use lumped heat capacitance [Hensen and Nakhi, 1994].
Hence, the analytical solution can be described as:

$$\theta(t) = \exp(-0.004366\ Fo) \tag{3.18}$$

Uncertainty Quantification

To determine if the simulated values are close enough to the analytical solution, we can add error bars for $\theta(t)$. Uncertainties independent of each other can be calculated as [V. R. Meyer, 2007; Todreas, Kazimi, and Massoud, 2021; BIPM et al., 2008]:

$$u_c^2(M) = \sum_i \left(\frac{\partial M}{\partial x_i}\right)^2 \sigma^2(x_i) \tag{3.19}$$

This equation assumes the uncertainties are uncorrelated. Now, since two different measurement instruments from two different laboratories were used, I can assume the uncertainty in these instruments would be uncorrelated. Of course, I could have obtained k_{solid} measurements from α but then we would have to take into account uncertainties in c_p and ρ. For simplicity, we shall just assume these are uncorrelated, that Bi has an uncertainty of $\pm 1.5\%$ and that Fo has an uncertainty of $\pm 2.0\%$ which comes from uncertainty in α. Also, while we could consider uncertainty that may come from a Nusselt correlation in obtaining h and measurement for L_c, these will be ignored. L_c measurement uncertainty is at most $\pm 0.1mm$ if we consider using Vernier Callipers [Çelebioğlu, 2005]. For our case, this only represents a $\pm 0.5\%$ error.

$$u_c^2(\theta(t)) = \left(\frac{\partial \exp(-Bi\ Fo)}{\partial Fo}\right)^2 \sigma^2(Fo) + \left(\frac{\partial \exp(-Bi\ Fo)}{\partial Bi}\right)^2 \sigma^2(Bi) \tag{3.20}$$

$$u_c^2(\theta(t)) = (-Bi)^2 \exp(-2\ Bi\ Fo)\sigma^2(Fo) + (-Fo)^2 \exp(-2\ Bi\ Fo)\sigma^2(Bi) \tag{3.21}$$

Let's substitute the Fo uncertainty of $\pm 2\%$ and Bi uncertainty of $\pm 1.5\%$:

$$u_c^2(\theta(t)) = 0.020^2 Fo^2(-Bi)^2 \exp(-2\ Bi\ Fo) + 0.015^2(Bi)^2(-Fo)^2 \exp(-2\ Bi\ Fo) \tag{3.22}$$

$$u_c(\theta(t)) = 0.025\ Bi\ Fo \exp(-Bi\ Fo) \tag{3.23}$$

$$u_c(\theta(t)) = 0.025\ Bi\ Fo\ \theta(t) \tag{3.24}$$

In terms of fractional or percentage error:

$$\frac{u_c(\theta(t))}{\theta(t)} = 0.025Bi\ Fo \tag{3.25}$$

Hence, the error bars shall be quantified using $0.025Bi\ Fo\ \theta(t)$.

Results

Now that we have settled the analytical solution with its associated uncertainty, we now discuss the simulation. This was done with timestep of 20s. The simulation was carried out until I could be sure that the temperature of the steel ball was more or less in equilibrium with the air around it. The results were then nondimensionalised and plotted alongside the analytical solution in Figure 3.4:

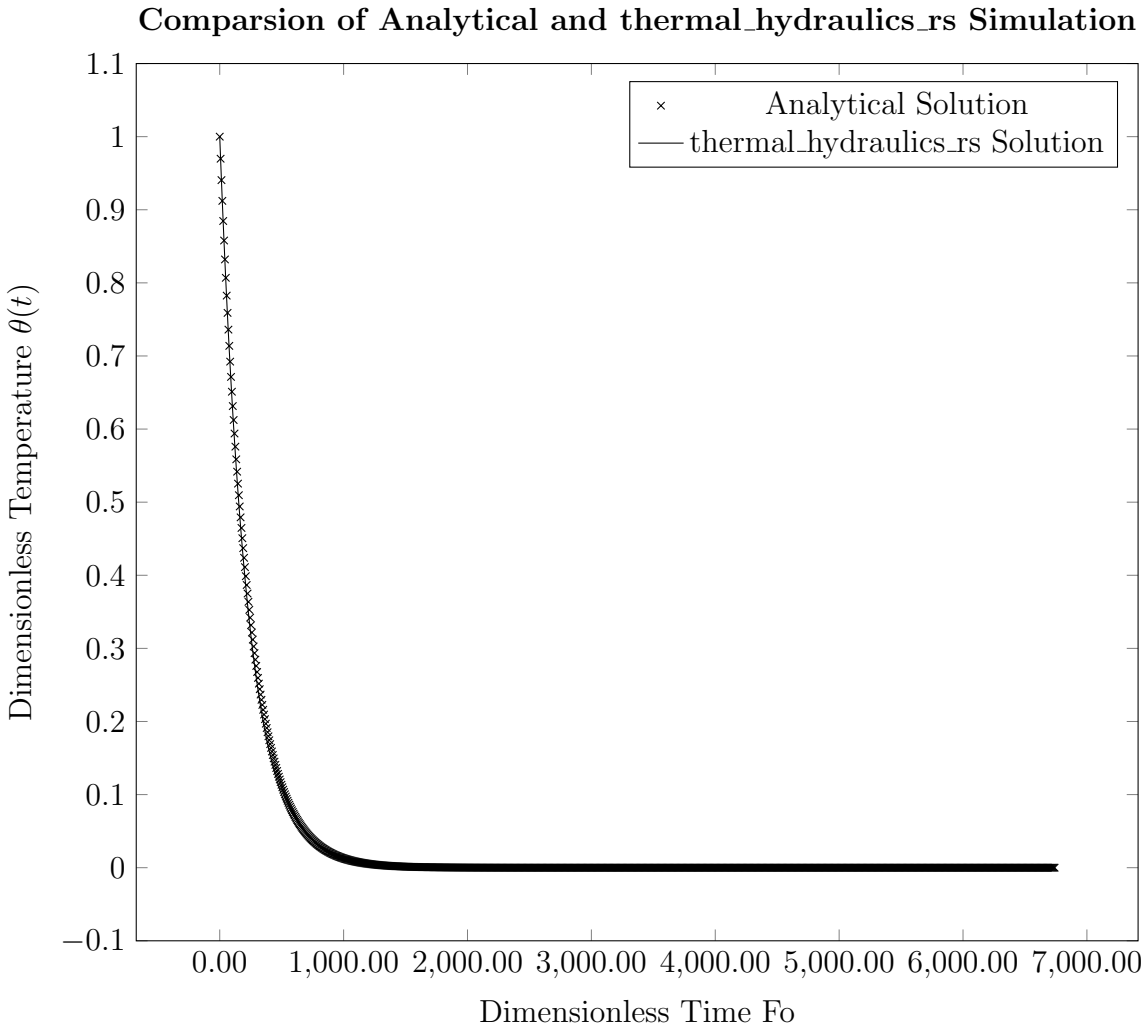


Figure 3.4: Comparison of Dimensionless Temperature vs Dimensionless Time for a Steel Sphere Cooling in Ambient Air

Now, Figure 3.4 shows that the simulation and analytical solution results were quite well matched. To see if they were indeed well matched, however, we need to introduce error bars. These error bars were based on Equation 3.25. The resultant plot is presented in Figure 3.5:

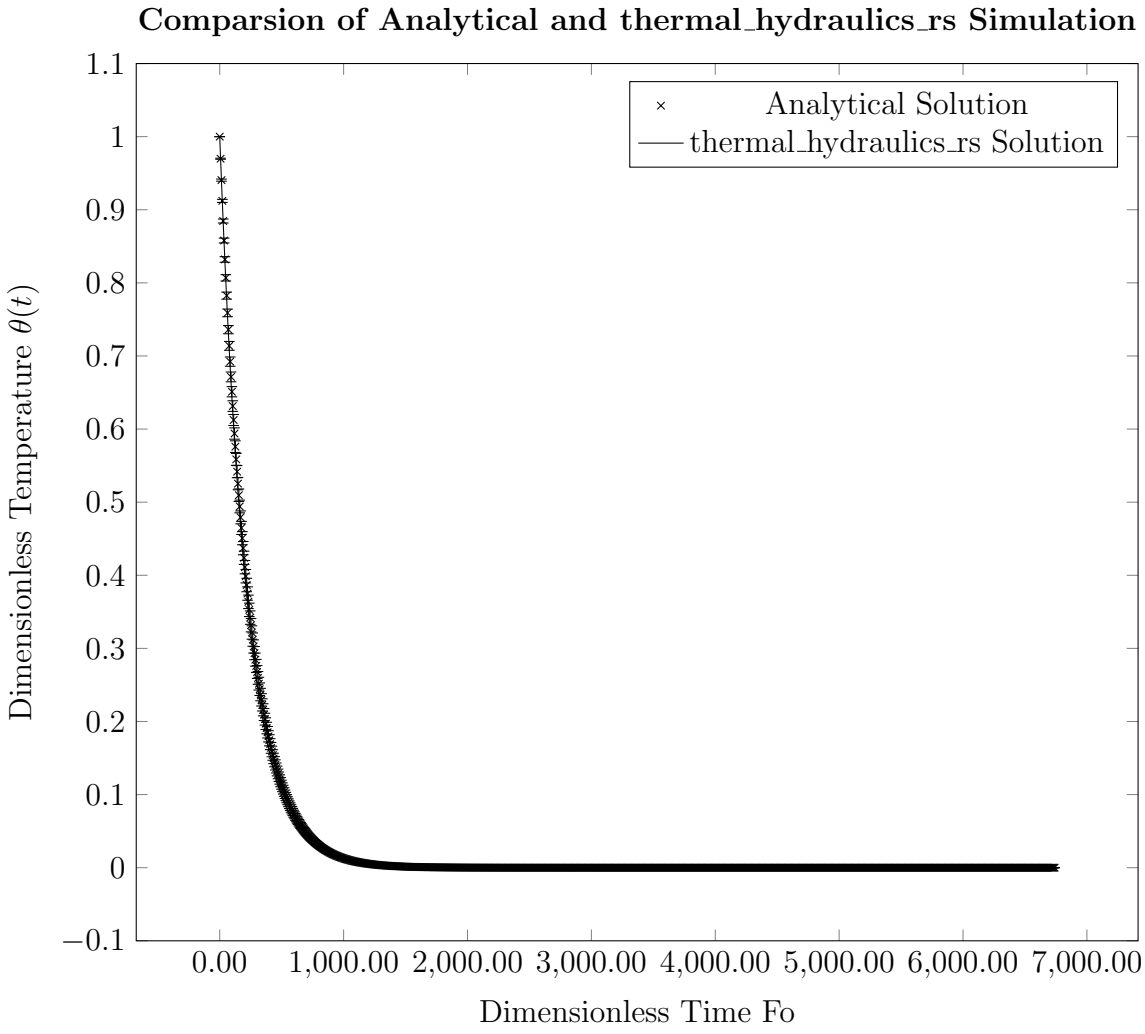


Figure 3.5: Comparison of Dimensionless Temperature vs Dimensionless Time for a Steel Sphere Cooling in Ambient Air with Measurement Uncertainty Error Bars

Of course Figure 3.5 has error bars which are not easy to see given how small the error is relative to the crosses in the figure. Hence, we shall look at residual plots to see how the residuals $\theta(t)_{rust\_library} - \theta(t)_{analytical}$ compare to the measurement uncertainty error bars in Figure 3.6:

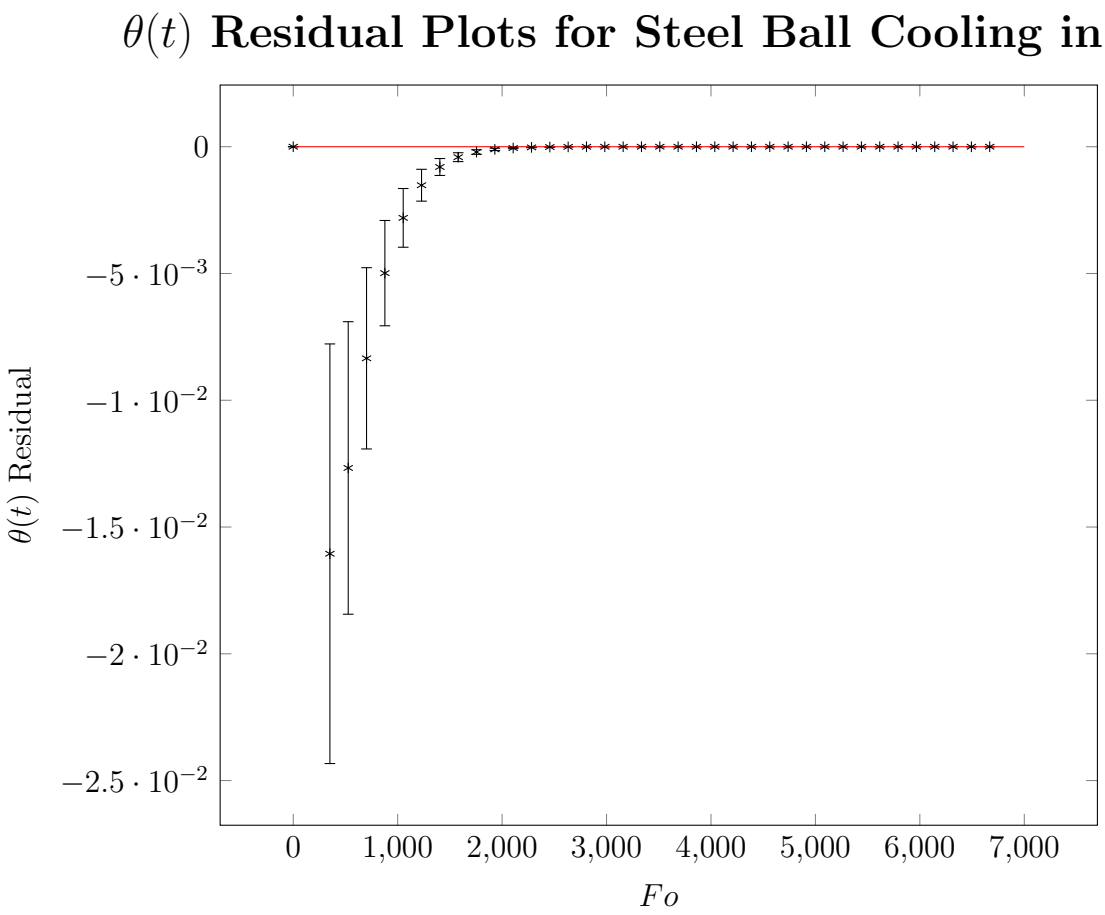


Figure 3.6: Residual Plot for $\theta(t)$ Lumped Capacitance Solution obtained using thermal_hydraulics_rs Library compared to Analytical Solution

Figure 3.6 shows that at some values of Fo, the residuals exceed the measurement error of α and k_{solid}. This is likely because the analytical solution fails to take into account the variability of the material k_{solid} and α. α for example at 355K (82 $^{\circ}C$) is about $0.0390 cm^2 s^{-1}$. But at 423.2K (150$^{\circ}C$), α is about $0.0416 cm^2 s^{-1}$ [Graves et al., 1991]. This is an extra 6.7% variation in α we need to account for. Likewise k_{solid} for steel is $15.27 W/(m\ K)$ at 354.9K and $16.63 W/(m\ K)$ at 423.1 K. This is an additional variability of about 8.9% in thermal conductivity between these two temperatures. To account for this, I opt to simply consider this variation as if it were another random error and use equation 3.19 since we know that variation in thermal properties happens independently of typical measurement errors. Of course, this is a rather simple way, and the more rigourous method is redoing the analytical solution to account for the temperature dependent thermophysical properties. However, I did not see this as strictly necessary because the largest absolute errors are on the order of $2^{\circ}C$. It is enough to show that the library is working. I want to show that a likely cause of this error is the variability in thermophysical properties of steel. For this, such rigourous analysis

is not strictly required. With this in mind, let us go back to our uncertainty propragation equations:

$$u_c^2(\alpha) = \sum_i \left(\frac{\partial \alpha}{\partial x_i}\right)^2 \sigma^2(x_i) \tag{3.26}$$

We consider two terms here, the temperature variation and measurement error:

$$u_c^2(\alpha) = \left(\frac{\partial \alpha}{\partial T}\right)^2 \sigma^2(T) + \left(\frac{\partial \alpha}{\partial \alpha}\right)^2 \sigma^2(\alpha) \tag{3.27}$$

$$u_c^2(\alpha) = 0.067^2\alpha^2 + 0.02^2\alpha^2 \tag{3.28}$$

$$\frac{u_c^2(\alpha)}{\alpha^2} = 0.067^2 + 0.02^2 \tag{3.29}$$

$$\frac{u_c(\alpha)}{\alpha} = 0.070 \tag{3.30}$$

In a similar fashion, the temperature variation and measurement uncertainty in thermal conductivity for steel can be estimated as:

$$\frac{u_c(k_{solid})}{k_{solid}} = \sqrt{0.089^2 + 0.015^2} \tag{3.31}$$

$$\frac{u_c(k_{solid})}{k_{solid}} = 0.090 \tag{3.32}$$

As before, percentage uncertainty of Fo and Bi depend on percentage uncertainty of α and k_{solid} respectively. Let's substitute the Fo uncertainty of $\pm 7.0\%$ and Bi uncertainty of $\pm 9.0\%$:

$$u_c^2(\theta(t)) = 0.070^2 Fo^2(-Bi)^2 \exp(-2\ Bi\ Fo) + 0.090^2(Bi)^2(-Fo)^2 \exp(-2\ Bi\ Fo) \tag{3.33}$$

$$u_c^2(\theta(t)) = 0.1141^2 Fo^2(-Bi)^2 \exp(-2\ Bi\ Fo) \tag{3.34}$$

Thus, the measurement uncertainty with temperature variations can be shown as:

$$u_c(\theta(t)) = 0.1141 Fo(-Bi) \exp(-2\ Bi\ Fo) \tag{3.35}$$

This essentially gives us a larger error bar on our plots. The plots with these new error bars is shown in Figure 3.7:

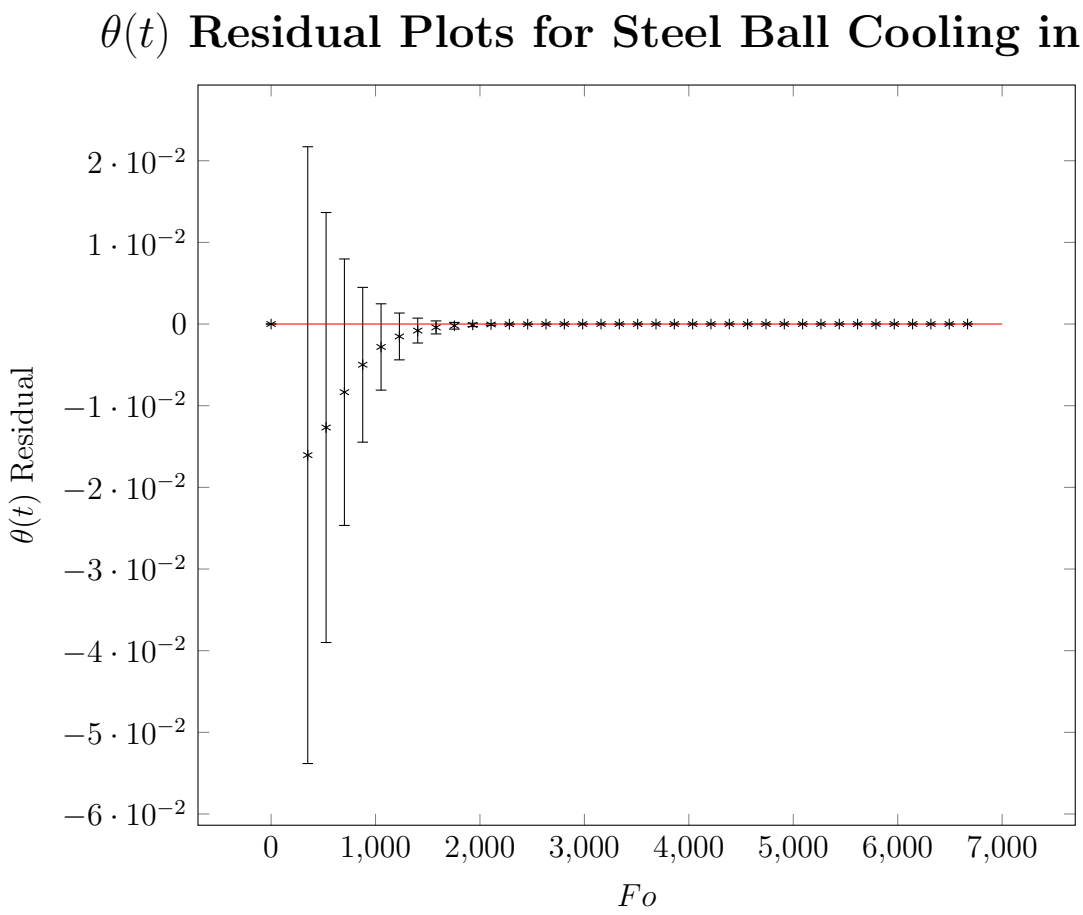


Figure 3.7: Residual Plot for $\theta(t)$ Lumped Capacitance Solution obtained using thermal_hydraulics_rs Library compared to Analytical Solution Accounting for Temperature Varying α and k_{solid} for Steel

In Figure 3.7, we observe much larger error bars when we account for temperature variance of α and k_{solid}. This shows that the simulated $\theta(t)$ using the thermal_hydraulics_rs library is well matched with the analytical solution taking measurement error and temperature variations in α and k_{solid} into account. Of course, we could have used more complex expressions for the analytical solution using polynomial expansions for α and k_{solid} for a more thorough comparison. However, these analytical expressions may be quite complex and impractical to derive analytically.

However, for our intents and purposes, we have verified that the library functions correctly and the code is working properly within a reasonable degree of error.

Learning Points for Temperature Variation of Thermophysical Properties

From this test, I observed that temperature variance of α and k_{solid} plays a large role in simulation accuracy and accuracy even of the analytical solution. This could potentially become

a problem when the temperature varies too much in one Δt such that the thermal resistances vary significantly over the Δt, thus causing calculation inaccuracy. The inaccuracy arises because in calculation of the heat fluxes between control volumes and boundary conditions, the thermal resistance is assumed constant or to vary little within that Δt. When this is assumption is untrue, significant inaccuracies would arise. To mitigate this inaccuracy, the Δt should be picked such that the temperature changes within each Δt are limited to a small temperature change. Now, limiting temperature variations within each Δt would become an additional Δt constraint in addition to the already existing Courant Number. Given the extra complexities of determining a suitable timestep for explicitly coupled control volumes or even the semi implicitly coupled control volumes, I decided to write some code which auto calculates timesteps based on Courant number, a maximum temperature change along with some other factors such as the mesh Fourier number. The mesh Fourier number is of particular interest in conduction, and that's what we shall cover next.

Conduction in Semi Infinite Medium

Analytical Solution

Now, since we have verified the ability of the solver to handle the simplest heat transfer case, which is lumped capacitance, we shall develop the solver to solve more complex cases. For explicitly coupled schemes, we consider conduction in a semi infinite medium. We choose this because the analytical solution is well known in literature [Trojan, 2014]:

$$\theta(x,t) = \operatorname{erfc}\left(\frac{x}{2\sqrt{\alpha t}}\right) \tag{3.36}$$

$$\theta(x,t) = \frac{T(x,t) - T_i}{T_{surface} - T_i} \tag{3.37}$$

$T(x,t)$ is the temperature in the semi infinite medium at position x and time t. T_i is the initial temperature, and $T_{surface}$ is the surface temperature at the boundary. The reader should note that this analytical solution is specific to a constant surface temperature boundary condition. There are other analytical solutions available for constant heat flux as well, but we shall consider only the constant temperature case to validate the thermal hydraulics solvers.

Time Step Considerations

One critical component for solving transient conduction is ensuring that the solver is numerically stable. For explicitly or semi-implicitly coupled schemes, the right timestep must be selected. Now, we could apply an implicitly coupled solution method, but we would still need to consider timescales relevant to simulate the physical phenomena of the system. These timescales are tied more to the length scales of the system rather than the length scales of the mesh. However, the mesh length scale is usually smaller or equal to the system length scale.

Hence, the timescales calculated using the mesh length scale would usually be a conservative estimate of the required system timescales. In the case where we have a ultra coarse mesh such that the mesh length scales and system length scales are similar, then the algorithms for calculating system timescales for physical phenomena and mesh stability timescales become quite interchangeable. As discussed in the literature review section, I used a semi-implicit coupling method for this dissertation. Since the algorithm is not fully implicit, methods timescales relevant to explicit coupling will be relevant for discussion here. For time stepping, the stability is highly dependent on a mesh Fourier number or discretisation Fourier number [Thomas, Samarasekera, and Brimacombe, 1984]. For such simulations, the mesh Fourier number must be lower than a set value in order for the simulation to be stable . The stability criteria for explicit time stepping [Thomas, Samarasekera, and Brimacombe, 1984] is shown in Equation 3.38:

$$Fo_{discretisation} = \frac{\alpha \Delta t}{\Delta x^2} \leq 0.25 \tag{3.38}$$

For this criterion to be fulfilled, the product of $\alpha \Delta t$ cannot exceed a certain value. Therefore, materials with higher α tend to require a smaller Δt to be stable. The material with one of the highest α values commonly used in heat transfer applications is copper. Hence, we shall restrict ourselves to using a copper medium for this case rather than steel as it has a higher thermal diffusivity. Firstly, I wanted to test and ensure that different material properties work within the thermal_hydraulics_rs library. Secondly, the solution stability is tied closely to the Fourier number, and lower Fourier numbers usually imply better numerical stability. Since this is the case, a higher thermal diffusivity means that the solution is more likely to become unstable compared to other materials of similar grid size and time step. Hence, if the solver can obtain a stable transient conduction solution for copper, it is likely to supply a stable solution for other materials as well.

To illustrate the point, we present data from Parker's original paper on using the flash method to determine thermal diffusivity [Parker et al., 1961] in Table 3.1:

Material	α $(cm^2 s^{-1})$ $22°C$	α $(cm^2 s^{-1})$ $135°C$
Aluminium	0.94	0.89
Copper Allow OFHC	1.15	1.04
Silver	1.61	not available
Steel Alloy 4340	0.096	0.09
Steel Alloy 1020	0.14	0.13

Table 3.1: α for various materials [Parker et al., 1961]

The precision of the information in Table 3.1 was reported to vary by about $\pm 5\%$ and the variance in measured data compared to other sources was found to be within $\pm 10\%$ of

the nominal values [Parker et al., 1961]. Of course, the instruments have gotten better over time as α for steel was measured with a measurement uncertainty of $\pm 2\%$ [Graves et al., 1991]. However, we shall make do for now with at least the $\pm 5\%$ precision error. Regardless, Table 3.1 shows that copper has the highest α of the common construction materials. Silver has a higher α but was not used for CIET and is not likely to be used as a construction material for IETs or reactor applications. Hence, we can consider copper to be suitable for a case with the highest α. Again, a high α means that Fo will be higher, all else equal.

Now, we know that if a mesh is evenly sized, then the entire mesh is described by only one $Fo_{discretisation}$. However, we do not take it for granted that the mesh is always evenly sized especially across various components in heat transfer systems such as CIET.It is likely that each local mesh would have its own local $Fo_{discretisation}$ criteria. In this case, whenever we join control volumes and nodes together with thermal resistances, one appropriate method to time step is to load each control volume node with a vector of maximum allowable time step given each interaction. The maximum allowable time step for this node is the minimum of this vector.

We should likewise calculate a maximum time step based on changing material properties and add it to the vector as well. We then take the global minimum Δt by repeating this over all control volume nodes to determine the best Δt for the current time step. I coded an algorithm that calculates the timestep based on the $Fo_{discretisation}$. The test code would then automatically determine the timestep used for the simulation. If the simluation is stable, it would mean that the auto timestepping algorithm works and that the algorithm is successfully programmed.

Methods

Geometry, Boundary Conditions and Initial Conditions The simulation of a semi infinite copper medium was compared to the analytical solution. In this setup, a constant wall temperature boundary condition of $80°C$ was placed against a semi infinite medium initially at $21.67°C$. The medium was 20 cm long and the simulation was done for 20 seconds. This was to ensure that $\frac{x}{2\sqrt{\alpha t}} \geq 2.0$ so that $\text{erfc}\left(\frac{x}{2\sqrt{\alpha t}}\right) \leq 0.005$ at the 20 cm boundary. The analytical solution was first generated using an average α to generate the temperature profiles. This average temperature was $45°C$.

For the numerical solution, 3D control volumes were used, but since the simulation is essentially 1D, a basis cross sectional area of $1m^2$ was used across all control volumes and boundary conditions. Radially, an adiabatic boundary condition was surrounded all other sides besides the constant temperature boundary condition.

Thermophysical Properties dditionally, the material properties for copper were retrieved from previous work in CIET's RELAP and SAM model [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015] as shown in Table 3.2:

Temperature (K)	k ($W\ m^{-1}\ K^{-1}$)	c_p ($J\ kg^{-1}\ K^{-1}$)
250	406	373.6018
300	401	384.7875
350	396	392.6174
400	393	398.2103
500	386	407.1588
1000	352	417.226

Table 3.2: Thermophysical Properties of Copper [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015]

To interpolate the material properties, the cubic spline function from the Peroxide crate in Rust was used. Whenever the material property was desired, a cubic spline object was constructed using all the available data points, the desired temperature for calculating the material properties was given, and then the material property was retrieved. At the end of the function, the spline object went out of scope. This was computationally inefficient, but a high degree of optimisation was not required at the time of testing. Therefore, it was left unoptimised. An example of this code is presented here:

```
fn copper_specific_heat_capacity(
temperature: ThermodynamicTemperature) -> SpecificHeatCapacity {

let temperature_value_kelvin: f64 = temperature.get::<kelvin>();
// here we use a cubic spline to interpolate the values
// it's a little calculation heavy
//
// and actually, rebuilding the spline is quite problematic
// we need to build it ONCE and read from it
//
let thermal_cond_temperature_values_kelvin = c!(200.0,
    250.0, 300.0, 350.0,
    400.0, 500.0, 1000.0);
let specific_heat_capacity_values_joule_per_kilogram_kelvin
= c!(355.7047,
    373.6018, 384.7875, 392.6174,
    398.2103, 407.1588, 417.2260);

let s = CubicSpline::from_nodes(&thermal_cond_temperature_values_kelvin,
    &specific_heat_capacity_values_joule_per_kilogram_kelvin);
```

```
    let copper_specific_heat_capacity_value = s.
        eval(temperature_value_kelvin);

    SpecificHeatCapacity::new::<joule_per_kilogram_kelvin>(
        copper_specific_heat_capacity_value)

}
```

For density of steel, it was assumed constant at 8940 $kg\ m^{-3}$ [Zou, R. Hu, and Charpentier, 2019].

After a mesh refinement study, a suitable mesh size was decided to be such that the distance between each node was 2cm. However, the distance between the first node and the wall was 1cm. The same applied for the 9th and 10th node, these were the last two nodes.

Uncertainty Quantification I simply used error bars of $\pm$ 0.5 K because these are characteristic of Type T thermocouples used in CIET [Zweibaum, J E Bickel, et al., 2015].

Results

The results of this test are presented in Figure 3.8:

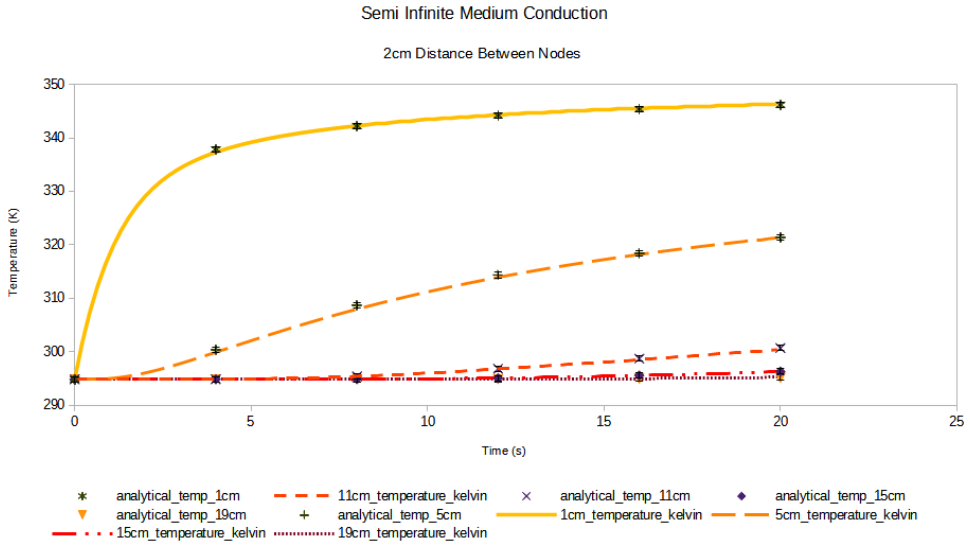


Figure 3.8: Semi Infinite Medium Copper Conduction, Temperature Evolution over Time for Fixed Positions within the Mesh

To evaluate the ability of the library to simulate transient conduction, I observe the temperature evolution over time for five points in Figure 3.8. These are at 1cm, 5cm, 11cm,

15cm and 19cm away from the constant temperature boundary. Additionally, error bars of $\pm$ 0.5°C are shown to give the reader a rough idea of the magnitude of the residual in comparison to a typical thermocouple uncertainty. Figure 3.8 shows relatively reasonable agreement between the analytical solution and the simulated values. The deviation is similar in the order of magnitude to a typical thermocouple uncertainty. Though some points lie outside this range, I conjectured that these were again due to variability in material properties at different temperatures as this seemed to be the case in the example of the steel ball cooling in air. I also wanted to spend my limited time conducting other tests for the library. Due to these two reasons, I decided not to plot the residuals and move on to other cases. For my intents and purposes, the thermal hydraulics library was able to reasonably predict temperature profiles for transient conduction with a reasonably acceptable deviation from the analytical solution.

One Dimensional Model of a Heated Pipe based on CIET's Heater

Now that we have developed code and verified it using simple cases for transient conduction and conjugate heat transfer, let us next test code meant for transient conjugate heat transfer (CHT) with advection. For this, I shall use existing models and data in literature relevant to CIET's Heater.

The simplest of the models based on CIET's Heater was developed by Poresky [Poresky, 2017]. As discussed in the previous chapter, CIET's heater was heated electrically via copper cables. The steel piping receiving current would have been heated volumetrically to temperatures of about 165°C to 185°C. As a result, the Therminol VP-1 or Dowtherm A flowing through the heater would have been raised from about 80 °C to 110 °C. A close up of CIET's Heater, especially its top head, is presented in Figure 3.9:

Figure 3.9: CIET's Heater with Emphasis on Top Head, brown material on heated section is Kapton tape to allow Infra Red Imaging

Figure 3.9 shows CIET's Heater as it stands today. It does not have any insulation as the insulation sustained damage previously [De Wet and Per F Peterson, 2020]. However, it used to have fiberglass insulation on it. Also, its insert used to be an annular tube [Poresky, 2017; Lukas, Kendrick, and P. Peterson, 2017] rather than a twisted tape.

As mentioned in the previous chapter, I used the term "CIET Heater v1.0" to denote CIET's heater in its original configuration, with fibreglass insulation and an annular tube. This naming convention is essentially the same as what De Wet has used before [De Wet and Per F Peterson, 2020]. CIET Heater v1.0 was also the version present in the RELAP model [Nicolas Zweibaum, 2015]. When the insert was changed for improved heat transfer performance [Lukas, Kendrick, and P. Peterson, 2017], the heater was called "CIET Heater v2.0" [De Wet and Per F Peterson, 2020]. When the heater insulation was damaged and the insulation removed [De Wet and Per F Peterson, 2020], CIET's Heater experienced greater parasitic heat losses. This version I will call "CIET Heater v2.0 bare" to denote that the heated pipe is without insulation and therefore bare.

Methods

Poresky's simplified model was based on CIET's Heater v1.0 with fiberglass insulation. While there was an inner pipe within CIET heater v1.0, Poresky's simplified model of the heater essentially ignored much of its participation in conjugate heat transfer [Poresky, 2017]. We can model such a simple model using an ordinary differential equation (ODE) which can be solved analytically using Laplace Transforms. Poresky has proposed that the fluid volume (F) within the pipe be modelled as a well mixed control volume with one fluid temperature T_F characterising it. The solid shell (S) would be modelled similarly using a lumped capacitance model with one characteristic temperature T_S. These assumptions were used to analytically derive a transfer function model of the heater. I intend to use this transfer function to generate Bode Plots. I then intend to test the frequency response of a similar model simulated using thermal hydraulics library and superimpose the Bode Plots on top of each other.

Analytical Transfer Function The analytical derivation of the transfer functions is presented here for the reader. While Poresky has essentially the same derivations in his work [Poresky, 2017], some of his derivations contained mistakes which can be verified by inspecting the terms for consistency with units. I present the corrected version here.

Poresky described the energy balance equations in terms of for the fluid and shell as follows [Poresky, 2017]:

$$\frac{dT_F}{dt} = \frac{1}{\tau_F}(T_{F,in} - T_F) + \frac{(hA)_S}{(mc_p)_F}(T_S - T_F) \tag{3.39}$$

$$\frac{dT_S}{dt} = \frac{P}{(mc_p)_S} + \frac{(hA)_S}{(mc_p)_F}(T_F - T_S) \tag{3.40}$$

Here, $(hA)_S$ represents the thermal conductance between the shell and the fluid, P represents the heater power, $(mc_p)_S$ represents the heat capacitance of the solid shell and $(mc_p)_F$ represents the heat capacitance of the fluid. τ_F is the residence time of the fluid. P was recorded in kW while T was recorded in $(^\circ C)$.

The analytically derived transfer function can be expressed as [Poresky, 2017]:

$$\frac{T_F}{P} = \frac{a}{s^2 + bs + c} \tag{3.41}$$

Where:

$$a = \frac{(hA)_S}{(mc_p)_F (mc_p)_S} \tag{3.42}$$

$$b = \left[\frac{1}{\tau_F} + \frac{(hA)_S}{(mc_p)_F} + \frac{(hA)_S}{(mc_p)_S}\right] \tag{3.43}$$

$$c = \frac{(hA)_S}{(mc_p)_S}\left[\frac{1}{\tau_F} - \frac{1}{\tau_F}\frac{(hA)_S}{(mc_p)_F} + \frac{(hA)_S}{(mc_p)_F} - \left(\frac{(hA)_S}{(mc_p)_F}\right)^2\right] \tag{3.44}$$

Unfortunately, the expressions look suspect because of unit inconsistencies. $\frac{(hA)_S}{(mc_p)_F}$ for instance should be in units of Hz and so is $\frac{1}{\tau_F}$. Strangely enough, in Poresky's expression for c, these are added alongside $\left(\frac{(hA)_S}{(mc_p)_F}\right)^2$ which have units of $(Hz)^2$. I therefore have reason to suspect that some of the calculations have some overlooked errata. Therefore, I will perform some calculations to re-derive the final transfer function.

Poresky uses the hat notation to denote a deviation variable. For example, deviation in T is denoted $\hat{T}$. We begin with Laplace Transforms of the equations can be written as [Poresky, 2017]:

$$\hat{T}_F = \frac{\frac{1}{\tau_F}\hat{T}_{F,in} + \frac{(hA)_S}{(mc_p)_F}\hat{T}_S}{s + \frac{1}{\tau_F} + \frac{(hA)_S}{(mc_p)_F}} \tag{3.45}$$

$$\hat{T}_S = \frac{\frac{(hA)_S}{(mc_p)_S}\hat{T}_F + \frac{1}{(mc_p)_S}\hat{P}}{s + \frac{(hA)_S}{(mc_p)_S}} \tag{3.46}$$

These seem correct with reference to the energy balance. Additionally, the units also seem to be consistent. We shall need to substitute out $\hat{T}_S$ to obtain $\hat{T}_F$ in terms of $\hat{P}$:

$$\left(s + \frac{1}{\tau_F} + \frac{(hA)_S}{(mc_p)_F}\right)\hat{T}_F - \frac{\hat{T}_{F,in}}{\tau_F} = \frac{(hA)_S}{(mc_p)_F}\hat{T}_S \tag{3.47}$$

$$\left(s + \frac{1}{\tau_F} + \frac{(hA)_S}{(mc_p)_F}\right)\hat{T}_F - \frac{\hat{T}_{F,in}}{\tau_F} = \frac{(hA)_S}{(mc_p)_F}\frac{\frac{(hA)_S}{(mc_p)_S}\hat{T}_F + \frac{1}{(mc_p)_S}\hat{P}}{s + \frac{(hA)_S}{(mc_p)_S}} \tag{3.48}$$

For a constant flowrate and constant inlet temperature, $\hat{T}_F$ is zero [Poresky, 2017], and these were indeed the conditions for the frequency response test conducted [Poresky, 2017]. For this situation, the term $\frac{\hat{T}_{F,in}}{\tau_F}\left(s + \frac{(hA)_S}{(mc_p)_S}\right)$ is zero, and so we get:

$$\left(s+\frac{1}{\tau_F}+\frac{(hA)_S}{(mc_p)_F}\right)\left(s+\frac{(hA)_S}{(mc_p)_S}\right)\hat{T}_F=\frac{(hA)_S}{(mc_p)_F}\frac{(hA)_S}{(mc_p)_S}\hat{T}_F+\frac{(hA)_S}{(mc_p)_F}\frac{1}{(mc_p)_S}\hat{P} \tag{3.49}$$

$$\left(s+\frac{1}{\tau_F}+\frac{(hA)_S}{(mc_p)_F}\right)\left(s+\frac{(hA)_S}{(mc_p)_S}\right)\hat{T}_F-\frac{(hA)_S}{(mc_p)_F}\frac{(hA)_S}{(mc_p)_S}\hat{T}_F=\frac{(hA)_S}{(mc_p)_F}\frac{1}{(mc_p)_S}\hat{P} \tag{3.50}$$

Thus we have a transfer function connecting $\hat{P}$ and $\hat{T}_F$.

$$\hat{T}_F=\frac{\frac{(hA)_S}{(mc_p)_F}\frac{1}{(mc_p)_S}\hat{P}}{\left(s+\frac{1}{\tau_F}+\frac{(hA)_S}{(mc_p)_F}\right)\left(s+\frac{(hA)_S}{(mc_p)_S}\right)-\frac{(hA)_S}{(mc_p)_F}\frac{(hA)_S}{(mc_p)_S}} \tag{3.51}$$

$$\frac{\hat{T}_F}{\hat{P}}=\frac{\frac{(hA)_S}{(mc_p)_F}\frac{1}{(mc_p)_S}}{\left(s+\frac{1}{\tau_F}+\frac{(hA)_S}{(mc_p)_F}\right)\left(s+\frac{(hA)_S}{(mc_p)_S}\right)-\frac{(hA)_S}{(mc_p)_F}\frac{(hA)_S}{(mc_p)_S}} \tag{3.52}$$

Now, we can multiply out the factors in the denominator,

$$\frac{\hat{T}_F}{\hat{P}}=\frac{\frac{(hA)_S}{(mc_p)_F}\frac{1}{(mc_p)_S}}{s^2+s\left(\frac{1}{\tau_F}+\frac{(hA)_S}{(mc_p)_F}+\frac{(hA)_S}{(mc_p)_S}\right)+\frac{1}{\tau_F}\frac{(hA)_S}{(mc_p)_S}+\frac{(hA)_S}{(mc_p)_F}\frac{(hA)_S}{(mc_p)_S}-\frac{(hA)_S}{(mc_p)_F}\frac{(hA)_S}{(mc_p)_S}} \tag{3.53}$$

After all this, we end up with a simpler looking expression than Poresky's:

$$\frac{\hat{T}_F}{\hat{P}}=\frac{\frac{(hA)_S}{(mc_p)_F}\frac{1}{(mc_p)_S}}{s^2+s\left(\frac{1}{\tau_F}+\frac{(hA)_S}{(mc_p)_F}+\frac{(hA)_S}{(mc_p)_S}\right)+\frac{1}{\tau_F}\frac{(hA)_S}{(mc_p)_S}} \tag{3.54}$$

The units here also look more consistent because s, $\frac{(hA)_S}{(mc_p)_S}$ and $\frac{1}{\tau_F}$ are all in units of s^{-1} or Hz.

The dimensions of the numerator are $\frac{1}{second}\frac{Kelvin}{Joule}$ or $\frac{K}{J\cdot s}$. When divided by the denominator, the units are indeed $\frac{K}{watt}$, which is consistent with the units $\frac{\hat{T}_F}{\hat{P}}$.

If the inlet temperature is taken to be an input to the system, we can re-introduce the ignored term in the numerator. As a result, we would end up with a multiple input single output MISO system:

$$\hat{T}_F=\frac{\frac{(hA)_S}{(mc_p)_F}\frac{1}{(mc_p)_S}\hat{P}+\frac{\hat{T}_{F,in}}{\tau_F}\left(s+\frac{(hA)_S}{(mc_p)_S}\right)}{s^2+s\left(\frac{1}{\tau_F}+\frac{(hA)_S}{(mc_p)_F}+\frac{(hA)_S}{(mc_p)_S}\right)+\frac{1}{\tau_F}\frac{(hA)_S}{(mc_p)_S}} \tag{3.55}$$

We note that the numerator is in units of $K\cdot(Hz)^2$ and the denominator is in units of $(Hz)^2$. For now, we shall just consider oscillating power input to the heater to make the CHT test as simple as possible. The transfer function is:

$$\frac{\hat{T}_F}{\hat{P}} = \frac{a}{s^2 + bs + c} \tag{3.56}$$

Where:

$$a = \frac{(hA)_S}{(mc_p)_F (mc_p)_S} \tag{3.57}$$

$$b = \left[\frac{1}{\tau_F} + \frac{(hA)_S}{(mc_p)_F} + \frac{(hA)_S}{(mc_p)_S}\right] \tag{3.58}$$

$$c = \frac{(hA)_S}{(mc_p)_S} \frac{1}{\tau_F} \tag{3.59}$$

Given this analytical solution, we are able to construct a Bode Plot given this transfer function in MATLAB.This would be the analytical solution Bode Plot. I could also construct a representation of this CHT case in "thermal_hydraulics_rs" using single control volumes and boundary conditions. I would then subject it to a MFBS signal at 10 seconds per bit similar to De Wet's work [De Wet and Per F Peterson, 2020] and at a 9 kW power output, induce oscillations of about 1 kW similar to Poresky's work [Poresky, 2017].

Simulation Setup

Initial and Boundary Conditions The simplified system is assumed be adiabatically coupled to its surroundings in terms of conduction. The initial temperatures were 80°C for both heater and Therminol-VP1, and the boundary conditions for axial conduction for the solid and fluid control volume were an adiabatic boundary condition. Only advection adds or removes heat from the system. Additionally, the inlet temperature is also set to 80°C so that fluid flowing into the system is at 80°C.

Thermal Inertia and Thermal Resistance The working fluid would be Therminol-VP1 (also called Dowtherm A), and the solid shell would be modelled using steel. Material properties for the analytical solution could be derived using properties for steel SS304L for the solid shell and Therminol VP-1 for the fluid at approximately 175 °C for the shell and 95 °C for the fluid [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014; De wet, Per F. Peterson, and Greenwood, 2019]. Mass flow rate would be around 0.18 kg/s as is typical for frequency response tests in CIET [Poresky, 2017]. This would leave the problem fully determined except for the heated pipe dimensions. While the Pr, Re and Nu would most certainly change with temperature, an average temperature can be used to obtained these dimensionless numbers so as to obtain thermal conductance.

The Nusselt number for CIET heater v1.0 based on CIET test bay data was recorded as [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014]:

$$\mathrm{Nu}_{D-hydraulic} = \begin{cases} 8 \text{ if } \mathrm{Re}_{D-hydraulic} < 2000 \\ 5.44 + 0.034\ \mathrm{Re}^{0.82}_{D-hydraulic} \text{ if } \mathrm{Re}_{D-hydraulic} > 2000 \end{cases} \tag{3.60}$$

The heat transfer area used to calculate conductance will be the outer tube heat transfer area for the CIET v1.0 heater at about 311 $inch^2$ or 2007 cm^2 [De Wet and Per F Peterson, 2020]. The main fluid volume is at 42.12 $inch^3$ or 690.2 cm^2 [De Wet and Per F Peterson, 2020]. The hydraulic diameter D_H in the RELAP model is is $6.60{*}10^{-3}m$ [Nicolas Zweibaum, 2015], flow area was about $3.64{*}10^{-4}m^2$ and the heater length excluding the head was about 1.6383 m [Nicolas Zweibaum, 2015].

A summary of important parameters is shown in Table 3.3:

Parameter	Quantity	Unit
Fluid Volume	0.0006902	m^3
Flow Area	0.000348	m^2
Length	1.98333333333333	m
-	-	-
Outer Diameter	0.04	m
Inner Diameter	0.0381	m
Heated length	1.676	m
Shell Volume	0.000195329811289933	m^3

Table 3.3: CIET Heater v1.0 Simplified 1D Model Dimensions

At the average temperature of 95 $°C$, we may use the temperature correlation for viscosity [Nicolas Zweibaum, 2015]:

$$\mu(Pa \cdot s) = \frac{0.130}{T(°C)^{1.072}} \tag{3.61}$$

The resulting viscosity is about 0.000986 $Pa \cdot s$. We can also determine the Nusselt number by first ascertaining what $\mathrm{Re}_{D-hydraulic}$ is at 0.18 kg/s at prevailing flow conditions:

$$\mathrm{Re}_{D-hydraulic} = \frac{\dot{m} D_{hydraulic}}{A_{xs}\mu} = \frac{0.18\ kg/s * 6.60 * 10^{-3}m}{3.64 * 10^{-4}m * 0.000986\ Pa \cdot s} = 3310 \tag{3.62}$$

At $\mathrm{Re}_{D-hydraulic} = 3310$, we have to consider this a turbulent flow regime. Hence, equation 3.60 shall be used. A Nusselt number of 31.6 would be obtained from that.

For the heat transfer coefficient h, we consider that the Nusselt number in this correlation uses the outer tube heated diameter of 3.81 cm as its relevant length scale [De Wet and Per F

Peterson, 2020]. Additionally, thermal conductivity at about 95 $^{\circ}C$ will need to be calculated as well using equation 3.63 [Nicolas Zweibaum, 2015]:

$$k(\frac{W}{m \cdot K}) = 0.142 - 0.00016T(^{\circ}C) \tag{3.63}$$

At 95 $^{\circ}C$, k is $0.1268\frac{W}{m \cdot K}$. We find that h is:

$$h = \frac{Nu_{D-hydraulic}k}{D_{hydraulic}} = \frac{31.6 * 0.1268\frac{W}{m \cdot K}}{6.60 * 10^{-3}m} = 607\frac{W}{m \cdot K} \tag{3.64}$$

For thermal inertia of steel and the Therminol VP-1, we consider that the outer tube of the heater had a heated length of 1.676 m [De Wet and Per F Peterson, 2020], an outer diameter of 4.0 cm and an inner diameter of 3.81 cm. The heat capacity of SS304L was taken at an average temperature of about 450 K or about $177^{\circ}C$ rather than $175^{\circ}C$ so that c_p data could be read directly from tables used in the SAM model [Zou, R. Hu, and Charpentier, 2019] as opposed to interpolation. This was more for convenience than anything else. c_p, ρ, k and μ were taken from previous RELAP and SAM models [Nicolas Zweibaum, 2015; Zou, R. Hu, and Charpentier, 2019].

With these, the thermal inertia of the solid and fluid, the heat conductance and the transfer function parameters a, b and c were calculated and shown in Table 3.4:

Parameter	Quantity	Unit
h_S	607	$W/(m^2\ K)$
A_S	0.2007	m^2
$(hA)_S$	122	W/K
-	-	-
ρ_F	997.25	kgm^{-3}
V_F	0.0006902	m^3
m_F	0.68830195	kg
"$c_{p,F}$"	1785.9	$J/(kg \cdot K)$
$(mc_p)_F$	1230	J/K
-	-	-
ρ_S	8030	kgm^{-3}
V_S	0.000195329811289933	m^3
m_S	1.56849838465816	kg
"$c_{p,S}$"	490.66	$J/(kg \cdot K)$
$(mc_p)_S$	770	J/K
-	-	-
$\dot{m}$	0.18	kg/s
m_F	0.68830195	kg
$\frac{1}{\tau_F}$	0.262	Hz
-	-	-
a	0.000129	$K/(J \cdot s)$
b	0.52	Hz
c	0.0415	$(Hz)^2$

Table 3.4: CIET Heater v1.0 Simplified 1D Model Parameters for Thermal Inertia

For the purposes of validating the thermal hydraulics library, a strictly accurate Nusselt number correlation is not strictly necessary for testing code within the thermal hydraulics library, only that they be consistent between the analytical model and the model built using the library. Therefore, the parameters used as described in Table 3.4 are values not exactly equal to CIET Heater v1.0, but is meant to be in the same ballpark as that of CIET Heater

v1.0.

Now, to test the accuracy of the thermal hydraulics library, I used MATLAB to obtain data for Bode Magnitude and Phase plots. Thereafter, I subjected the heater modelled using control volumes to a multifrequency binary signal (MFBS) and varied its power input from a steady state of 9 kW with an amplitude of 1 kW. The MFBS signal was generated in the Rust programming language using a logspace function which provides a list of frequencies which is evenly spaced in the log-plots. This relatively simple logspace function was generated using Perplexity Artificial Intelligence (Perplexity AI) given to expedite work and also because logspace functions are already well used in other codebases such as Numpy [Harris et al., 2020]. The rest of the code was written without help from AI.The angular frequencies are generated from 0.001 rad/s to 1 rad/s over 15 evenly log spaced points.

Time Stepping For time stepping, the Courant number was used as a basis of the time step. For the Courant number not to exceed 1.0, the timestep had to be about 3.83 s or less. Hence, the timestep used for this simulation was about 3.83 s. For simplicity, the timestep and sampling interval are the same. This implies a sampling frequency of about 0.261 Hz and the Nyquist frequency was about 0.130 Hz. This means that at the prevailing sampling frequency, higher frequencies in the MFBS signal cannot be captured and are instead registered as noise. For example, a frequency corresponding to 1 rad/s cannot be captured as it is over the Nyquist frequency of 0.130 Hz, and therefore it would register as noise. This initial result was problematic, and therefore the timestep in subsequent experiments was set to 0.1s, where the Nyquist frequency was 5 Hz. This would enable it to capture the 1 rad/s signal as well, which is the highest perturbing frequency in the MFBS signal.

Results

Figure 3.10 shows that at frequencies close to the Nyquist frequency, the gains do not correspond to the analytical solution.

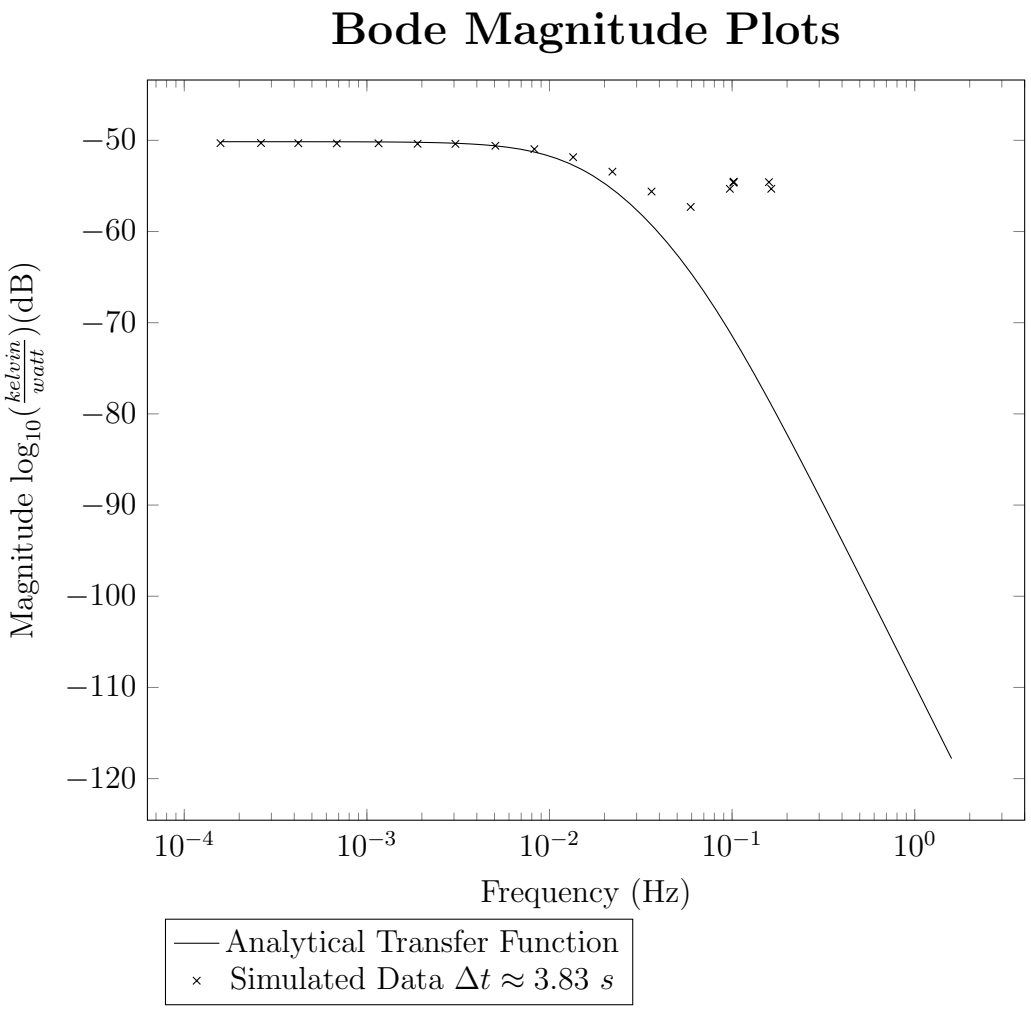


Figure 3.10: Bode Magnitude Plots for One Dimensional CIET Heater

A similar situation applies for the phase plots in Figure 3.11.

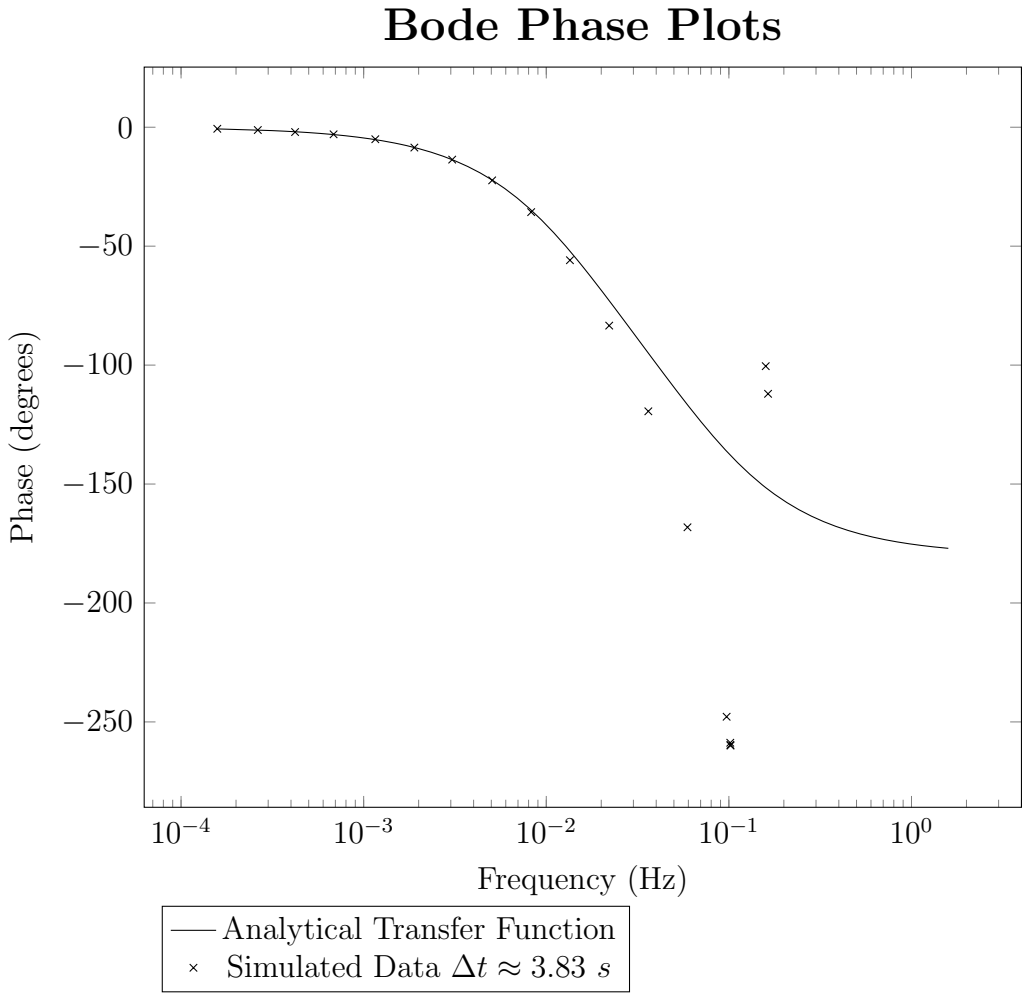


Figure 3.11: Bode Phase Plots for One Dimensional CIET Heater

While first series of tests, does not characterise frequencies higher than the Nyquist frequency well, it does show that the simulation behaves in a stable manner given the auto timestepping algorithm. This is likely because the calculated simulation time is 3.83s. Consequently, using data generated every 3.83s for the Bode Plots means that the sampling interval is 3.83s as well. The sampling frequency here is 0.261 Hz, and therefore, the highest frequncy that can be sampled is around 0.1305 Hz. This is the Nyquist frequency. For oscillations higher than the Nyquist frequency, it would register as noise in the various other frequencies. This would likely explain why there is such a bad fit at the higher frequencies in Figure 3.10.

To obtain a better frequency response signal, a higher sampling frequency is used of about 10 Hz. This means the timestep is 0.1s and the Nyquis frequency is 5 Hz. Therefore frequencies below 5 Hz can be sampled. The resulting Bode plots in Figure 3.12 and Figure 3.13 show that when the sampling frequency increases to 10 Hz, the frequency response

data due becomes cleaner:

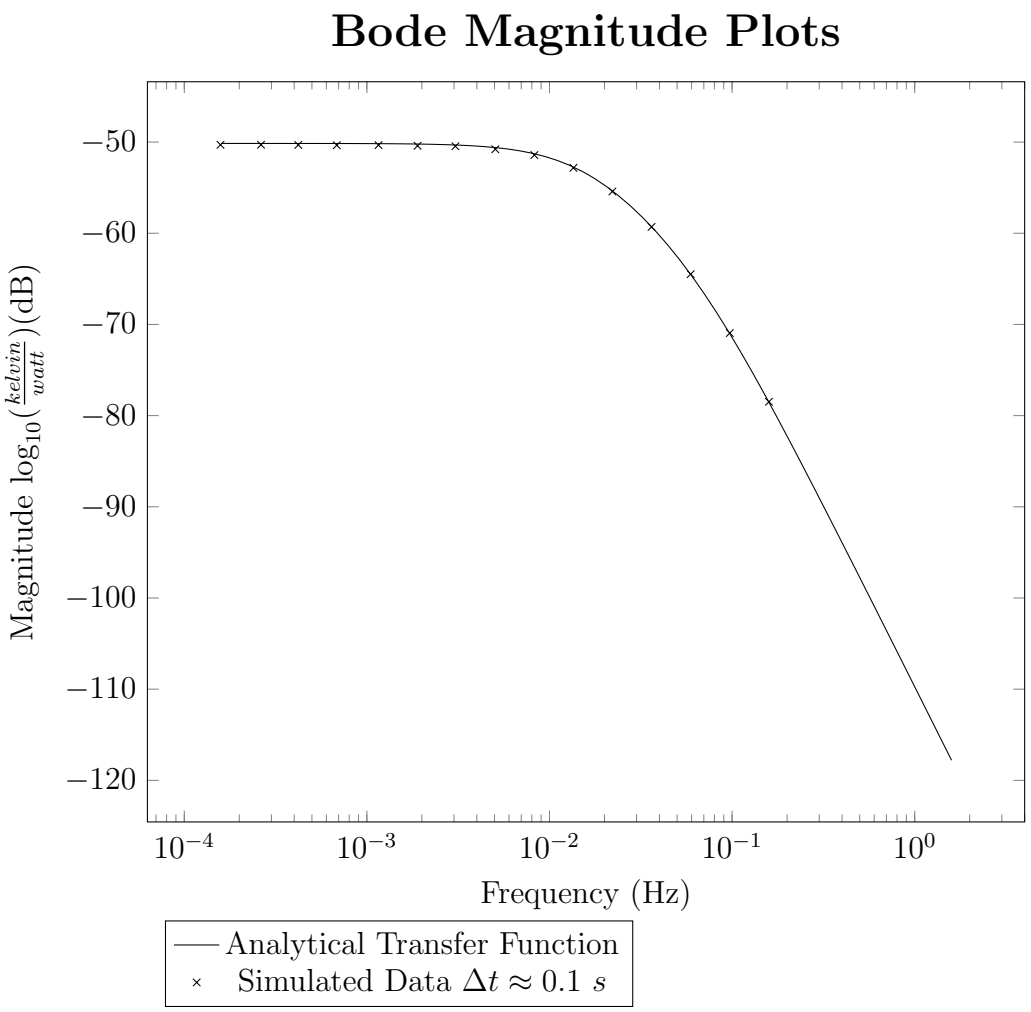


Figure 3.12: Bode Magnitude Plots for One Dimensional CIET Heater

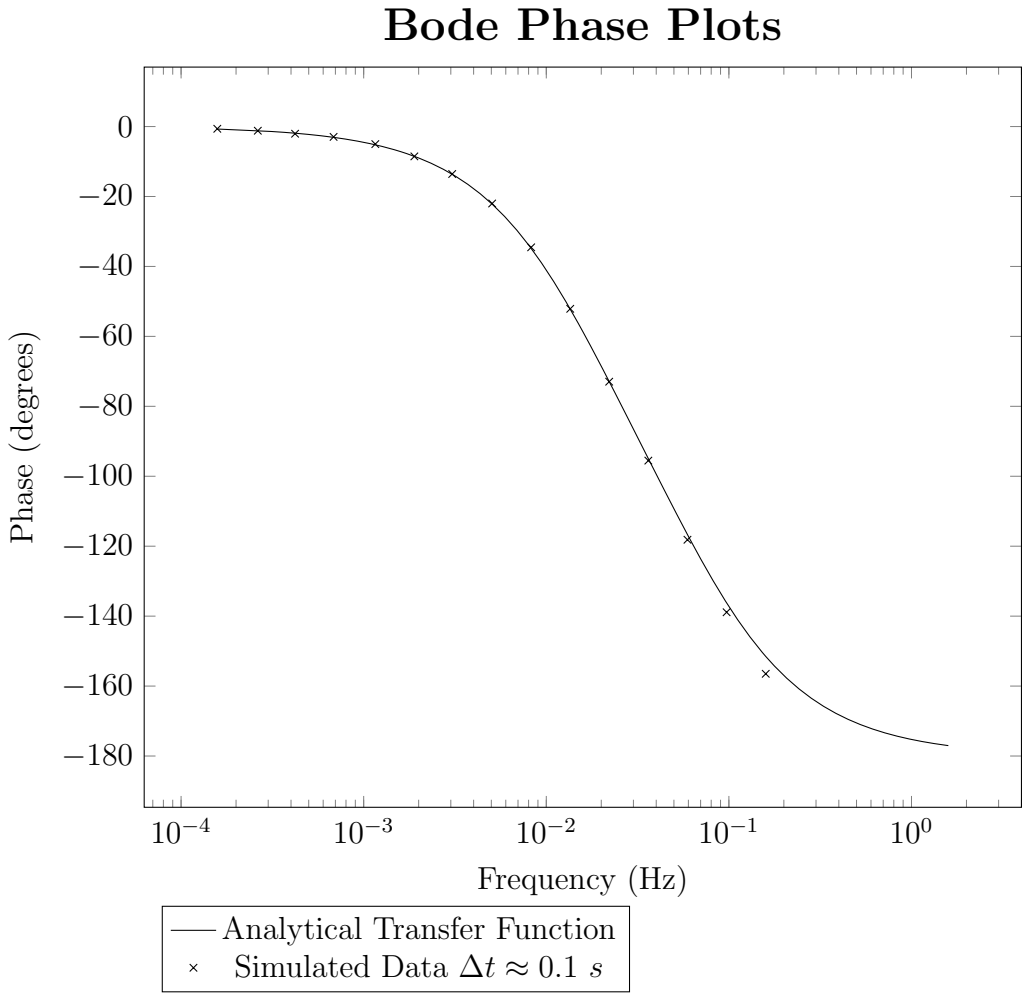


Figure 3.13: Bode Phase Plots for One Dimensional CIET Heater

Now, the plots in Figure 3.12 and Figure 3.13 show that the simulation data matches quite well with the analytically derived transfer function. This shows that the advection heat transfer logic has been programmed correctly into the system.

Once again, I could be more rigourous in this validation effort by providing error bars and showing a step function plot. I will not do so since this test case is not the end product. Its purpose was only to show that the advection heat transfer logic was working correctly. The Bode Phase and Magnitude plots seem to show that this is indeed the case. However, we still need to ensure that the library is successful in simulating the nodalised components within CIET.Since we have bigger fish to fry, I will not conduct further tests on this setup, but rather validate the library in the context of a higher fidelity model of CIET's heater.

CIET Heater v2.0 Bare

Now that we have validated the thermal hydraulics library using simpler test cases, it is now time to fry bigger fish. We shall now attempt to simulate CIET Heater v2.0 Bare. As shown in the last chapter, CIET Heater v2.0 is a heated section with a top head and bottom head. Experimental data from CIET Heater v2.0 was obtained by recording the heater power, inlet temperature and outlet temperature while various steady state experiments and transient experiments were run. The inlet temperature and outlet temperature were measured by thermocouples BT- 11 and BT-12 respectively [De Wet and Per F Peterson, 2020]. If we want to build a Digital Twin of CIET Heater v2.0 Bare which not only replicates the experimental data from BT-11 and BT-12, but also makes predictions of how this same system would behave under other circumstances, then we shall have to model the components as close to the experiment as possible. Between BT-11 and BT-12, there is the CIET Heater v2.0 Bare, and mixer MX-10 [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014; Nicolas Zweibaum, 2015]. This is shown in Figure 3.14:

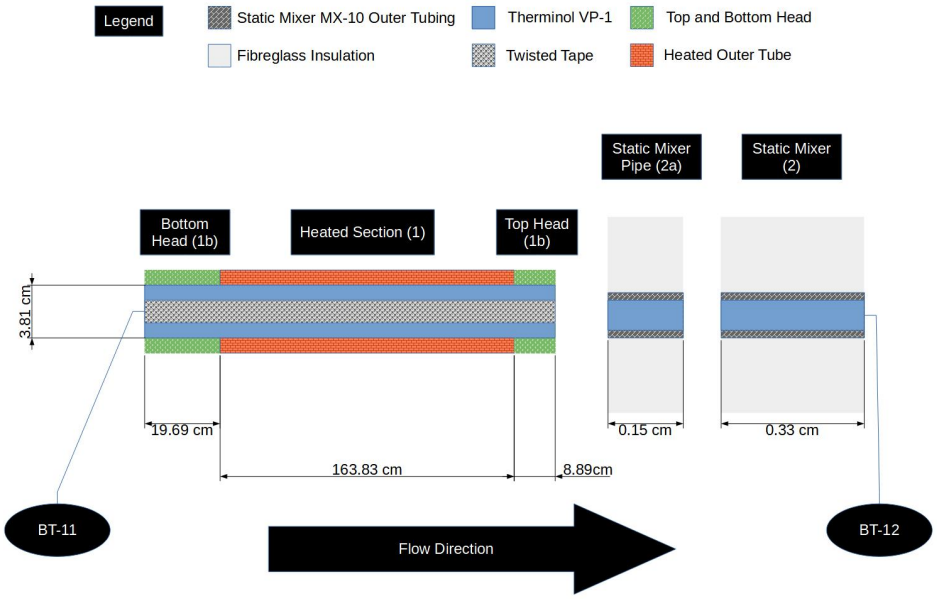


Figure 3.14: CIET Heater v2.0 Bare Layout with BT-11, BT-12 and MX-10 (Not Drawn to Scale)

Figure 3.14 shows the five primary components present in the experiment which I intend to model for the purposes of validation. These components are numbered based on the component numbering convention by RELAP5 [Nicolas Zweibaum, 2015] and SAM [Zou, R. Hu, and Charpentier, 2019].

Now, to produce a Digital Twin of the components between BT-11 and BT-12, I not only require a model which can reproduce the experimental steady state BT-12 temperatures and transient behaviour, I also need to ensure that these models have calculation speeds faster than real-time. Hence, the calculation time relative to the time steps used by the simulations becomes an important metric. This is a complex problem where I use the principles of rapid prototyping and avoid premature optimisation in my development process.

For this dissertation, this means planning, designing and building several prototypes of the eventual heater simulation, and then testing these prototypes to learn lessons that can be applied to future prototypes. This meant that I used extremely simplified heater models during early stages of development work. As I progressed in development work, I added complexity to the heater model to make it more realistic. Figure 3.15 shows what this methodology may look like for this dissertation:

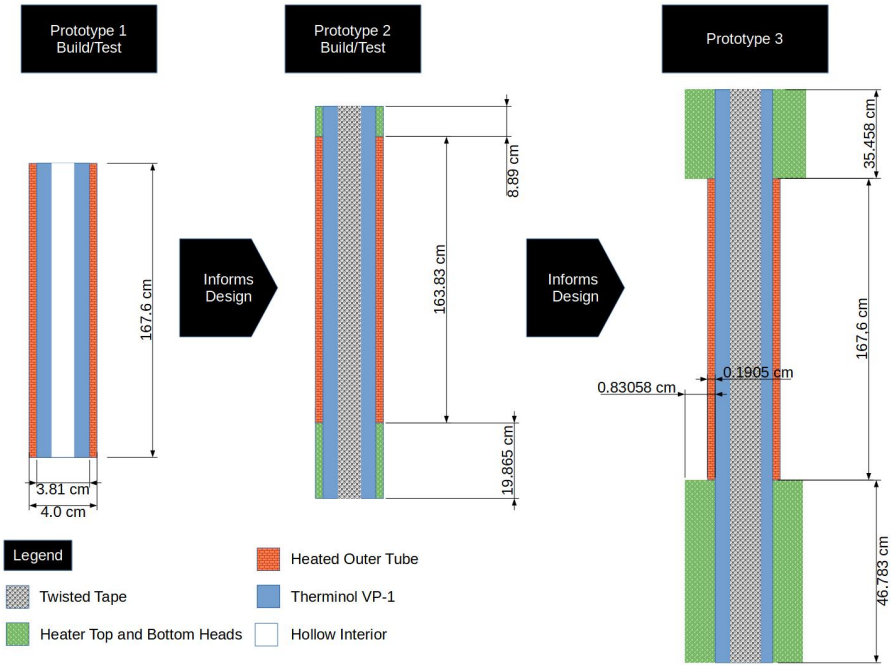


Figure 3.15: Rapid Prototyping Methodology for Designing a Digital Twin of CIET Heater v2.0 Bare (Not Drawn to Scale)

In Figure 3.15, we show three different prototypes of CIET Heater v2.0 Bare models used as a basis to build successively more optimised and realistic simulations of the CIET Heater v2.0 Bare. Prototype 1 is the easiest to develop as it ignores the top and bottom heads as well as the twisted tape and perforated tube within the CIET Heater v2.0 Bare. Therefore, Prototype 1 will be used in the first phase of simulation and code optimisations. Doing so helped me to speed up initial iterations and optimisation of of the control volume calculations. Since Prototype 1 was merely a simplified version of CIET Heater v2.0 Bare

meant to speed code development work, I did not validate it as thoroughly using experimental data for transients. This is because Prototype 1 was never meant to simulate the thermal inertia within CIET Heater v2.0 Bare, but only replicate the steady state data. Therefore, the heater internals are forgone.

After testing and designing several optimisation techniques based on Prototype 1, I then developed simulations based on Prototype 2 which now simulates the heater top and bottom heads using the RELAP5/SAM nodalisation and twisted tape within the heated section as well as the top and bottom heads. This means it replicates the thermal inertia better than Prototype 1. Notably, simulating Prototype 2 would require more control volumes than Prototype 1 as more components are simulated. Hence, the computational demands for simulations based on Prototype 2 are higher as compared to simulations base on Prototype 1. Due to these higher computational demands, Prototype 2 required a higher degree of optimisation in order to ensure that simulations were run in real time. Therefore Prototype 2 was used in the second phase of optimisation for simulations of the CIET Heater v2.0 Bare.

Now, Prototype 2 based on the RELAP5/SAM nodalisation does not account for additional thermal inertia accounted for in the Transform model [De Wet and Per F Peterson, 2020] as described in the literature review chapter. More detailed modelling of these thermal masses is reserved for future work. For this purpose, Prototype 3 is presented as a possible future model which is meant to account for the extra fluid and thermal masses. Its dimensions are based upon my interpretation of Transform Model dimensions of CIET Heater v2.0 Bare as discussed in the last chapter.

In this subsection, we shall outline the experimental data the Prototype 2 model must replicate and the time constraints it must achieve. Next, we shall outline methods used to get the Prototype 1 and Prototype 2 models to meet its time constraints and the results of using some of these methods. Lastly, we shall discuss methods and results for validation of the optimised model against experimental data for the Prototype 2 model of CIET v2.0 Heater Bare.

Experimental Data

For steady state validation, I used forced circulation data for CIET Heater v2.0 Bare at various heater power settings [De Wet and Per F Peterson, 2020] at a prevailing mass flow rate of 0.18 kg/s. The experimental data and heater inlet temperature is shown in Table 3.5:

Heater Power (Watts)	Experimental Heater Inlet Temperature ($^{\circ}C$) BT-11	Experimental Heater Outlet Temperature ($^{\circ}C$) BT-12	Simulated Heater Inlet Temperature ($^{\circ}C$) BT-11
3000	78.75	86.93	78.75
4000	79.00	90.25	79
6000	79.40	96.50	79.4
8000	79.12	102.20	79.12
10000	78.90	107.75	78.9

Table 3.5: Heater v2.0 Bare Steady State Data [De Wet and Per F Peterson, 2020], Prevailing mass flow rate $\dot{m} = 0.18$ kg/s

As shown in Table 3.5, the inlet temperature is 79.0 $^{\circ}C \pm 0.5$ K. The corresponding heater outlet temperature BT-12 given at various power settings is then given in Table 3.5. This data is used for validating steady state behaviour of CIET's heater.

For transient validation tests, we shall use the transient response of the BT-12 temperature to that of De Wet's empirical transfer function of heater power to BT-12 temperature shown in in Equation 3.65 [De Wet and Per F Peterson, 2020]:

$$G(s) = e^{-4s} \frac{3.217 * 10^{-5} s^3 + 6.675 * 10^{-7} s^2 + 1.139 * 10^{-8} s + 2.423 * 10^{-11}}{s^5 + 0.2251 s^4 + 0.01688 s^3 + 0.0003548 s^2 + 3.057 * 10^{-6} s + 1.632 * 10^{-9}} \quad (3.65)$$

Equation 3.65 expresses the deviations in heater power to deviations in BT-12 temperature as an empirical transfer function. This was based on frequency response testing for a PRBS signal of 63 bits, 500 watts and 10 seconds per bit after CIET reached a steady state based on a heater power of 8 kW at a Therminol VP-1 mass flow rate of 0.18 kg/s [De Wet and Per F Peterson, 2020]. The heater inlet and outlet temperature at this steady state prior to the frequency response test is the same as that reflected in Table 3.5 for the 8 kW run. As mentioned in the previous chapter, this transfer function also accounts for thermal pulses that traverse round the loop. these thermal pulses take about 1 minute to traverse the loop, thus the inlet temperatures would change. If we subject this transfer function to some step test, the response prior to roughly 1 minute would be based on a constant inlet temperature. This data is usable for validating the response of the heater outlet temperature BT-12 given a constant inlet temperature and user set step input for heater power. This is the data I use to validate the BT-12 response to a step increase in heater power.

Time Requirements

Now that we have discussed the experimental data used to validate the model of CIET Heater v2.0 Bare, we now want to discuss some of its time constraints. Modelling CIET Heater v2.0 Bare is part of a much bigger ambition where I want to model the whole of CIET in real-time. To do so, I want to ensure that the digital twin library I develop is capable of simulating the entirety of CIET in future. To do so, there are certain time requirements this digital twin of CIET Heater v2.0 Bare must meet.

Let's consider the breakdown in time available for us. Based on the SAM model [Zou, R. Hu, and Charpentier, 2019], there are about 57 fluid components to model. This includes the scaled natural circulation Direct Reactor Auxiliary Cooling System (DRACS) loop as well as the forced circulation loop. Both loops are shown in Figure 2.4, except for any structural supports. For structural components, we consider that based on De Wet's Transform model diagram in his work, there are about 7 modelled support structures within the forced circulation loop [De Wet and Per F Peterson, 2020]. Let us assume now that we need to model another 7 within the within the natural circulation loop for the sake of estimating computational burden. We would then arrive at a total of 71 components to model in CIET.

For all components, I want to solve the mass balance, energy balance and momentum balances. Of course, we do not have to solve for mass balances across these support structures, but then we can still use this number as a conservative estimate for the computational burden required at each time step. Let us assume that 70 fluid components worth of computation need to be simulated in real-time. The methodology I chose was the operator split method as explained in the last chapter. Hence, the mass, momentum and energy balances are to be solved separately before being recombined as a solution for the next time step. Solving these equations in real-time can be challenging. For real-time calculations, we need to find out a time interval for which all these equations need to be solved. As discussed in previous work, a suitable calculation time we need to meet is about 100 ms as this is the time interval for which data is sent from CIET's sensors to the Advanced Reactor Control and Operations (ARCO) control system [Ong, 2023]. We now need to ration these 100 ms of calculation time for solving mass, momentum and energy balance across all 70 components. Again, these are fluid components shown in Figure 2.4 plus roughly 14 support structures.

To estimate the computational burden for the mass and momentum balance, we can first refer to my master's thesis. Based on previous work, I was able to get the mass and momentum balances for the isothermal digital twin of CIET under 80 ms on a gaming laptop [Ong, 2023]. This time scale was based on unoptimised compilation, otherwise known as "debug" compilation. If I optimise the compilation, the speed would increase by roughly 10 times based on previous anecdotal experience. While I could use 8 ms as an estimate for which I could solve the mass balance and momentum balance equations, I could not use this timing because I needed to make changes to my original isothermal digital twin and re-run the timed trials.

I had one primary reason for this re-run: if I wanted to build a full digital twin of CIET eventually, I would need to ensure that passing information between mass, energy and

momentum balance is as seamless as possible. The most practical way to ensure this is done is to ensure that the Digital Twin is written in one language only. For this case, it is the Rust programming language. As it stands in my master's thesis, this was not the case because the OPC-UA client and some of the numerical solvers use Application Program Interface (API) in the Python programming language [Ong, 2023] in addition to using the Rust programming language for large parts of my solver. Hence, there was a language interface where I had to convert data in my Rust based fluid mechanics calculation data into Python based floating point numbers. Repeating this process for about 50 components would prove cumbersome. Hence, I wanted to rewrite my isothermal Digital Twin of CIET completely in Rust so that I would not have the to put in extra effort to convert Rust data into Python data and vice versa.

To convert the Python-Rust Isothermal Digital Twin into one based fully on Rust, I needed to find a way to use an OPC-UA server API based completely in Rust. Moreover, I also needed to use a root finding library in Rust rather than use Python's optimised "SciPy" library. I found that the OPC-UA server and client implementation for Rust by Github user "locka99" [Locka99, 2022] most useful for this purpose. For the root finding library, I used the "roots" crate by Github user "vorot" in Rust [vorot, 2022], which had a Brent-Dekker algorithm similar to what I used in numpy. Compared to the isothermal Digital Twin in my master's thesis, were no changes to the code or algorithm other than essentially a language switch. While a language change may seem insignificant in terms of the algorithms and principles involved, it is significant for time. While the Rust programming language is known for its speed, the "SciPy" libraries in Python could be faster because it uses optimised C and Fortran under the hood [Virtanen et al., 2020]. Therefore, I needed to re-run the timed trials in order to ensure that the code timing data collected was relevant to a Digital Twin completely written in Rust.

For these tests, the test conditions were essentially the same as that mentioned in my master's thesis [Ong, 2023]. That is, CIET is completely isothermal at $20°C$. The tests were run by setting a loop pressure drop (or equivalently the pressure difference over the pump), and observing the mass flow rate. The mass balance I solved was that the sum of mass flow rates flowing through the DHX branch and Heater branch in Figure 2.4 is equal to the mass flow rate of the CTAH branch. The mass balance across all i components within each branch becomes:

$$\dot{m}_i = constant$$

For a circuit of fluid components in series, the mass flowrate through each component ($\dot{m}_i$) is the same. The momentum balance essentially reduces to the pressure changes over all three branches being equal. The pressure change across each branch was given in the last chapter as Equation 2.21 [Ong, 2023]:

$$\Delta P_{change} - \sum_{i}^{n} \Delta P_{hydrostatic\ i} - \sum_{i}^{n} \Delta P_{source\ i} = -\sum_{i}^{n} \frac{1}{2} \frac{{\dot{m}_i}^2}{\rho A_{XS,i}^2} (f_{darcy,i} \frac{L_i}{D_i} + K_i) \qquad (2.21)$$

The factors ($f_{darcy,i}\frac{L_i}{D_i} + K_i$) as well as the flow area (cross sectional area) A_{XS} are found in Table 2.9. The hydrostatic pressure change across each pipe can also be found based on the component length and vertical angle Table 2.9. The Brent- Dekker algorithm iteratively solves the mass flow rates across each branch until the pressure change across each branch is equal. This algorithm was mentioned in detail in my master's thesis [Ong, 2023], and I will not repeat it here. Again, while the algorithms are conceptually the same, the language and degree of optimisation differs. Therefore, we need to re-time the isothermal solver runs.

The digital twin server essentially solves these equations within a set time. If the server is able to reproduce the data from the previous Digital Twin in my master's thesis, then the language conversion process for the Digital Twin is successful. I can then use the timed trial of this Digital Twin written in Rust to determine the time constraints for CIET Heater v2.0 Bare. These timed trials are run on the same AFTERSHOCK PC PTE LTD. (from Singapore) custom gaming laptop as before in my master's thesis, with 31 GB of Random Access Memory (RAM), using the Arch Linux operating system with the following Central Processing Unit (CPU):

```
Intel i7-10875H @ 2.400 GHz
```

Do note that while the Intel i7-10875H CPU can have clock speeds up to 5.100 GHz, I chose to throttle down my CPU on my gaming laptop so that it does not overheat. At 5.100 GHz with 16 active threads, the temperatures can reach up to 91 $°C$ for high intensity calculations. This is not good for CPU longevity. At clock speeds of 2.400 GHz, I did not get temperatures higher than 80 $°C$ even when all cores are used for calculation.

The OPC-UA client I used was still written in Python, same as in my previous work [Ong, 2023] because of prior familiarity. However, the choice of client should not affect the timed runs because the calculations are performed on the server side.

For code validation, I ran the digital twin by specifying several set loop pressure drops in both forward and reverse flow configurations to produce the system curve, essentially repeating validation tests outlined in my master's thesis [Ong, 2023]. The results are shown in Figure 3.16:

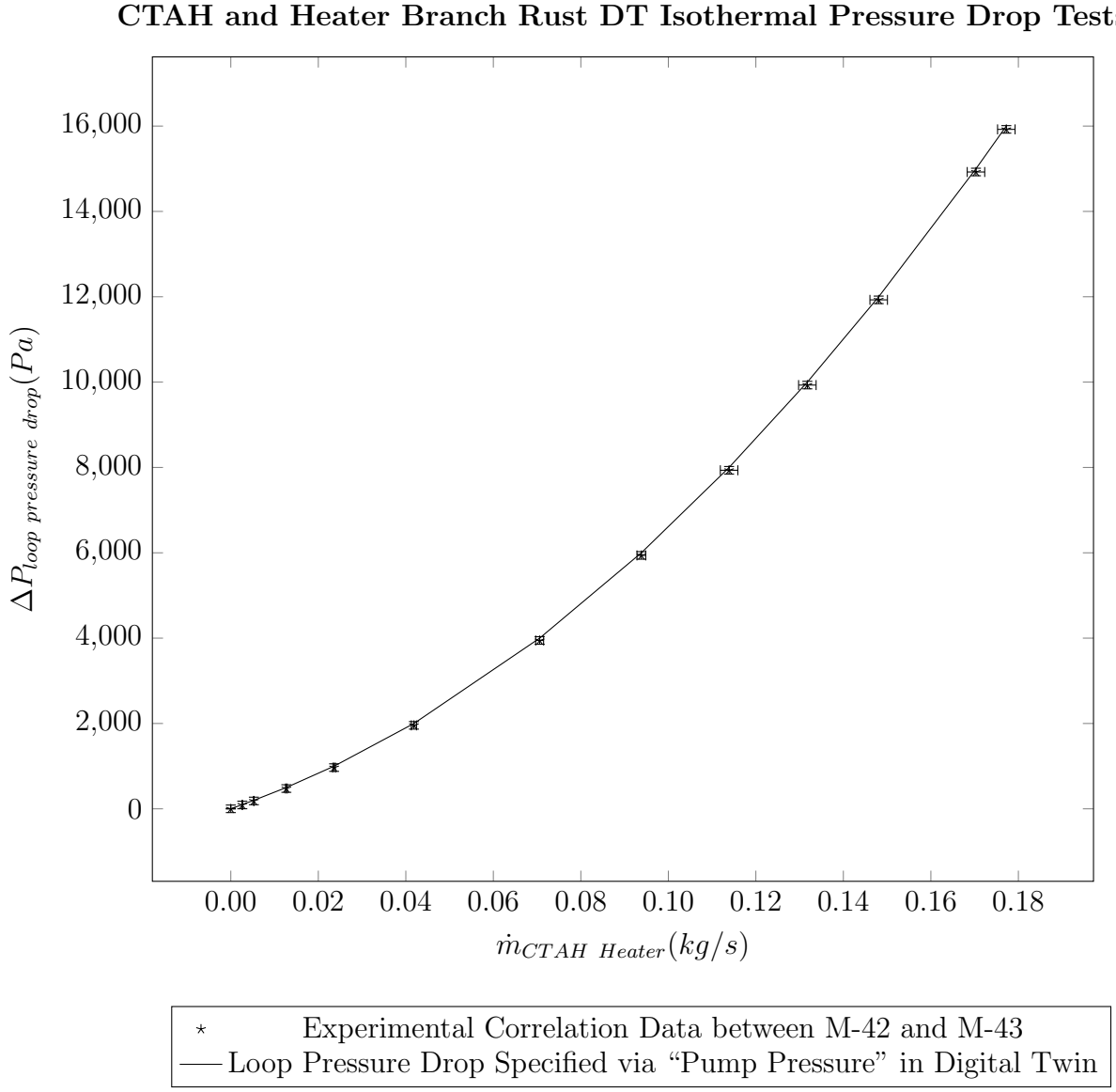


Figure 3.16: Rust Isothermal Digital Twin System Curve Compared to Experimental System Curve from Manometer M-42 and M-43 Plotted in Absolute Pressure Units (CTAH and Heater Branch)

Tabulated values for Figure 3.16 are shown in Table 3.6:

FM-40 Mass Flowrate (kg/s)	Digital Twin Loop Pressure Drop (Pa)	Experimental Data Correlation Pressure Drop (Pa)
0.177	16000	15920
0.170	15000	14920
0.148	12000	11930
0.132	10000	9930
0.114	8000	7930
0.0938	6000	5940
0.0706	4000	3950
0.0418	2000	1960
0.0236	1000	970
0.0127	500	480
0.00527	200	180
0.00263	100	90
0	0	0
-0.0418	-2000	outside data range
-0.132	-10000	outside data range

Table 3.6: Rust Digital Twin System Curve Compared to Experimental System Curve from Manometer M-42 and M-43 Plotted in Absolute Pressure Units (CTAH and Heater Branch)

The results for the Rust Digital Twin are identical to that found in previous work [Ong, 2023] as the algorithms are essentially the same, just expressed in different code. Hence, these numerical simulations rewritten completely in Rust produce the essentially the same results. Therefore, residual plots are not provided as these are identical to those found in my master's thesis. The results show that the port from a Python OPC-UA server to a Rust OPC-UA server was done successfully.

For timed trials, the root finding algorithm written in Rust was somewhat slower than Python's numpy libraries when compiled in "debug" mode. This took on the order of 200 to 240 ms to solve for mass flow rate given the prevailing CPU clock speeds. Thus, compiling

the Rust Digital Twin in the unoptimised debug mode yielded a slower result than the Rust and Python Digital Twin based on the Optimised "SciPy" libraries. However, when compiled in release mode, the calculation times take only 25 ms at most. Thus, these calculations are faster than real-time. While the Rust digital twin is slower than the previous isothermal Digital Twin using the optimised SciPy libraries, this speed is still sufficient to ensure that the calculations are run in real-time.

Based on these calculations, I infer that the fluid calculations in the natural circulation DRACS loop would take < 25 ms. This is because the DRACS loop has a simpler flow configuration in comparison to the main loop as shown in Figure 2.4. Thus, the fluid calculations take 50 ms at most if done sequentially. However, I could just run the DRACS loop calculations in parallel with the forced circulation loop calculations to essentially have fluid calculations take a total of 25 ms.

If we consider having some buffer time for these calculations, then for every 100 ms, we have about 30 ms with which to calculate our fluid mechanics and 70 ms with which to calculate heat transfer for each of our components. If have 70 components within CIET, then we have about on the order of 1 ms to perform the heat transfer calculations per 100 ms of simulation time per component. The requirements are even more stringent if the required time steps are shorter than 100 ms. If the required time step for stability is 15 ms. Then we shall need to perform about 6 to 7 heat transfer calculations per 100 ms. This implies that the time allowed to calculate heat transfer for each component is about 140 μ s. Therefore, for 8 components, that is the three components for CIET Heater v2.0 Bare, the two components for the MX-10 static mixer and three dummy support structures, we are allowed roughly 1 ms of real world time to calculate one time step's worth of heat transfer calculations for all 8 components. This number, of course, assumes that we perform the mass, momentum and energy balance calculations sequentially (serially) rather than in parallel. If we leverage parallel computation to calculate the mass flow rates and energy balances in parallel, we would have even more time per 100 ms of simulated time. There is, of course, a limit to how many processes we can run in parallel. For the hardware I used, the i7-10875H, it comes with 8 cores and 16 threads. Having 8 cores means I can run roughly 8 tasks in parallel at maximum.

Regardless, based on this approximate analysis, as long as we can meet this requirement of 1 ms for all 8 components, there is a good chance that a real-time digital twin can be constructed. This, of course, assumes that the time step required for simulation stability is about 15 ms. The time steps required for simulation stability, based on the Courant number and mesh Fourier number, is quite dependent on the mesh configuration of the system. In the finalised mesh, I found a time step of 15 ms to be suitable for a stable simulation.

Let us now consider next how the mesh is constructed and how the system is optimised to meet these real-time requirements.

Methods for Meshing and Optimisation

We have established previously that if the heat transfer calculations require a time step of 15 ms, we must complete all these heat transfer calculations within 1 ms in order to ensure that the final digital twin has real-time simulation capability. I did a number of prototyping iterations for the heater simulation in order to develop the final product with this capability. Here, we discuss some of the key learning points learnt in the iterative development process. We also share some of the final results of the optimisation as well as how the intermediate models compare to the experimental data.

Optimisation Geometry and Components Developing a CIET Heater v2.0 Bare simulation from scratch in Rust with real-time was challenging. Therefore, I started initial optimisation tests by creating models based on Prototype 1 as described in Figure 3.15.

In these initial optimisation tests, I used values for the dimensions of the heated section of CIET Heater v2.0 Bare based on De Wet's Transform model at first rather than the RELAP5 and SAM model because the Transform model has a simpler nodalisation scheme. The number of axial nodes in De Wet's Transform models was 8 [De Wet and Per F Peterson, 2020], whereas the number of axial nodes in RELAP5 and SAM was 15 [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015]. I call these initial optimisation tests based on Prototype 1 "phase one" of my optimisation process.

After phase one optimisation, I grew more familiar with the optimisation techniques and had developed code that automates constructions of uniform 1D meshes. With lessons learnt from optimisation phase one, I could then base my heater model on Prototype 2 shown in Figure 3.15 and model all components between BT-11 and BT-12 and optimise the calculation speed for all these components. Optimisation processes based on prototype 2 became what I called "phase two" optimisation. The key components simulated during phase one and phase two optimisation can be visualised using Figure 3.17:

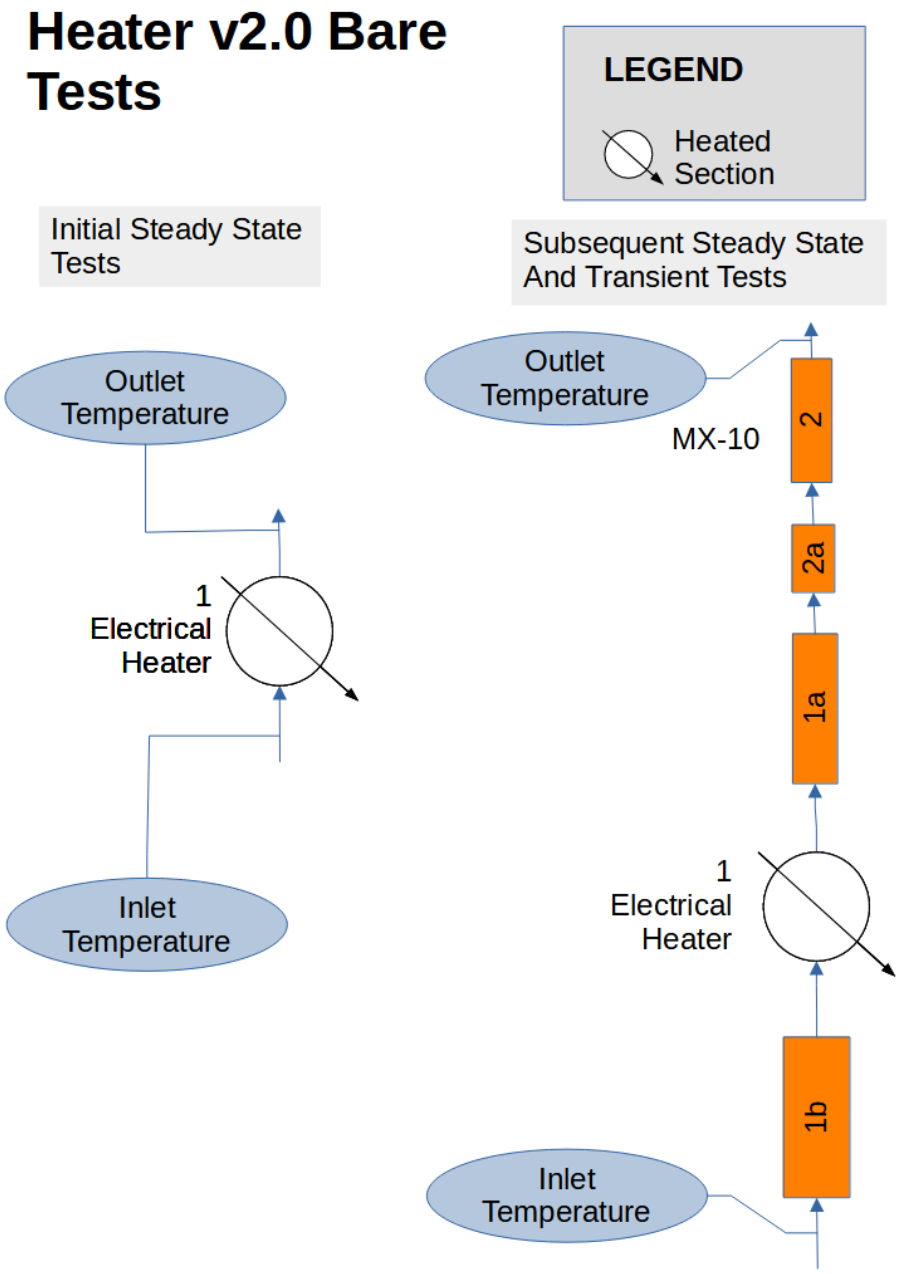


Figure 3.17: Heater v2.0 Bare Initial Tests (Optimisation Phase One) and Subsequent Tests (Optimisation Phase Two)

As seen in Figure 3.17, we only simulate the heated section in phase one, whereas we simulate several components in phase two. In Figure 3.17, 1b is the heater bottom head, 1a

is the heater top head, 2a is the static mixer pipe for MX-10 and 2 is the static mixer MX-10. I decided that these components were important to simulate based on CIET's Process and Instrumentation Diagram (P&ID) which showed the locations of thermocouples BT-11 and BT-12. Thermocouple BT-11, which captured the CIET Heater inlet temperatures, is situated before tee F-11 in the P&ID diagram [Nicolas Zweibaum, 2015] whereas thermocouple BT-12 measuring the Heater outlet temperature is situated after static mixer MX-10 [Nicolas Zweibaum, 2015]. However, it was not explicitly stated where these thermocouples should be in the RELAP5 or SAM model, I could only estimate what these components were. Based on my best estimates, the components I needed to model were the CIET Heater v2.0 Bare as well as the Static Mixer MX-10. Therefore, the two MX-10 components, the heater top and bottom head are added into phase two of the optimisation procedures.

In addition to simulating the five components shown in Figure 3.17, I also wanted to simulate the computation time required by support structures. This is because the support structures contribute significantly to the thermal inertia and parasitic heat losses in the loop [De Wet and Per F Peterson, 2020]. I do not intend to simulate the parasitic heat losses which occur due to these support structures in an accurate manner for this dissertation. Only their computational burden is simulated. Therefore, I added three dummy support structures to the heater and to Static Mixer MX-10 during phase two optimisation. For the purposes of model validation, these dummy support structures are decoupled from the heater and MX-10.

Any study of related to support structures and their associated parasitic heat losses is left for future work. Hence, I did not simulate support structures in the final model. For this dissertation, as long as the final model is able to replicate the transient response of the transfer function to within thermocouple uncertainty, it will suffice as a model with which to test the simulated neutronics feedback controller. In this dissertation, I aim to use validate the final model developed in phase two optimisation using De Wet's experimental [De Wet and Per F Peterson, 2020] described earlier in the chapter.

Optimisation Techniques

Optimisation Phase One Techniques and Procedures In phase one, I focused my efforts on studying the effectiveness of several optimisation techniques using Prototype 1 geometry shown in Figure 3.15. These techniques included compiler optimisation, parallelisation, and mesh coarsening. Table 3.7 shows the test runs used to investigate the effectiveness of these different techniques:

Timed Trial	Number of Uniform Axial Fluid Control Volumes	Number of Uniform Axial SS 304L Heated Tube Nodes	Number of Radial SS 304L Heated Tube Nodes	Compilation Mode	Maximum Number of Threads
Debug	8	8	2	Debug	1
Release	8	8	2	Release	1
Release and Parallelised Node Connections	8	8	2	Release	8
Release and One Radial Steel Node	8	8	1	Release	1
Release and Three Axial Nodes	3	3	2	Release	1

Table 3.7: Table of Phase One Test Inputs, All Timed Trials have Input Temperature of $79.12°C$ and Ambient Air Temperature of $21.67°C$

Table 3.7 shows five different test runs used in the earliest runs of phase one optimisation. The "Debug" timed trial was a base case in which the compiler was run in an unoptimised fashion, otherwise known as "Debug" mode in Rust. The "Release" timed trial was the same as the "Debug" timed trial except that optimised compilation was used. The third trial in Table 3.7 was the "Release" timed trial with computer parallelism introduced. For the first three runs in Table 3.7, the mesh is identical. However, I studied mesh coarsening using the last two timed trials in Table 3.7. Hence, a different mesh was used. The fourth run was used to study mesh coarsening in the radial direction and fifth run was used to study mesh coarsening in the axial direction.

In the first three timed trials in Table 3.7, namely the "Debug", "Release" and "Released and Parallelised Node Connections" timed trials, the heated section itself is simulated using the meshing scheme shown in Figure 3.18:

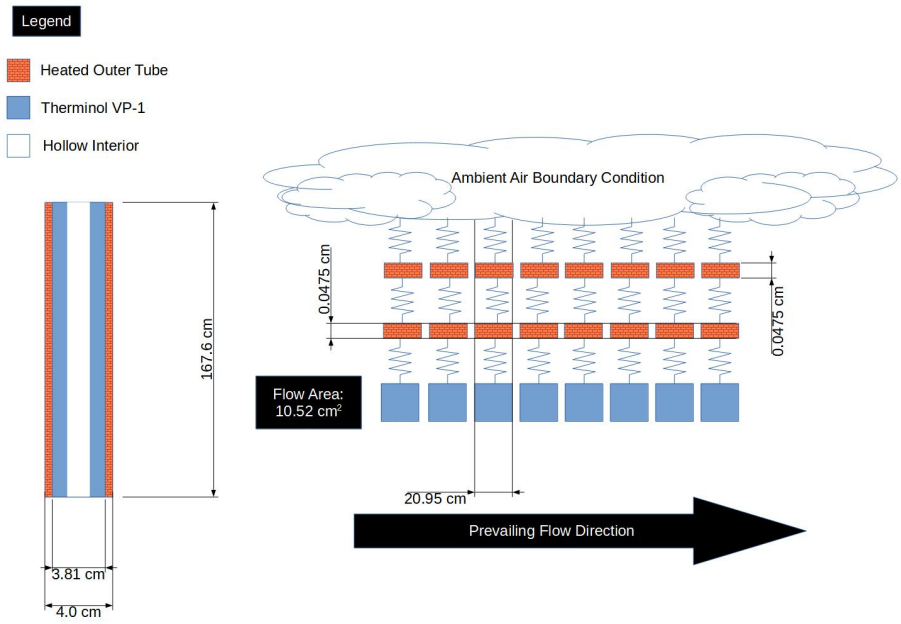


Figure 3.18: Optimisation Phase One Geometry and Mesh for the Baseline Model

As shown in Figure 3.18, I used De Wet's transform nodalisation and heated section geometry for phase one optimisation tests where two radial nodes and eight axial nodes were used to simulate the heated section. The axial thermal conductances are assumed to be zero for solid and fluid nodes. The time coupling scheme was also explicit because I had not developed semi-implicit solvers at this stage of iterative development. These control volumes were simulated using "SingleCV" objects mentioned earlier. The energy balance equations for "SingleCV" objects are explicit as shown in Equation 2.25 in the previous chapter:

$$
\begin{aligned}
m_{cv}\frac{h^{t+\Delta t}_{enthalpy,i} - h^{t}_{enthalpy,i}}{\Delta t} = & \sum_{j}^{N} H^{t}_{thermal,self \leftrightarrow j,i}(-T^{t}_{i} + T^{t}_{j,i}) \\
& + \dot{m}^{t}(-h^{t}_{enthalpy,i} + h^{t}_{enthalpy,i-1}) \\
& + H^{t}_{thermal,conduction,(i-1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i-1}) \\
& + H^{t}_{thermal,conduction,(i+1)\leftrightarrow i}(-T^{t}_{i} + T^{t}_{i+1}) \\
& + \sum Q^{t}_{gen,i} + \sum Q^{t}_{boundary\ conditions,i}
\end{aligned}
$$

The radial thermal conductances in Equation 2.25 need to be calculated. Figure 3.19 shows the radial thermal resistance diagram for the baseline model used in the "Debug" run:

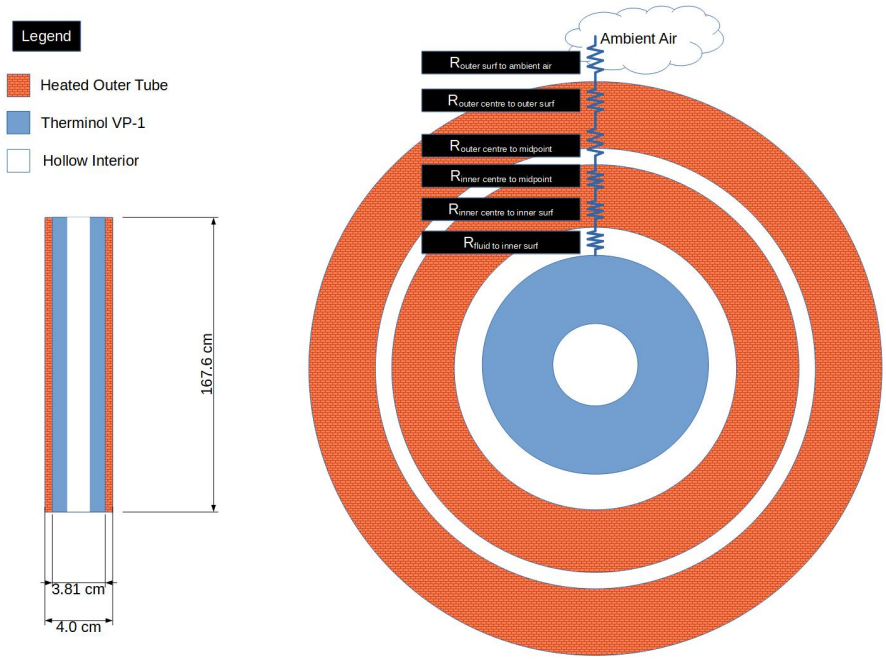


Figure 3.19: Thermal Resistance Diagram for Phase One Baseline Model (Not Drawn to Scale)

Figure 3.19 shows six thermal resistors in the radial direction. This represents the full thermal resistance from the Therminol VP-1 to ambient air through the heated tube. The ambient air thermal conductance is calculated using:

$$\begin{aligned} H_{outer\ surf\ to\ ambient\ air} &= \frac{1}{R_{outer\ surf\ to\ ambient\ air}} \\ &= h_{outer\ surf\ to\ ambient\ air} A_{surface} \\ &= h_{outer\ surf\ to\ ambient\ air} \pi L_{node} \mathrm{OD}_{heated\ tube} \end{aligned}$$

$h_{outer\ surf\ to\ ambient\ air}$ is 20 $W/(m^2 \cdot K)$ as described in the literature review chapter. L_{node} is the node length. Nodes are uniformly spaced with lengths and radial thicknesses shown in Figure 3.18. Finally, $\mathrm{OD}_{heated\ tube}$ is 4.0 cm. The thermal conductance from the fluid to inner surface of the heated tube is calculated in a similar manner using using empirical Nusselt number correlations developed for the heater [Lukas, Kendrick, and P. Peterson, 2017; De Wet and Per F Peterson, 2020]:

$$\mathrm{Nu}_{D-hydraulic} = 0.04179\ \mathrm{Re}_{D-hydraulic}^{0.836}\ \mathrm{Pr}^{1/3} \tag{3.66}$$

The hydraulic diameter to be used in calculating $\mathrm{Re}_{D-hydraulic}$ is 1.467 cm as discussed in the previous chapter. For simplicity, all fluid properties are evaluated at the bulk fluid temperature of each control volume for the purposes of calculating a local $\mathrm{Nu}_{D-hydraulic}$.

As mentioned in the previous chapter, the thermal resistance for cylindrical geometry can be expressed as [Perry and Green, 2015]:

$$R_{thermal,cylinder} = \frac{\ln(r_2/r_1)}{2\pi kL}$$

As shown Figure 3.19, there are four thermal resistors that need to be calculated using $R_{thermal,cylinder}$. In all cases, the thermal resistance from the control volume surface to control volume centre is approximated by using the half thickness of the cylinder.

As an example, let us calculate r_1 and r_2 for thermal resistances for a cylinder of inner diameter ID and outer diameter OD. From the inner surface to centre:

$$r_1 = \frac{ID}{2}$$

$$r_2 = r_1 + \frac{OD - ID}{4}$$

From the outer surface to centre:

$$r_1 = \frac{ID}{2} + \frac{OD - ID}{4}$$

$$r_2 = \frac{OD}{2}$$

In essence, I neglected the effect of curvature so that the thermal resistance from the outer surface to the tube centre is slightly different from the inner surface to the tube centre for simplicity. In this regard, I model the thermal resistance in a slightly asymmetric manner. However, for steady state simulations, this should not matter because the sum of thermal resistances from the inner Therminol VP-1 region to the ambient air should still be the same. For this dissertation, I used this simplified method to calculate thermal resistances in cylinders. I will test if this simplification significantly affects the transient results, but this will only be done in the model validation portion later in the chapter.

Now that we have discussed the meshing scheme in the radial direction for the base case, let us now discuss mesh coarsening. The meshing scheme for mesh coarsening in the radial direction, the "Release and One Radial Steel Node" in Table 3.7 is shown in Figure 3.20:

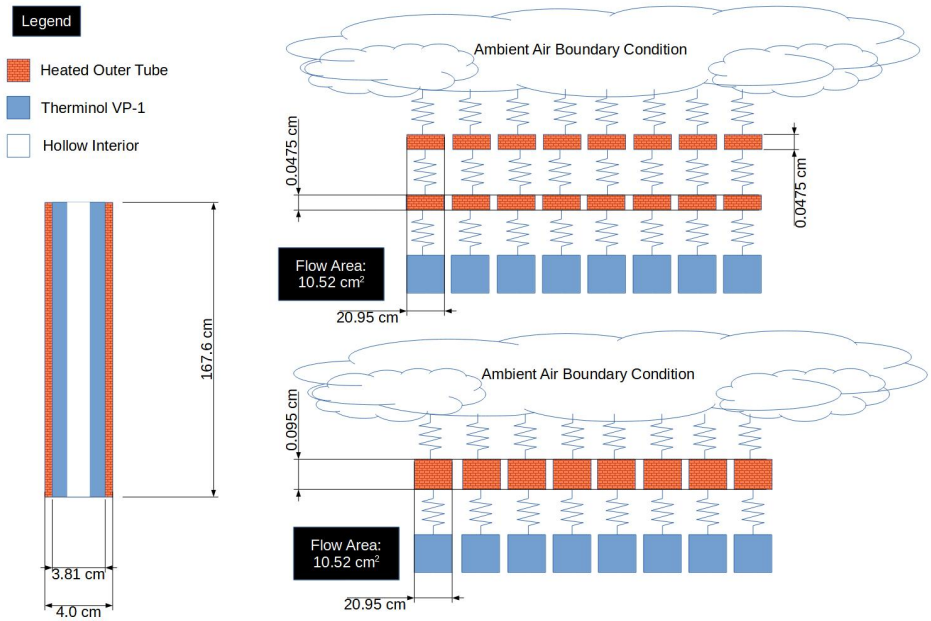


Figure 3.20: Meshes Before Radial Mesh Coarsening (Above) and After Mesh Coarsening (Below) for phase one optimisation (Not Drawn to Scale)

The radial thermal resistance diagram when mesh coarsening in the radial direction is shown in Figure 3.21:

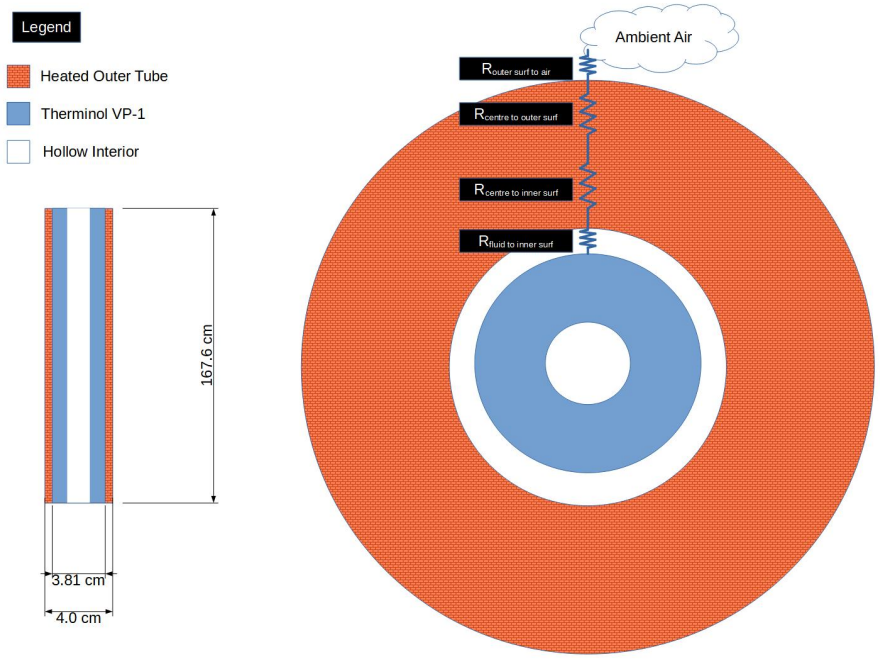


Figure 3.21: Thermal Resistance Diagram after Radial Mesh Coarsening (Not Drawn to Scale)

The principles for calculating the relevant thermal resistances are the same as that for the meshes for the base case shown in Figure 3.19. Essentialy, the thermal resistances of the centre of the control volume to both the inner surface and the outer surface are based on the same radial thickness for simplicity such that I ignore its curvature effects as discussed earlier.

The mesh for axial mesh coarsening is shown in Figure 3.22:

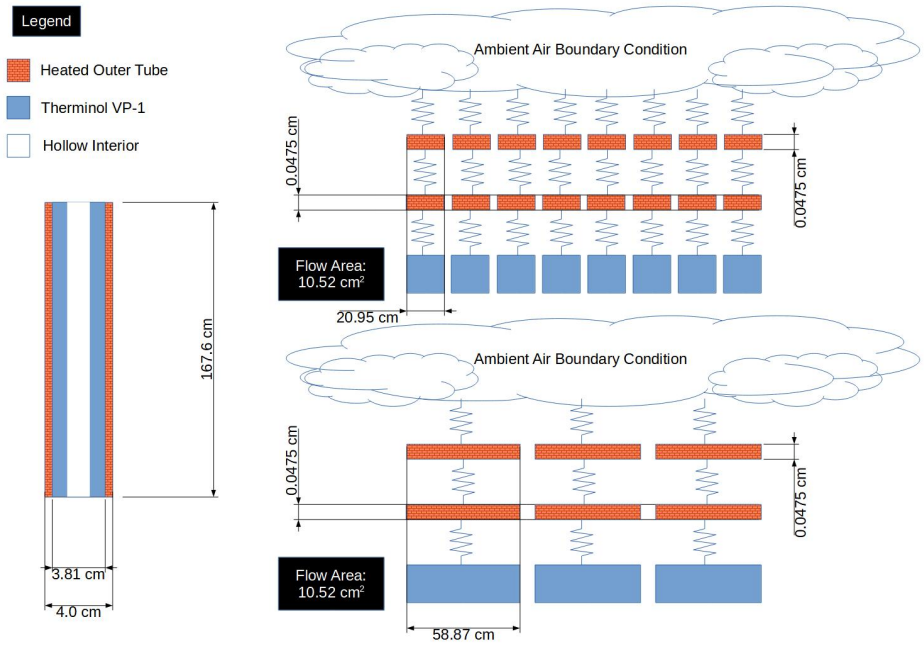


Figure 3.22: Axial Mesh Coarsening for Phase One Optimisation (Not Drawn to Scale)

The principles for calculating the relevant thermal resistances are the same as that for the meshes for the base case shown in Figure 3.19. The only difference here is that the surface areas used to calculate thermal resistances are bigger because the nodes are longer in the axial direction as shown in Figure 3.22.

Given the meshes shown for each run, let us now consider the boundary conditions and test procedures for all runs in Table 3.7. Figure 3.23 shows the test procedure for all modelling iterations developed under phase one.

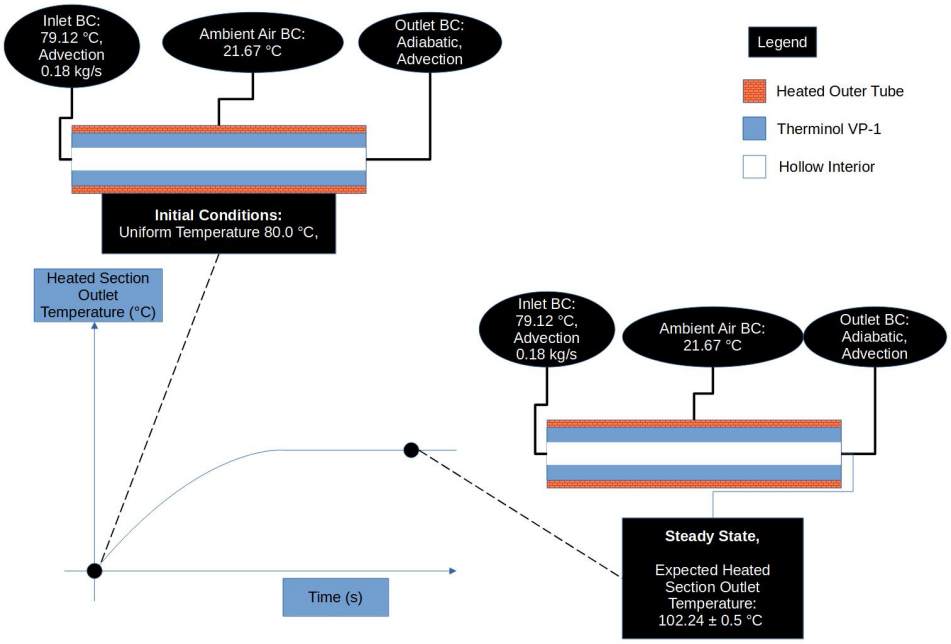


Figure 3.23: Phase One Test Procedure

Figure 3.23 shows that the phase one simulations started at a uniform initial temperature, and at t = 0 seconds, the heater was switched on. As a result, the heated section outlet temperature increased until it reached some steady state. The heater power distribution for the first three runs in Table 3.7 is shown in Figure 3.24:

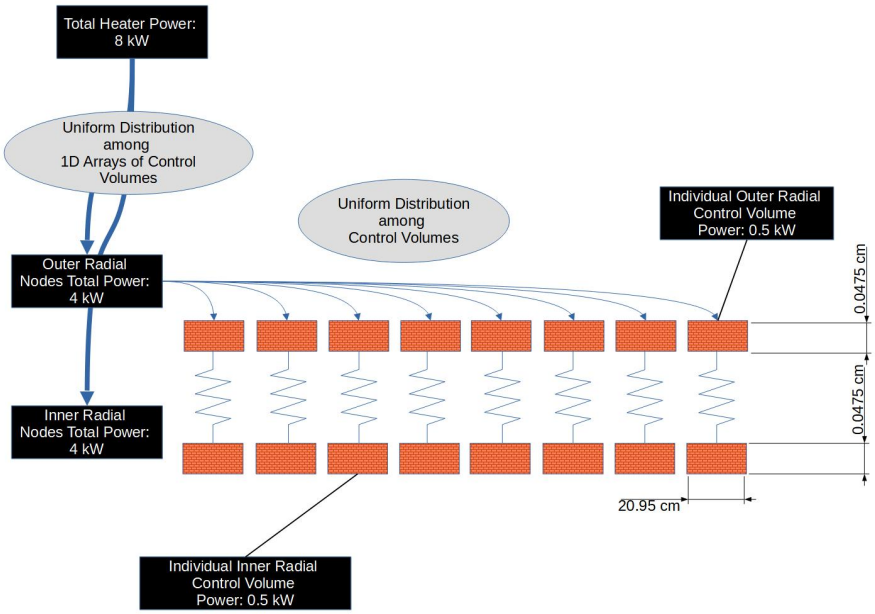


Figure 3.24: Power Distribution for Phase One Baseline Model (Not Drawn to Scale)

In essence, power is uniformly distributed across each control volume though the thermal masses of the control volumes are not the same. This is done for simplicity. The actual power distribution should be based on electrical resistivity of steel, but this is left for future work as described in the previous chapter. Therefore, I will be unable to reproduce surface temperature profiles using a uniform volumetric heat generation term for each node as described in this dissertation. Similar to the first three runs in Table 3.7, I determined the power distribution over all heated tube control volumes based on this same simplification. The resulting uniform power distribution for the axial and radial mesh coarsening tests is shown in Figure 3.25:

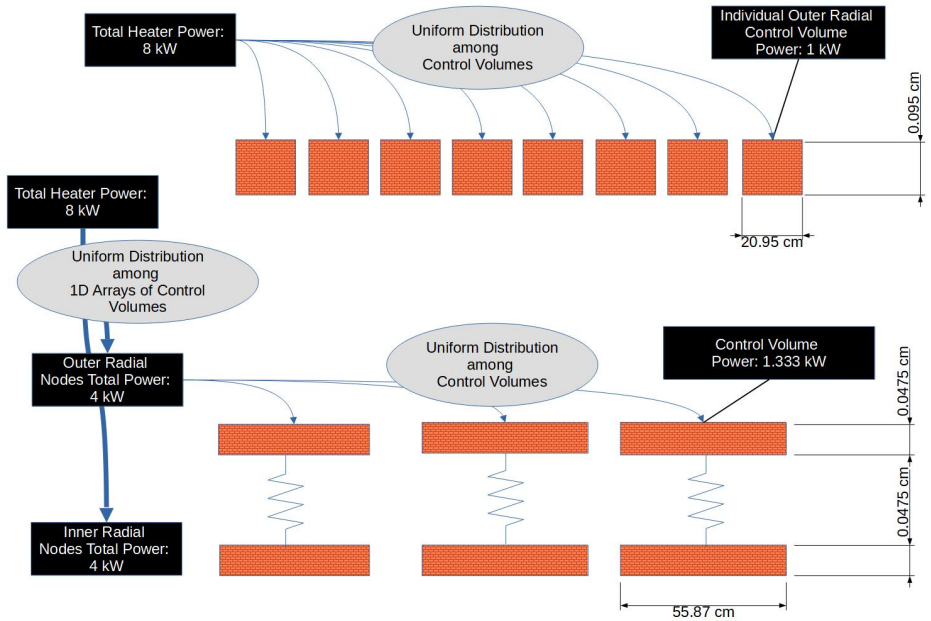


Figure 3.25: Power Distribution Diagrams after Mesh Coarsening (Not Drawn to Scale)

Hence, when the power was switched on at time = 0 seconds as shown in Figure 3.23, the reader should understand that power was distributed evenly among all the heated tube nodes as shown in Figure 3.24 and Figure 3.25.

As shown in Figure 3.23, the heater was left to run until the heated section outlet temperature reached steady state. Assuming the heat losses in the heater top head and MX-10 components are minimal, the heated section outlet temperature should be 102.2 $\pm 0.5°C$ given a heater power of 8 kW. If this temperature is reached, then the simulated model is deemed to be validated with experimental data. This should imply that the code is functioning as intended. Given the procedure shown in Figure 3.23, I timed the respective runs shown in Table 3.7 using at least 1000 calculation samples for each run to obtain an average calculation time.

To test the effects of mesh coarsening, I compared the steady state heated section outlet temperatures as shown in Figure 3.23 against the expected experimental data of 102.2 $\pm$ 0.5 $°C$. While there was no transient experimental data for the CIET Heater v2.0 Bare in isolation, I could use the transient response produced by the mesh in the baseline model shown in Figure 3.18 and observe its temperature prior to reaching steady state. This start-up transient is shown in Figure 3.23 in the sketch of the graph on the bottom left prior to reaching steady state. I then used this as a baseline transient response and compared the same start-up transients shown in Figure 3.23 for the two mesh coarsening cases in the fourth and fifth run of Table 3.7.

For the all test runs in phase one shown in Table 3.7, the coupling scheme between

each control volume was explicit because this was the simplest scheme to adopt for early prototypes. Nevertheless, information from test runs using explicit time marching schemes were used to inform the how test runs should be conducted for solvers with semi-implicit time marching schemes. Moreover, I could use the behaviour of the phase one models coupled using explicit time schemes as a basis of comparison for which later models developed in phase one should behave. This is known as a "regression test" in programming. Hence, I performed these regression tests during testing and development phase for the semi-implicit solvers. This was performed in the second part of phase one.

Based on the test runs in Table 3.7, I deemed that the radial mesh coarsening was the most suitable for this use case. The details are more thoroughly discussed in the results section. Using the radially coarsened mesh shown in Figure 3.20, I then proceeded to write semi-implicit solvers based on GeN-Foam Code [Robert et al., 2023].

In semi-implicit coupling scheme, I model the heater based on matrices and vectors rather than a individual control volumes to speed the simulation up. I then used the "ndarray" linear algebra libraries based on the "intel-mkl" libraries in order to solve the equations expeditiously. I also took Robert's GeN-Foam code in C++ (licensed under GNU GPL v3) [Robert et al., 2023], translated it into Rust and adapted it for matrix construction within the semi-implicitly solved 1D Array of Control Volumes. The solvers were based on the N-Dimensional linear algebra libraries mentioned earlier in the chapter. These 1D arrays of semi-implicitly solved control volumes are known as "ArrayCV" objects within the thermal hydraulics library. The semi-implicit equations that ArrayCV objects solve internally were already presented in the previous chapter as:

$$
\begin{aligned}
m_{cv} c_p(T^t) \frac{T^{t+\Delta t} - T^t}{\Delta t} = & \sum_j^N H^t_{thermal,self \ \leftrightarrow \ j,i}(-T^{t+\Delta t}_i + T^t_{j,i}) \\
& + \dot{m}^t(-h^t_{enthalpy,i} + h^t_{enthalpy,i-1}) \\
& + H^t_{avg,axial}(-T^{t+\Delta t}_i + T^{t+\Delta t}_{i-1,i}) \\
& + H^t_{avg,axial \ partitioned}(-T^{t+\Delta t}_i + T^t_{i+1,i}) \\
& + \sum Q^t_{gen,i} + \sum Q^t_{boundary \ conditions,i}
\end{aligned}
$$

Constructing "ArrayCV" objects rather than using several "SingleCV" objects to represent a 1D array of control volumes was done for several reasons. Firstly, I hoped to increase calculation speed using optimised linear algebra libraries. Secondly, when I construct "ArrayCV" objects, I designed it to abstract away the complexity of constructing the control volumes. Thirdly, the time marching scheme in the axial direction would be semi-implicit, therefore there should be added benefits for numerical stability for heat transfer calculations in the axial direction.

ArrayCV objects are based on the principle of partitioned meshes described in the last chapter. Internally, they are coupled in a semi-implicit fashion. When coupled externally to other HeatTransferEntity objects, they are coupled explicitly. Again, to couple the ArrayCV

objects explicitly to other HeatTransferEntity objects, "ArrayCV" objects would contain at least two control volumes, one at each tail end of the array. The user would then specify the number of "inner control volumes" the array would have in addition to the two at the peripheries. When coupling ArrayCV objects to other heat transfer entities in the axial directions, I would actually couple the SingleCV objects within these ArrayCVs to the other HeatTransferEntity. Therefore, code used to couple SingleCV objects explicitly to each other is re-used in ArrayCV objects as shown in Figure 3.3. If lateral coupling of ArrayCV objects was done, then this would contribute to the $\sum Q^t_{boundary\ conditions,i}$ term for each control volume. Supposing the control volume i in two ArrayCV objects, ArrayCV 1 and ArrayCV 2 were coupled laterally for radial heat transfer, then the heat transfer from ArrayCV 2 to Array CV1 would be:

$$Q^t_{lateral\ heat\ transfer,i} = -H^t_i(T^t_{ArrayCV\ 1,i} - T^t_{ArrayCV\ 2,i})$$

This lateral heat transfer would be one of the terms contributing to $\sum Q^t_{boundary\ conditions,i}$ for each control volume within the ArrayCV. Thus, laterally coupling ArrayCV objects is done using an explicit time marching scheme.

As I was developing the ArrayCV objects based on the semi-implicit solver for phase one, I also used learnings from the earlier phase one tests in Table 3.7. One key learning point was that calculating thermal conductances was computationally expensive. To reduce computational costs, I decided to calculate an averaged lateral thermal conductance (or resistance) based on the arithmetic mean temperature of all the control volumes in the 1D array. Another motivating factor to use an averaged thermal conductance was that it was easier and faster to program than having individual thermal conductances used for each node in the radial direction. This averaged lateral thermal resistance was used to calculate the radial thermal resistances shown in Figure 3.21 for all control volumes coupled radially.

To expedite development, I also did not thoroughly record the impacts that these two changes had on the heated section outlet temperatures individually. This is because of my conjecture that using an averaged lateral (or radial) thermal resistance should not significantly impact heat transfer such that the outlet temperatures were significantly different. Hence, I only recorded and presented the steady state and transient heated section outlet temperature when both these changes were applied. I then subjected the resulting model to the same test Figure 3.23 and compared its steady state heated section outlet temperature to previous runs and experimental data. This concluded phase one of the optimisation tests.

Optimisation Phase Two Techniques and Procedures In phase one, I found a suitable mesh, developed the "ArrayCV" object which uses a semi-implicit solver internally and decided to use an averaged lateral (radial) conductance to couple "ArrayCV" objects using an explicit time scheme. The timing results were deemed to be acceptable and I moved on to phase two, where I would add the phase two components as shown in Figure 3.17.

In phase two, all components were modelled according to their SAM/RELAP5 dimensions. The key differences in heater modelling between phase one and phase two are shown in Figure 3.26:

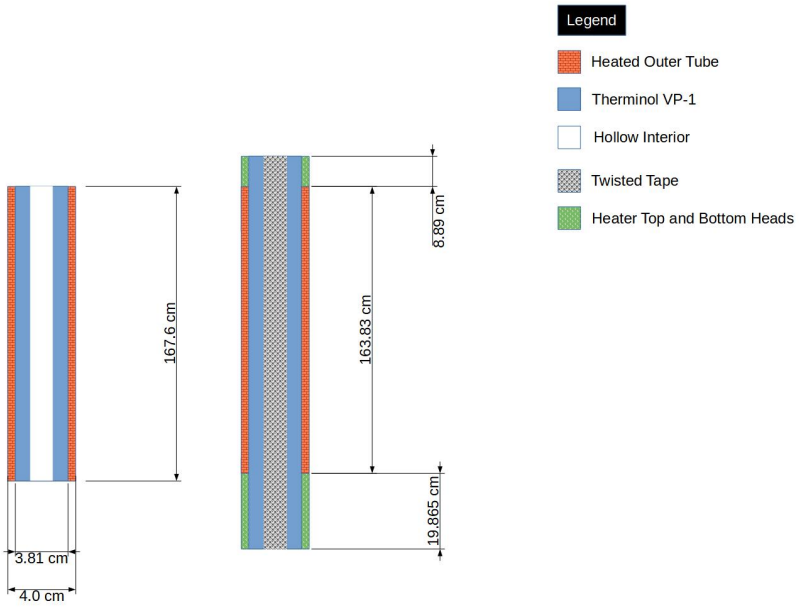


Figure 3.26: Heated Sections for Phase One and Phase Two Optimisation Tests

For phase two, the dimensions of the components and support structures are are shown in Figure 3.27:

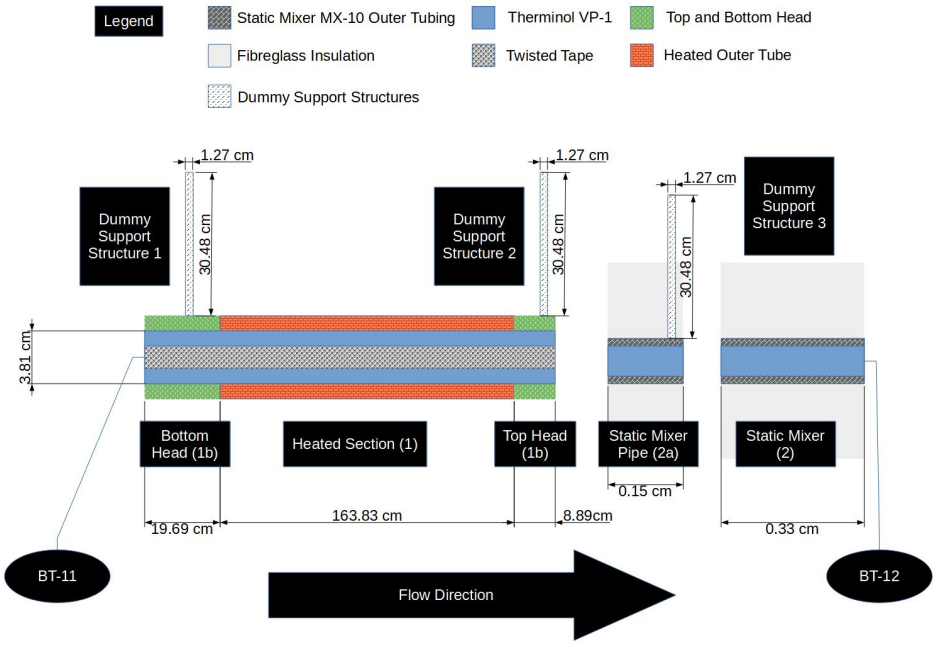


Figure 3.27: Components for Phase Two Optimisation Tests

The explanation of the component dimensions, as well as the Nu correlations for estimating the heat transfer coefficient between twisted tape and Therminol VP-1 can be found in the previous chapter. The thickness of the fibreglass shown in Figure 3.27 is 2 inches, while the MX-10 pipe thickness is 0.277 cm as discussed in the last chapter. The convective thermal resistance to air is determined by the heat transfer coefficient to air of 20 $W/(m^2 \cdot K)$.

For the phase two nodalisation of these components, one can refer to Figure 3.28:

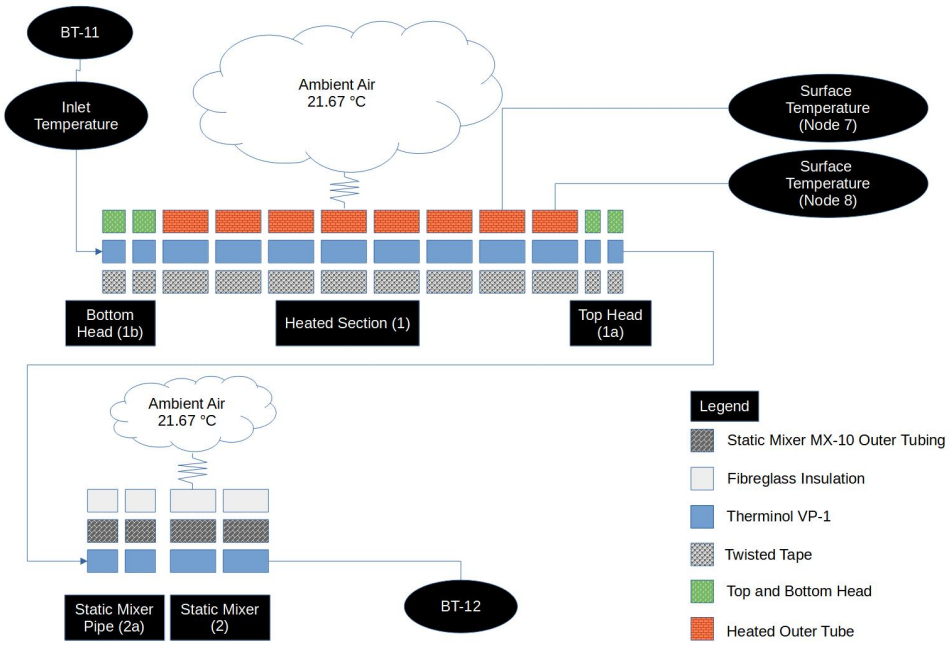


Figure 3.28: CIET Heater v2.0 Bare and MX-10 Nodalisation (Not drawn to Scale)

Figure 3.28 shows the nodalisation scheme and boundary conditions for all fluid components in phase two. Fluid shall now enter from the inlet (BT-11) at the bottom of the heater bottom head (1b), flow through the heated section, exit the top head (1a), enter the static mixer pipe (2a) and static mixer (2) before exiting the system. There is no axial conduction between the fluid components in Figure 3.28, and the boundary conditions for conduction are adiabatic. However, there is advection to allow fluid to carry enthalpy in at the inlet (temperature measured by BT-11) and carry enthalpy out at the outlet (temperature measured by BT-12).

For the sake of simulating the computational burden due to conduction in the support structures shown , I simulated the three support structures using two equally sized control volumes each. All three support structures are identical in terms of dimensions. They are represented as 0.5 inch diameter cylinders one foot long as explained in the previous chapter. I then coupled them to the heater top, bottom head and the steel pipe of the MX-10 static mixer pipe as shown in Figure 3.27. The first support structure was coupled to the heater bottom head outer pipe control volume adjacent to Node 1 of the heated section as seen in Figure 3.28. The other end of the first support structure was coupled to the ambient air boundary condition through a convective thermal resistance determined by h = $20\ W/(m^2 \cdot K)$. The second support structure was coupled to the metallic pipe control volume of the heater top head (1a), this is the control volume in the top head furthest from Node 8 in Figure 3.28. The last support structure was connected to the Static Mixer MX-10 Outer Tubing of the Static Mixer Pipe (2a) in Figure 3.28, the control volume is the one adjacent

to Static Mixer (2) in Figure 3.28. The other ends of the remaining two support structures were coupled to ambient air boundary conditions similar to the first support structure.

These three support structures are coupled to their respective components with the thermal resistance of 0.5 foot of the 0.5 inch diameter cylinders to simulate the time taken to couple these components thermally to each other. This was done for the sake of estimating computational burden of the support structures. Additionally, in timed runs, I also used this same thermal resistance based on the thermal resistance of 0.5 foot of the 0.5 inch diameter cylinders to couple the heated outer tube of the CIET Heater to their adjacent top and bottom heads. I also used this same thermal resistance to couple the twisted tape within the heated section to the twisted tape within the top and bottom heads. This was originally done for other reasons at first[2], but it became useful for simulating some of the computational burden required for coupling adjacent structures via axial conduction.

Hence, for the purpose of time trials, the support structures and heater components were coupled to each other. A dummy thermal conductance was also used to couple the heated section to the top and bottom heads. However, for the purposes of validation, the support structures and heater components were decoupled from each other, and the axial conduction between adjacent components was switched off.

As shown in Figure 3.28, the mesh adopted is one where the radial mesh is coarsened. This meshing scheme is fixed for all test runs in phase two.

In phase two, I attempted further code optimisation to bring the calculation time for the seven components down to about 1ms for 15 ms of simulation time. I tried using Callgrind and flamegraphs (I installed "cargo-flamegraph") for optimisation. These would help me pinpoint which sections of the code were bottlenecks. Figure 3.29 shows an example of one of these flamegraphs:

[2]I originally did this to investigate I could calibrate the support structures such that they could account for parasitic heat losses. I later gave up this endeavour.

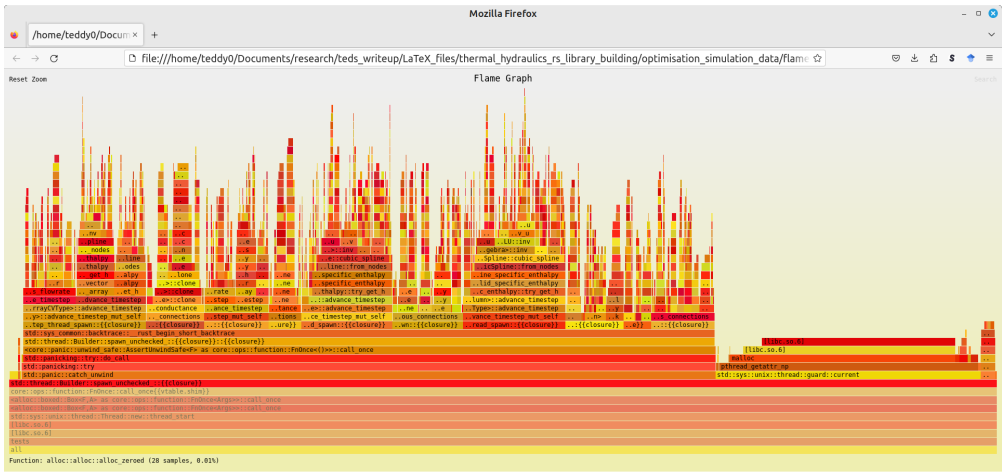


Figure 3.29: Example of Flamegraph

Using the "cargo-flamegraph" package, I generated a flamegraph and this indicated to me that there two main bottlenecks. Firstly, the SingleCVs contained within "ArrayCV" objects had to have their temperatures calculated iteratively from their current specific enthalpy. The temperatures were required whenever thermophysical and thermodynamic properties were to be obtained. Therefore, the function for iteratively obtaining temperature from enthalpy was executed several times in one time step, and thus it was computationally expensive. The second bottleneck was specifically the construction of cubic splines in order to interpolate the thermophysical properties of steel. This was also computationally expensive.

Besides these bottlenecks, I also found that the automatically calculated time step remained roughly constant throughout the simulation. From phase one optimisation, I found that the process of calculating time steps took a significant amount of computational resources. Hence, I opted to use a constant time step instead to save on computation cost. I tested using an automatic time step of 10 ms and 15 ms and found that these time steps were suitable for a numerically stable calculation given the mesh in Figure 3.28. Based on these findings, I made changes to several portions of code to improve the calculation times in phase two. Again, about 1000 or more calculation samples were taken and the average calculation time for these samples were found.

For optimisation phase two, I used only one test case for these timed trials. The test case was again based on CIET Heater v2.0 Bare at inlet temperature of 79.12 $°C$ and 8 kW of heater power. Figure 3.30 shows the test procedure for phase two:

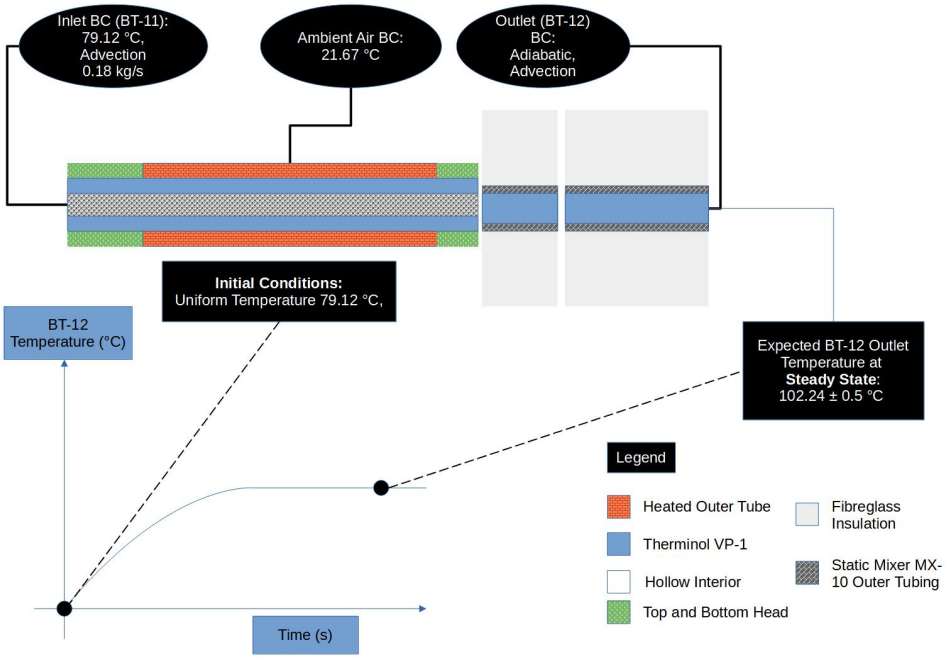


Figure 3.30: Test Procedure for Optimisation Phase Two

As shown in Figure 3.30, the test procedure is quite similar to phase one except that the initial temperature is the same as the inlet temperature. This was done out of convenience as there were several numbers present in the code that made debugging cumbersome. Having one less number to keep track of made coding easier. Other than this detail, and the fact that more components were added into the phase two optimisation, the procedure was quite similar to Figure 3.23. Once phase two optimisation was complete, the model developed was then validated using experimental data. This is described in the validation portion later in this chapter.

Results for Meshing and Optimisation

Optimisation Results for Phase One The initial results for phase one optimisation are provided in Table 3.8:

Timed Trial	Total Timestep Calculation Time (ms) $\mu + 2\sigma$	Pointer Dereferencing and Mutex Locks (ms)	Node Connection (ms)	Data Recording and Timestep Computation (ms)	Timestep Advance (ms)
Debug	602± 5.2	0	402	139	61.5
Release	54.2± 6.8	0	36.7	12.3	5.17
Release and Parallelised Node Connections	33± 3	0	16	12	5.02
Release and One Radial Steel Node	32.1± 3.2	0	24.1	6.03	2.02
Release and Three Axial Nodes	19.6± 7.8	0	13.4	4.18	2.03

Table 3.8: Initial Timed Trials for CIET Heater v2.0 using Singular Control volumes

Table 3.8 shows the times required at each stage of the calculation. The total computation time taken for each time step was broken down into four steps.

The first step was the Atomically Reference Counted (Arc) Pointer dereferencing to access data. The Arc Pointer is a type of smart pointer within Rust which allows multiple threads to access the same piece of data allocated on the heap memory. Mutually exclusive (Mutex) locks prevent two threads from changing the same piece of data at the same time, thus preventing some data races when during parallel computation. The time required for dereferencing these Arc pointers and obtaining the Mutex locks was negligible.

The second step was Node connection in Table 3.8 refers to calculating the energy transfer between two control volumes. In Table 3.8 specifically, we are using an explicit time marching scheme. This includes calculating thermal resistance between adjacent control volumes and calculating enthalpy transfer due to advection. This seemed to take the most significant time out of all the steps.

The third step in Table 3.8 was for data recording and time step computation. Time step computation is where a stability time scale for all control volumes were determined based on the methodology prescribed in the last chapter. The smallest of these time scales is then taken to be the time step for the next calculation. Additionally, the data for each time step

was also written to a csv file. This is the data recording step.

The fourth step in Table 3.8 is the "Timestep Advance" step. This is where the enthalpy changes for each control volume are summed up and used to calculate the enthalpy of the control volume at the next time step. Several other miscellaneous operations are carried out to clean up redundant data.

These timings were deduced from the "SystemTime" functions within Rust. Based on Table 3.8, I saw that optimised compilation is a must for enabling real-time calculations as the unoptimised compilation speeds were far too slow. An estimated speed up of 10 times was observed when using the "Release" compilation mode in Rust.

Since the node connection step took the most time, I decided to introduce parallelised calculation using 8 parallel threads from the "std::thread" libraries to speed up the node connection step[3]. This had limited success as the computation speed was only reduced by roughly 55%. From this test run, I determined that parallelisation was not sufficient for producing the necessary speed up needed for real-time computation. Therefore I decided to coarsen the mesh.

The fourth run in Table 3.8 involved mesh coarsening in the radial direction. This had comparable timing results to the parallel run even though I only used one thread. Additionally, mesh coarsening in the radial direction enabled me to use a larger time step for computation. The baseline model in the "Release" had an auto calculated time step of 2.3 ms. However, after mesh coarsening in the radial direction, the auto calculated time step became 13 ms. Hence, radial mesh coarsening for the heated steel tube from two radial nodes to one radial nodes helped me reduce computational burden by about 40% and the computational requirement by about 500% compared to the "Release" timed trial. This made mesh coarsening in the radial direction a very attractive proposition.

The fifth run in Table 3.8 involved mesh coarsening in the axial direction as described earlier. While the calculation time decreased significantly, the auto calculated time step remained at 2.3 ms.

Of course, for mesh coarsening in the fourth and fifth runs in Table 3.8 one may be concerned about accuracy loss. Therefore, we can ascertain if the losses are acceptable by comparing steady state data of the radially and axially coarsened timed trials, "Release and One Radial Steel Node" and "Release and Three Axial Nodes" in Table 3.8 respectively, to the steady state and transient response from the timed runs based on the mesh the "Release" and "Debug" runs. Again, the first three runs in Table 3.8, the "Debug", "Release" and "Release and Parallelised Node Connections" timed trials all use the same mesh as discussed earlier in the chapter. Hence, they produced identical results.

The results for the simulations based on these different nodalisation schemes is presented in Figure 3.31:

[3]I did not want to use the Rayon crate as I wanted to reduce the number of dependencies for my library

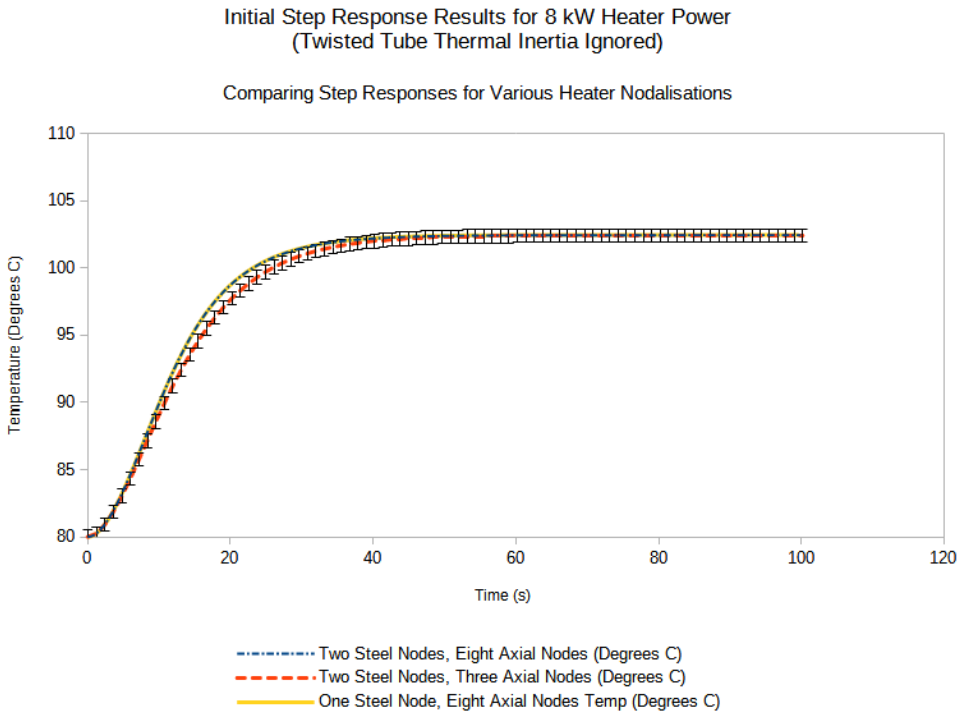


Figure 3.31: CIET Heater v2.0 Initial Step Response Tests with Thermal Inertia of Internal Twisted Tube Ignored

Figure 3.31 shows that regardless of nodalisation scheme, the steady state temperature for all the tests was 102.43 $\pm$ 0.5 $^{\circ}C$. The differences were at most $\pm$ 0.02 K between the different test runs, much smaller than the measurement error of $\pm$ 0.5 K. Hence, the mesh coarsening did not significantly affect the steady state heated section outlet temperatures. For the transient start-up response shown in Figure 3.31, the radial coarsening method (One Steel Node, Eight Axial Node) produced identical results to that produced by the original mesh. However, the axial mesh coarsening method (Two Steel Nodes, Three Axial Nodes) the heated section outlet temperature seems to change more slowly as compared to that in the original mesh. This exceeds the measurement of $\pm$ 0.5 K at t $\approx$ 20 seconds. From Figure 3.31, I deduced that radial mesh coarsening resulted in an acceptable fidelity loss. Therefore, I decided on the radially coarsened mesh seen in Figure 3.20 for all prototypes from this point on.

The next step was to test and implement the semi-implicit solvers with averaged lateral (radial) thermal conductances to further expedite calculations. As mentioned in the results section, I made several improvements to speed up calculation times for this test run to expedite development. This included using semi-implicit solvers and averaged thermal conductances. Using an averaged thermal conduction would significantly help to reduce time required for the "node connection" step, whereas using semi-implicit solvers should improve the time required for the "advance timestep" part of the calculation where temperatures for

each control volume during the next time step are computed. I also used a constant time step rather than an auto calculated time step to further reduce on calculation time. The constant time step I used was 10 ms or 0.01s. This is because 13 ms was determined to be a suitable time step from the previous tests, and I just decided to round this to a round number. Implementing several improvements in one test did not allow me to thoroughly investigate the reasons as to why the speed improvements came about. I did not think this was important enough to study at the time of running the tests because I mostly cared about obtaining the end result of having a real-time digital twin with which to test the simulated neutronics feedback controller. Hence, I skipped past the systematic studies during phase one optimisation.

The simplification where averaged thermal conductances were used to calculate heat transfer in the radial direction warranted another test to ascertain whether this would adversely affect the transient response and steady state response of the heated section outlet temperature. Hence, I performed the same test procedure in outlined in Figure 3.23 to obtain Figure 3.32:

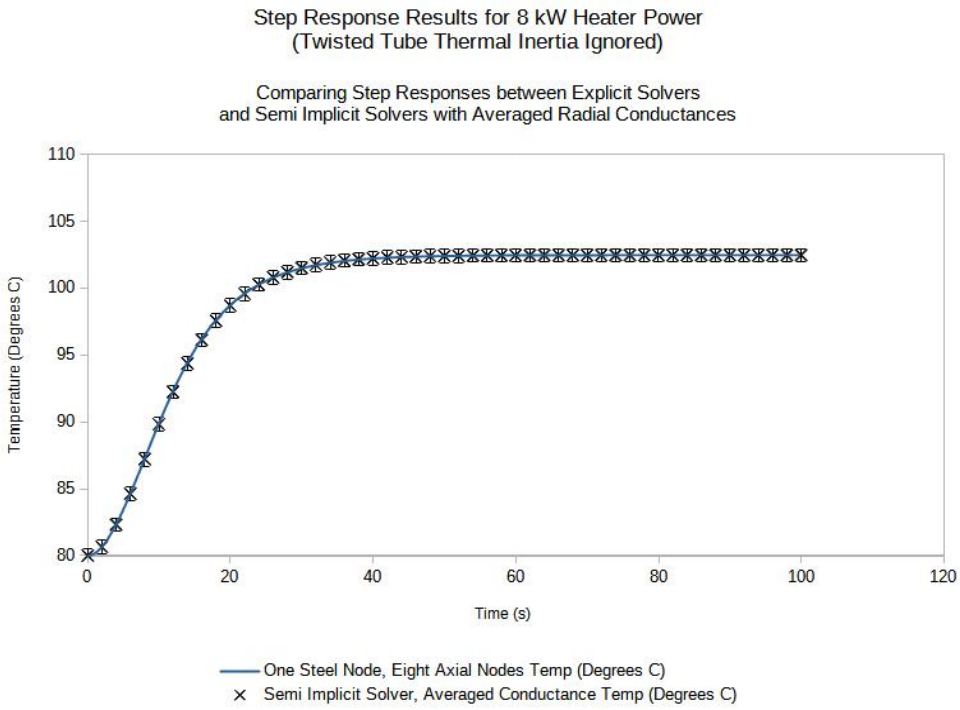


Figure 3.32: CIET Heater v2.0 Step Response Tests with Thermal Inertia of Internal Twisted Tube Ignored for Averaged Thermal Conductance Semi-Implicit Solver

Figure 3.32 superimposes the results from the semi implicit solver onto the results for the explicit solver using the radially coarsened mesh. The error bars of $\pm$ 0.5 K are representative of thermocouple measurement error. Figure 3.32 shows that there is no significant

difference in simulated transient outlet temperatures when using the semi-implicit solvers with an averaged conductance compared to an explicit solver with the radially coarsened mesh. Moreover, the steady state temperature reached was 102.46 $\pm$ 0.5 $^{\circ}C$, which means that the model could reproduce the steady state experimental data of 102.2 $\pm$ 0.5 $^{\circ}C$ to within thermocouple measurement error. This was a satisfactory result.

In terms of calculation time, the total time used to calculate each time step was about 0.23 ms. Therefore, there was a calculation speed improvement of about 100 times when applying these optimisation techniques. The node connection time was about 0.2016 $\pm$ 0.032 ms. This is mainly because the number of times that thermal conductances were calculated was reduced. Therefore, using one averaged thermal conductance to calculate the radial heat flux for all nodes along the 1D array of control volumes was an effective technique in reducing computation time. I found that even when using an averaged axial and radial thermal conductance for the ArrayCV, the steady state heated section outlet temperatures did not differ significantly from that of BT-12. Additionally, the step to solve for and update the temperature profile of the arrays, which is the "advance timestep" calculations step, took 0.01028 $\pm$ 0.0014 ms. This was about 200 times faster than the standard explicit control volume method of the same mesh. Thus, for this single heated section, parallelisation was not required to ensure that the calculations were fast enough.

Given these satisfactory results, I decided to move on to phase two of the optimisation process.

Optimisation Results for Phase Two In phase two, I added several components to the simulation as described in Figure 3.27 and Figure 3.17. These components were constructed using lessons learnt from phase one. This included using the semi-implicit solvers within the ArrayCV objects, using average thermal conductances, and also coarsening the mesh radially. I used these lessons to construct an initial model for phase two.

Initial iterations of the code using the semi-implicit solver were performed faster than real-time, (on a i7-10875H CPU) but each component took approximately 1-2 ms to calculate such that the total calculation time took approximately 10 ms. This was even after I used a fixed timestep of 10 ms so that we could do away with the automatic time step calculations. In these initial iterations, the extra computational burden of the new structures and extra time spent coupling structures thermally to each other caused the simulation to be slower than real time. These simulations took on the order of 10 ms, which was too slow for real-time computation. I increased the time step as much as possible, but it seemed that 15 ms was a practical upper limit before instability problems plagued the solver. Additionally, I used parallel computation to reduce the computation time to about 3 to 5 ms. However, this was still short of the 1ms goal which I had for every 15 ms of computation time. To achieve real-time simulation capability, I had to make the code even faster. Hence, I had to use the flamegraph tool to identify which parts of code caused the biggest calculation slowdowns.

Based on these flamegraphs, I identified that in the Single Control Volume (SingleCV) objects, the algorithm for obtaining thermodynamic properties was slowing the calculations.

For the SingleCV objects, their thermodynamic state was stored in the form of an enthalpy $h_{enthalpy}(T)$. Whenever thermodynamic properties were requested, I would obtain the control volume temperature T iteratively from $h_{enthalpy}(T)$ using the Brent-Dekker algorithm. If the properties were requested five times, then I would recalculate T five times iteratively. To optimise the code, I decided to add a field for the current temperature of the SingleCV. This was added so that I could calculate the temperature of the SingleCV at each time step once, and store it in that field. At every timestep advancement step, the temperature of each SingleCV object is obtained from the specific enthalpy only once and then stored in the object using this field. Consequently, whenever thermophysical properties are required, I could just use the temperature from the SingleCV object at little or no computational cost. These results imply that storing the thermodynamic state of the control volume as a temperature may be computationally cheaper than storing it as an enthalpy.

For my SingleCV objects which use explicit coupling, I stored their thermodynamic state using enthalpy. However, when using the semi-implicit solvers within ArrayCVs, I store the thermodynamic state of the object using temperature. This may have contributed to the observed speed up when using the semi-implicit solver rather than the explicit solver during phase one optimisation. Hence, the speed up is not solely due to using optimised linear algebra libraries such as OpenBLAS and intel-mkl, but also due to the fact that I stored the thermodynamic state of the control volume using temperature as opposed to enthalpy. While I did not quantify the calculation time savings that semi-implicit schemes alone had over explicit coupling schemes, the added benefit of stability and the ease of instantiating ArrayCVs of various lengths meant that I kept using them for all future prototypes of the model.

Secondly, I found that accessing the thermodynamic properties of steel was computationally expensive. This is because I stored the thermodynamic properties of steel as tabulated data based on the data used by RELAP5 and SAM models [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015]. To obtain thermal conductivity k for example, I would use the tabulated k data to construct a cubic spline first, and then interpolate k at the desired temperature using the cubic spline object. After the calculation was complete, the cubic spline was deleted. Therefore, every time that thermodynamic properties of steel were requested, this cubic spline had to be constructed. To speed up the code, I used steel thermophysical properties from Grave's work [Graves et al., 1991] which was already in polynomial form as opposed to using cubic splines. While this This is computationally cheaper to calculate compared to constructing a cubic spline from data every time the thermophysical property calculation function is accessed. Both of these changes sped up the code significantly such that the heat transfer calculations for seven components could be computed on the order of 1 ms to caluclate a timestep of about 15 ms. Thus, we have ample room computation time for more computational burden. Hence, we could fit as many as seventy components and still be within real-time. This shows that the code profiling and optimisation with "cargo-flamegraph" was quite successful.

When it comes to accuracy, the results after phase two optimisation were quite acceptable. The heater outlet temperatures were around $102.41 \pm 0.5\ ^{\circ}C$ at steady state, which is

indistinguishable from the $102.2 \pm 0.5°C$ recorded in experimental runs for heater power of 8 kW. I did not do transient testing comparisons during phase two because the model developed after phase two was more or less ready for validation. Hence, the transient tests would be done during model validation rather than phase two optimisation.

Validation using Steady State Tests At the end of phase two optimisation, I arrived at a suitable model where I obtained 1 ms of calculation speed for every 15 ms of simulated time. In these simulations, 8 computation threads were used. For the steady state validation tests, I thermally decoupled the structural supports from the main components and also decoupled the dummy thermal conductances between the heated section of the heater and its top head and its bottom head.

I then ran steady state tests using a process similar to that shown in Figure 3.30. Figure 3.33 shows the steady state validation test procedure:

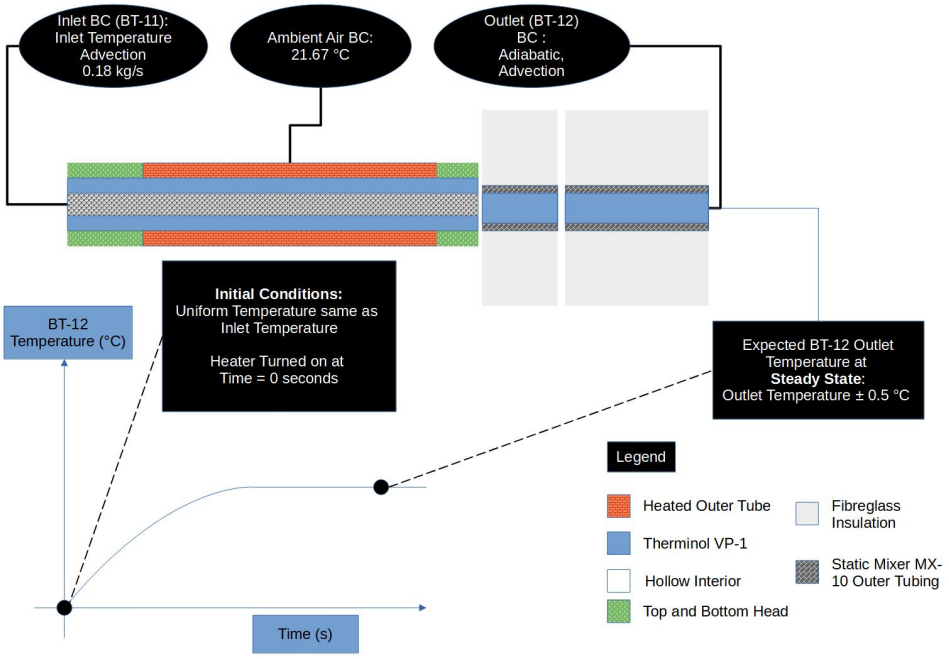


Figure 3.33: CIET Heater v2.0 Bare Steady State Validation Test Procedure

I used these same inlet temperatures as the inlet boundary condition for my test runs. For these runs, the initial temperature of all the nodes in all components is set to be same as the inlet temperature. The heater power was set to the values in Table 3.5. The simulation was allowed to run until the heater outlet temperature (BT-12) reached a steady state. I left the tests to run up till 300 seconds of simulated time, but based on these tests, 100 seconds would have been enough for this simulated heated to reach steady state. The results from the simulation runs were compared to experimental data in Table 3.5.

Validation using Transient Tests For this validation effort, I run am subjecting the transfer function to two tests. Firstly, a step increase of 500 watts. Secondly a step decrease of 500 watts. These are separate tests, and they are not run one after the other. Figure 3.34 shows the transient validation test procedure used to obtain data for the transient tests:

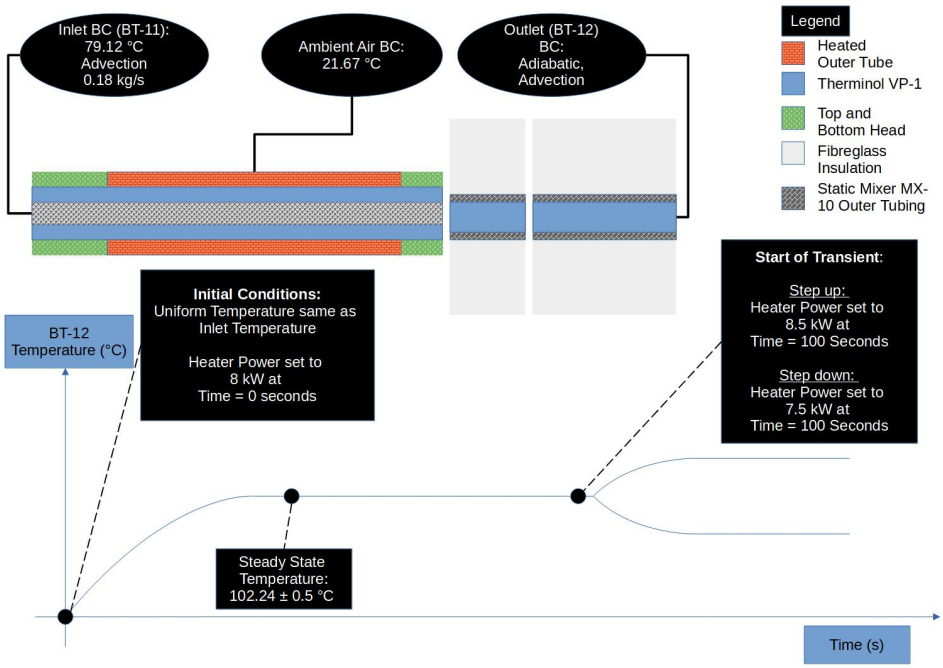


Figure 3.34: CIET Heater v2.0 Bare Transient Validation Test Procedure

As shown in Figure 3.34, the CIET Heater v2.0 Bare and the MX-10 piping components are set to a uniform inlet temperature equal to the inlet temperature of 79.12 $°C$ shown in Table 3.5 for the 8 kW test. The heater power was set to 8 kW at t = 0 seconds and the BT-12 temperature was allowed to settle at a steady state of approximately 102.2 ± 0.5 $°C$. This took approximately 60 seconds. The system was then allowed to further rest at steady state for another 40 seconds of simulated time because I wanted to start the transient at a round number of 100 seconds. At 100 seconds, the transient started, and heater was brought to 8.5 kW and 7.5 kW for the two step input tests respectively. The resulting simulated data was then compared to that shown by the empirical transfer function in Equation 3.65.

The empirical transfer function itself records temperature deviations given deviations in heater power from 8 kW. Hence a step input of ± 500 Watts was given as input to this transfer function at t = 0 seconds. 500 watts was given for the step up test, and -500 watts was given for the step down test. The resulting temperature deviations of BT-12 are over time are recorded and then added to the steady state outlet temperature (BT-12) of the simulated CIET Heater v2.0 Bare. At the time of step response tests, this simulated CIET Heater BT-12 temperature would have already been matched to the experimental steady

state BT-12 temperatures in Table 3.5 to within thermocouple measurement error of ± 0.5 K. Hence, whether I add the resulting temperatures to the experimental steady state values (102.2 ± 0.5 $°C$) or simulated steady state values (102.41 ± 0.5 $°C$) should make little difference.

With these simulated temperatures from the transfer function, I would then validate the transient response of the simulated BT-12 temperature in response to the same 500 watts increase or decrease. This would correspond to data recorded after t = 100 seconds. However, the transient for the transfer function started at t = 0 seconds. I merely added 100 seconds to the time coordinate for all the data points generated by the transfer function to make this data directly comparable to that of the simulation.

As mentioned before, the transfer function Equation 3.65 would measure the heater power to heater outlet temperature based on a full loop experiment with thermal pulses traversing the loop as described in the previous chapter. Therefore, only the short term response, roughly one minute after the start time of the transient, will be used for validation.

Results for Validation

Validation using Steady State Tests The simulated steady state temperatures obtained using the steady state validation test procedure in Figure 3.33 is presented in Table 3.9:

Heater Power (watts)	Heater Inlet Temp BT-11 ($°C$)	Experimental BT-12 ($°C$)	Simulated BT-12 ($°C$)
3000	78.75	86.93	87.11
4000	79.00	90.25	90.36
6000	79.40	96.50	96.74
8000	79.12	102.20	102.41
10000	78.90	107.75	108.09

Table 3.9: Results used for Steady State Validation Tests, Experimental Data used is from CIET Heater v2.0 Bare Tests [De Wet and Per F Peterson, 2020]

Based on Table 3.9, the simulated BT-12 data is in good agreement with the experimental BT-12 data to within thermocouple measurement error of ± 0.5 K. Data from Table 3.9 is presented in Figure 3.35:

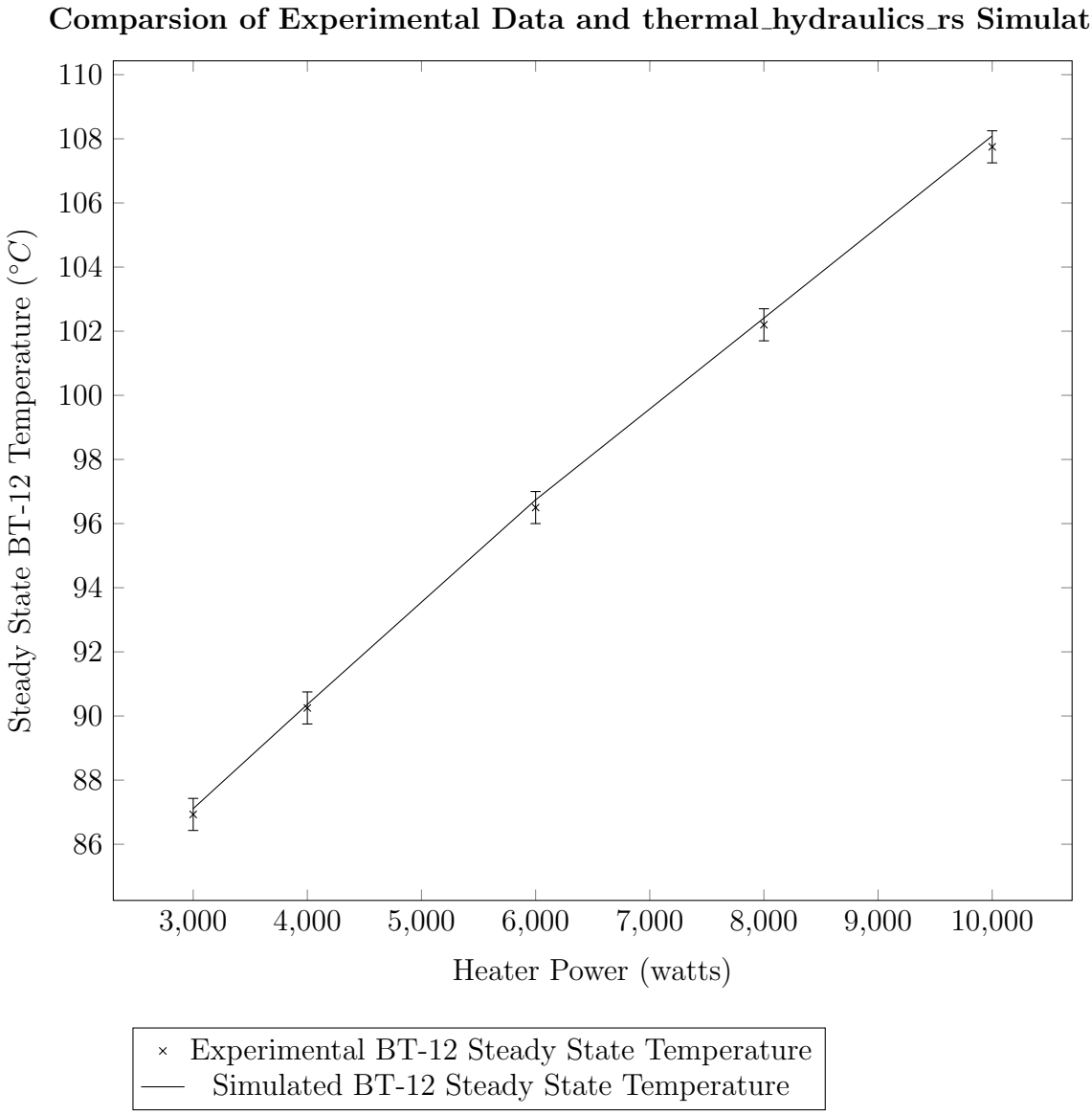


Figure 3.35: CIET Heater v2.0 Bare Steady State Forced Circulation Validation Test

Figure 3.35 shows that the steady state BT-12 outlet temperatures are in good agreement with experimental data to within the error bar of ± 0.5 K. Again, ± 0.5 K is the thermocouple measurement error typical of Type T thermocouples found within CIET [Zweibaum, J E Bickel, et al., 2015]. It is apparent that the many simplifications made for when constructing this model of CIET Heater v2.0 did not significantly impact its ability to replicate the steady state data. This includes ignoring the wall viscosity correction factor for the surface temperature within the Heater Nusselt Number correlation present in literature [De wet, Per F. Peterson, and Greenwood, 2019], using simplified methods to estimate entrance effects, using the same radial thickness to calculate thermal resistances for centre to inner surface

and centre to outer surface for the cylindrical control volumes (ignoring curvature effects as mentioned earlier), as well as using Gnielinski's correlations for the heater top and bottom heads. Other simplifying assumptions were used as mentioned in earlier parts of the chapter, but I will not list them all for the sake of brevity.

Also, another point of interest is that the temperature of the outlet of the heated section at steady state was only about 0.01 K to 0.02 K hotter than the BT-12 temperature in the 8 kW test run. This means that the heated section outlet temperature and the BT-12 temperature did not differ significantly in comparison to thermocouple error. This reinforces the assumptions used in phase one optimisation described earlier in the chapter.

Validation using Transient Tests For transient model validation I subjected the CIET Heater v2.0 Bare simulation to a 500 watts step increase as shown in the methods section and compared its BT-12 temperature against the transfer function data. The result is presented in Figure 3.36:

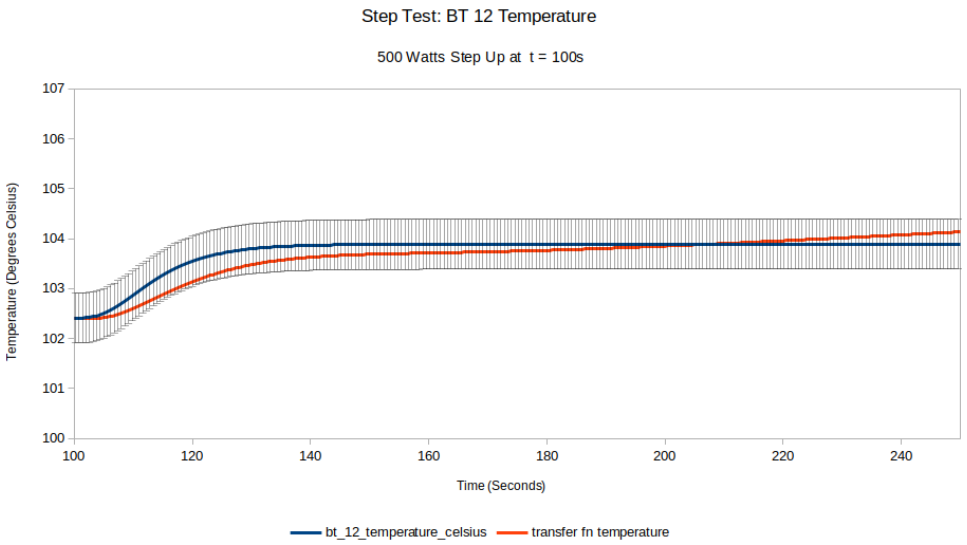


Figure 3.36: Comparison of Step Transient from 8 kW to 8.5 kW for Simulated Heater and Transfer Function [De Wet and Per F Peterson, 2020]

Figure 3.37 shows that the simulated heater behaviour, the "bt_12_temperature_celsius" plot agrees with the transfer function simulation, the "transfer fn temperature" plot to within ± 0.5 K of Type T thermocouples uncertainty typical of thermocouples found in CIET [Zweibaum, J E Bickel, et al., 2015]. For the first sixty seconds from the start of the transient at t = 100 seconds to t = 160 seconds, there is agreement between the simulation and the empirical transfer function to within thermocouple uncertainty. However, we observe that the empirical transfer function has a longer process time than the simulated BT-12 temperature. This is difference is smaller than the thermocouple measurement error but still observable within Figure 3.37. For the purposes of this dissertation, I consider this transient

validation sufficient for testing the simulated neutronics feedback controller. However, we can explore the extent to which modelling the additional thermal masses as described in De Wet's dissertation [De Wet and Per F Peterson, 2020] improves the agreement between the simulated BT-12 temperatures and the empirical transfer function in future work.

A similar good agreement between simulation and empirical transfer function can be found for the step down test of 500 watts from 8 kW to 7.5 kW. This is shown in Figure 3.37:

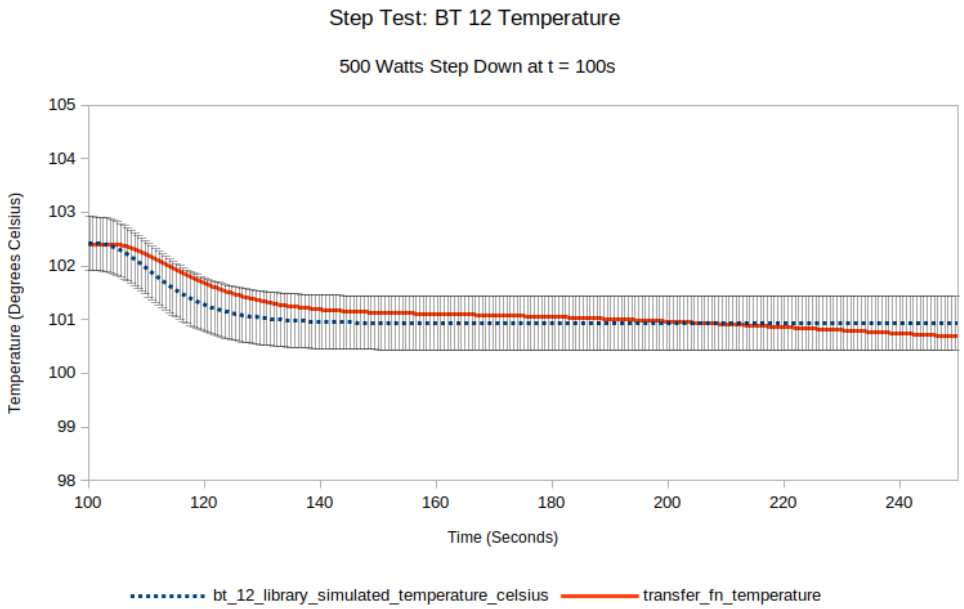


Figure 3.37: Comparison of Step Transient from 8 kW to 7.5 kW for Simulated Heater and Transfer Function [De Wet and Per F Peterson, 2020]

It is apparent that the many simplifications made for when constructing this model of CIET Heater v2.0 did not significantly impact its ability to replicate the transient data, at least when a 500 watt step input was used. However, it is apparent from Figure 3.37 and 3.37 that for larger step inputs (perhaps 1 kW or 2 kW), it is plausible that neglecting the thermal mass may cause the behaviour of the CIET Digital Twin to deviate from experiment beyond the threshold of thermocouple measurement error of $\pm$ 0.5 K. However, using larger step inputs for heater power would likely induce non-linear behaviour [De Wet and Per F Peterson, 2020] and complicate the analysis. Studying this is a subject for for future work. Also, while the deviations for the transient response of the BT-12 temperatures Figure 3.37 are smaller than $\pm$ 0.5 K, there is still a visible deviation for up to one minute after the transient started. I could model the additional thermal masses in the Transform Model to investigate if they reduce the deviation between model and experiment, but that is beyond the scope of this work.

Now, given that, at least for these transients, the transfer function and simulated heater agree to within thermocouple error, I can say that the optimised library has been successfully

validated for the purposes of this dissertation. This version v0.0.9 of the library is published as "thermal_hydraulics_rs" on the "crates.io" website for all Rust crates (packages).

3.4 Conclusion

Summary

In this chapter, we have successfully developed and tested a thermal hydraulics library written in the Rust programming language. This library was validated using analytical solutions and some steady state and transient forced convection flows in CIET Heater v2.0 Bare. We have also demonstrated that in terms of computational burden of the heat transfer and fluid mechanics portions of the code, it is indeed plausible that the library can simulate CIET in real-time.

Future Work

For now, we only wish to have a working and validated model of CIET's Heater so that we can use it to test and develop a simulated neutronics feedback controller. However, if one wants to investigate the effect of simulated neutronics feedback during a transient in CIET or, more specifically, CIET's Digital Twin, more work needs to be done in future. We could model CIET Heater v2.0 Bare in higher fidelity by modelling the additional thermal inertia present in the Transform model but absent in RELAP5/SAM models. We could also simulate the resistive heating effects in the heater so that the heat flux distribution is more accurate. Most importantly, for transient simulation, we need to construct a fully working version of CIET and validate it using experimental data. While we have ascertained that there is a good chance we will be able to simulate CIET in real-time, there may be unforeseen factors that add on to the calculation burdens at every time step such that we can no longer have a real-time simulation. In such a case, it is important that we have relaxed stability requirements for time stepping. For this purpose, it is important to have semi-implicit coupling as opposed to explicit coupling in the radial direction. In this case, we can assign a larger time step so that it becomes easier to achieve real time calculations.

Let us go into some of the possible areas of future work in more detail.

Resistive Heater Surface Temperature Modelling

In this work, we modelled CIET Heater v2.0 Bare based on a uniform user set heat generation term at each time step. This is where each equally sized control volume generates an equal amount of heat in the heated section. This was sufficient for the purposes of reproducing the BT-12 steady state data and the transient data.

However, CIET Heater v2.0 Bare is a resistance heater and its power distribution along the length of the heater is determined by the relative resistance at each section of the heater. As discussed in the literature review section, the resistive heater outer tube made of 304L

stainless steel has a temperature dependent resistivity $\rho_{\mu\Omega-cm}$ given as [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014]:

$$\rho_{\mu\Omega-cm}(T) = 0.0612 \cdot T(^{\circ}C) + 73.109 \tag{3.67}$$

Here, $\rho_{\mu\Omega-cm}(T)$ is the temperature dependent resistivity of steel with the units of $\mu\Omega - cm$. In Equation 3.67, the resistivity increases with temperature. Therefore, for the long heated tube present in the heated section of CIET Heater v2.0 Bare, the hotter parts of the heater have higher resistance than the cooler parts of the heater.

Given that the power produced by a section of the heater $P_{section} = I^2 R_{section}$, where I is the current flowing through the entire heated tube, and $R_{section}$ is the resistance of a section of the heated pipe, we should expect regions of higher resistance to have a higher power output. In other words, hotter parts of CIET's Heater should have a higher power output. This is quite different in contrast to the uniform heat generation profile we assumed earlier.

While this may seem trivial, especially for predicting BT-12 temperatures, having a validated heater surface temperature profile is particularly interesting from the perspective of implementing simulated neutronics feedback (SNF). In SNF, the fuel temperature directly impacts reactivity. While we do not have fuel in CIET, we do have a heating element. In previous experiments on CIET, the heater temperature was used to represent the fuel temperature in SNF [De Wet and Per F Peterson, 2020]. Of course, we may not get the thermal inertia of the pebbles and reflector structures scaled properly to the thermal inertia of the heater. Despite this, using heater temperature to represent fuel temperature may serve as a good starting point for implementing SNF based on fuel temperature. It is noteworthy that in those experiments, SNF was implemented using the Point Reactor Kinetics Equations (PRKE). In contrast, we are using a different methodology which will be discussed in the following chapters. Therefore, there was no explicit need to use the heater surface temperature profile. Nevertheless, it would be interesting to explore modelling the heater surface temperature profile in future should the need arise.

For validation of heater surface temperature profiles, there is also are additional sets of data available. This is the heater surface temperature profile [De Wet and Per F Peterson, 2020]. Heater surface temperature data is presented in the form of an empirical correlation at several heater power inputs. In Equation 3.68, the surface temperature of the heater would be correlated with the height at which the surface temperature was taken x [De Wet and Per F Peterson, 2020] at 8 kW:

$$T_{surface}(^{\circ}C) = -0.00016353x^3 + 0.013686x^2 + 0.32904x + 133.460 \text{ (x in inches)} \tag{3.68}$$

This data used to obtain Equation 3.68 was measured using four thermocouples on the heater surface at about 4 inches, 19 inches, 33 inches, and about 64 inches [De Wet and Per F Peterson, 2020]. Now, in CIET's original design, there were give thermocouples from bottom of the heater to its top, these are ST-10, ST-11, ST-12, ST-13, and ST-14 [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014]. Assuming these thermocouples are

evenly spaced, from each other, it appears ST-10 was placed at 4 inches from the bottom of the heated section, ST-11 at 19 inches, ST-12 at 33 inches and ST-14 at 64 inches. ST-13 should be placed at around 49 inches, but its data was not present in this dataset.

De Wet also presented a transfer function where deviations in the steady state heater surface temperature was taken as the output and deviations in heater power from 8 kW was the input [De Wet and Per F Peterson, 2020]:

$$G(s) = \frac{0.0002056}{s^2 + 0.3351s + 0.02822} \tag{3.69}$$

Temperature was measured in $^{\circ}C$ and power was measured in watts [De Wet and Per F Peterson, 2020]. The surface temperature was measured using thermocouple ST-11 [De Wet and Per F Peterson, 2020]. This is located closer to the bottom of the heater as ST-11 is one of five thermocouples attached to the heater [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014]. From bottom of the heater to its top, these are ST-10, ST-11, ST-12, ST-13, and ST-14 [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014]. ST-11 is most likely second from the bottom [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014], likely 19 inches from the start of the heated section.

Based on this empirical temperature profile in Equation 3.68, I did some preliminary studies on how the surface temperature profile would compare with the experimental correlation in Figure 3.38 during phase one optimisation:

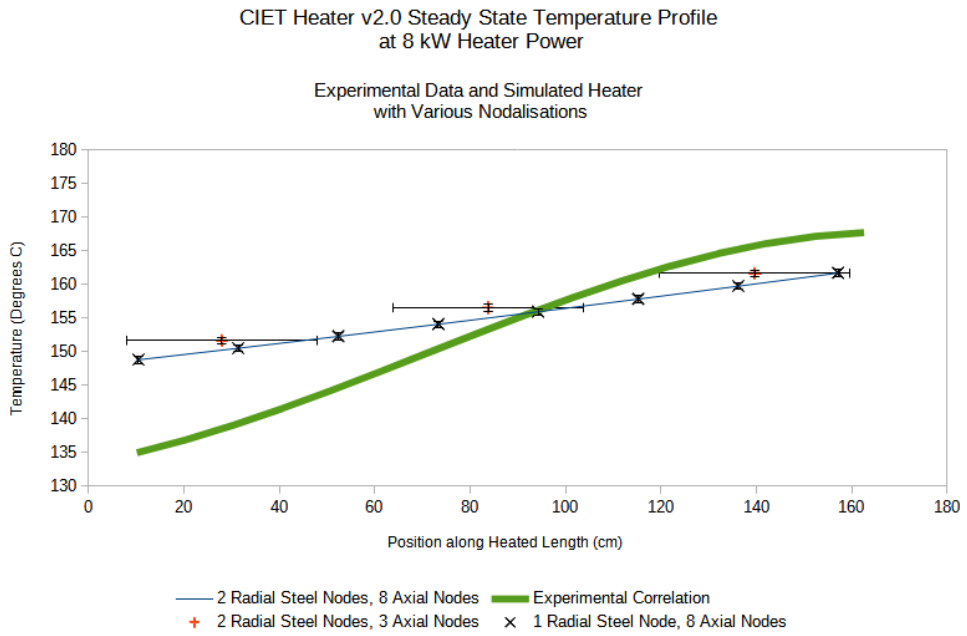


Figure 3.38: CIET Heater Surface Temperature Profile for Phase One Test Settings overlaid with Experimental Data [De Wet and Per F Peterson, 2020]

Unsurprisingly, I was not able to reproduce the experimental surface temperature profile. This did not change regardless of the nodalisation scheme. Based on our earlier analysis, we should expect the heat generation term to be higher in hotter regions of the heater. Therefore, the heat received by the fluid should be higher in hotter sections of the heater if we simulated the temperature dependent resistivity of steel properly as opposed to uniform heating along the heated pipe. While I did not record the exact heat transfer rates, I can perform a rough analysis assuming the thermal conductance is roughly equal throughout the whole heater tube. I would then use the temperature difference between the fluid and heated tube ΔT at the start and the end of the heater to estimate the relative power distribution along the heated tube.

For my simulation using uniform heat generation, the heated section temperature at the entrance is roughly 149 $^{\circ}C$, whereas the fluid temperature at the entrance is 79.12 $^{\circ}C$. Of course, the first fluid control volume temperature within the heated section used to calculate ΔT is slightly higher at about 82 $^{\circ}C$. Regardless, ΔT is about 70 K. At the exit of the heated section, the surface temperature is roughly 162 $^{\circ}C$ and the fluid temperature is 102.2 $^{\circ}C$. Therefore ΔT is roughly 60 K at the exit. While I expect ΔT at entrance and exit to be roughly equal when a uniform heat generation term is applied throughout the tube, the slightly lower ΔT at the exit seems to imply that heat flow to the fluid is lower at the exit as compared to the entrance. One possible explanation for this is that with a higher temperature at the exit, there is additional parasitic heat loss to the surrounding air as compared to the entrance. Therefore, for uniform heat generation among all control volumes, a larger portion of this heat generated gets lost to the air at the exit as compared to the entrance. Therefore, ΔT at the exit is lower than that at the entrance.

With the uniform heat generation case in mind, let us now get a sense of the heat generation profile using the experimental correlation. At the entrance of the heated section, the surface temperature is roughly 135 $^{\circ}C$. Given a fluid temperature of 79.12 $^{\circ}C$ at the entrance, ΔT is about 55.86 K. At the exit, the surface temperature is roughly 168 $^{\circ}C$, whereas the fluid temperature at the exit is 102.2 $^{\circ}C$. Hence ΔT at the exit is about 65 K. Now, we expect the heated section near the exit to lose more heat to the air as compared to the entrance as previously mentioned. Despite this, the ΔT at the exit is higher than that at entrance. This means that we not only lose more heat to air at the exit compared to the heated section entrance, we also transfer more heat to the Therminol VP-1 as compared to the entrance region. This strongly indicates that heat generation and therefore local heat flux at the exit is much higher than that at the entrance. This is expected behaviour because the heated section is indeed hotter at the exit than at the entrance region of the heated tube. Therefore, the power distribution is asymmetric. More of the power should be distributed at the hotter exit section of the heated tube rather than the entrance section. From this, we can infer that the expected behaviour due to temperature dependent thermal resistivity of steel is very likely present in the experimental data. Therefore, it is important to consider $\rho_{\mu\Omega-cm}(T)$ of steel if we wish to have a reasonably accurate surface temperature profile. This will, however, slow down calculations as it will add computational burden at each time step. This would be interesting to explore in future work.

Thermal Inertia and Support Structure Modelling

As mentioned earlier in the chapter, the thermal inertia and structural present in the Transform model are absent in the SAM/RELAP5 model and in this current prototype of the CIET Heater v2.0 Bare Digital Twin. We can account for these extra thermal masses and support structures in models developed in future work. We can then validate these forced circulation models using De Wet's Transfer Functions for various components around CIET [De Wet and Per F Peterson, 2020] and also check if the deviations from the empirical transfer function are smaller by performing the same tests used to obtain Figure 3.37 and Figure 3.36.

Semi-Implicit Coupling in the Radial Direction

To relax requirements on the largest allowable time step, we could enable semi-implicit coupling in the radial direction for pipe flows. This is because, at present, it is the bottleneck preventing us from using larger time steps. As mentioned earlier, implicit or semi-implicit coupling allows us to use larger time steps. This would be beneficial for performing calculations in real-time so long as matrices do not grow too large. This, however, would take a fair amount of development time given that one has to develop the algorithms for matrix construction and for retrieving elements for n number of nodes radially. For a 2D system, matrix indexing is much more complex than for a 1D system, and therefore we need a longer development time. Another issue to consider is that a semi-implicitly coupled system is tightly coupled both numerically and programmatically. This means that modifications to a tightly coupled system would often require rewrites to the entire matrix construction algorithm. For example, if I were to design an implicitly or semi-implicitly coupled code capable of constructing matrices for n radial nodes, I cannot then adapt this same code easily for a heat exchanger. In contrast, for a pipe represented by a 1D array of solid control volumes and a 1D array of fluid control volumes coupled explicitly to each other, I could convert it into a heat exchanger by coupling another 1D array of fluid control volumes to the 1D array of solid control volumes. I could also control the radial discretisation scheme by changing the number of arrays of solid control volumes or arrays of fluid control volumes. I could even neglect the thermal inertia of the solid control volumes by coupling the 1D fluid arrays to each other directly via some thermal resistance. This explicit coupling gives me a large degree of flexibility in constructing new components which I would then lose when using an implicit or semi-implicit coupling scheme. Given this state of affairs, one should judiciously consider which components really need tight semi-implicit coupling so that the coding and development effort is justified.

Natural Convection Validation

If we want to construct a full Digital Twin of CIET, we must validate the Digital Twin using natural convection data. This was done with previous RELAP models [Nicolas Zweibaum, 2015] and SAM models [Zou, R. Hu, and Charpentier, 2019]. Most of the work and underlying

logic for heat transfer and fluid mechanics is already present in the thermal hydraulics library, but we do need a working model of the entire natural circulation loop in order to perform this test. Moreover, most of the natural circulation data uses CIET Heater v1.0, so a validated model of CIET Heater v1.0 must also be constructed in addition to the DRACS Heat Exchangers and other components.

Forced Convection Validation

We can also validate the CIET Digital Twin using forced convection transients. This was done for both RELAP [Nicolas Zweibaum, 2015] and SAM [Zou, R. Hu, and Charpentier, 2019] in the time domain. De Wet also has data for validation based on frequency domain tests [De Wet and Per F Peterson, 2020]. We can perform a similar frequency response test using CIET's Digital Twin and validate it using existing Bode Plots.

Heater v1.0 and Heater v2.0 Validation using Frequency Response Data

For now, we only validated CIET Heater v2.0 Bare as there was already an empirical transfer function derived for it [De Wet and Per F Peterson, 2020]. However, we still need a model of CIET Heater v1.0 because earlier forced convection and natural convection tests were done using CIET Heater v1.0. Additionally, Poresky did tests for both CIET Heater v1.0 and CIET Heater v2.0 when the insulation was still intact [Poresky, 2017]. Poresky used sinusoidal inputs to perturb the heater power for the CIET Heater v2.0 insert prior to the removal of insulation on May 2018 [Poresky, 2017]. At a flow rate of 0.18 kg/s, the heater power was varied from 9 kW (rather than 8 kW for De Wet's data) using sinusoids of 1kW amplitude. The surface temperature and heater outlet temperatures were observed and their oscillation amplitudes were recorded. Additionally, phase lags were recorded with respect to the phase of CIET's oscillating power input. We can use this dataset to validate future models of CIET Heater v2.0.

Loss of Heat Sink Transients and Beyond

The ultimate goal of these tests and validation efforts is for CIET to be able to simulate the effects of a loss of heat sink transient on the entire loop under various scenarios. Hence, once all the validation work is done, we shall want to use the digital twin of CIET to simulate transients of all sorts with different reactor feedback models. We could also use the CIET real-time model for use cases such as operator training, cybersecurity applications or controller development. Moreover, we could even extend the capabilities of the thermal hydraulics library to simulate molten salt, high temperature compressible flows or even multiphase flows. These are all possibilities worth exploring in future work.

Part II

Use Case Demonstration for the Digital Twin: Simulated Neutronics Facility Test bed

Chapter 4

Simulated Neutronics Feedback Construction Principles and Literature Review

4.1 Background

Introduction to Part II of the Dissertation

Now that we have constructed and validated our Type I Digital Twin of CIET, or at least one version of its heater, we now wish to provide a demonstration of this Digital Twin as a safe development testbed for research and development specifically in simulated neutronics feedback controller development. This specific controller would in turn be used to enable the electrical heater in CIET to behave as if the temperature feedback mechanisms in the reactor core are active. It is hoped that once this use case is demonstrated, we can then show that Type I Digital Twins can expedite research and development in general where the physical IET is not available.

Simulated neutronics is the focus of the demonstration because it was the focus of my work prior to the COVID-19 pandemic. Converting CIET into a simulated neutronics feedback facility was the original focus of my PhD dissertation. When CIET was unavailable, I decided to construct the Type I Digital Twin of CIET to serve as an experimental surrogate so that I could use it to develop a simulated neutronics feedback controller[1]. However, doing both digital twin development and studies based on simulated neutronics feedback controllers proved to be too much work to do for one dissertation. To limit the scope of work, the focus of my PhD dissertation has evolved to emphasise the construction of the free and open source (FOSS) Type I Digital Twin of CIET and demonstrate its use in expediting SNF Controller Development, or at least its heater, instead of transient simulation using

[1]This simulated neutronics feedback controller was to be used in studying possible effects of reactor transients on the primary loop using CIET

simulated neutronics feedback facilities.For this topic, detailed construction of a working simulated neutronics feedback controller would have been out of scope. Nevertheless, I could still use previous simulated neutronics feedback work to demonstrate how the Type I Digital Twin could be used as a testbed to assist in the development of controllers such as a simulated neutronics feedback (SNF [2]) controller.

In the longer term, Figure 4.1 illustrates how the two parts of the thesis are meant to come together to support eventual construction of a Digital Twin of CIET with SNF capability:

[2]I know there may be some confusion because "SNF" in a nuclear engineering context may also refer to spent nuclear fuel. Hence, I will spell it out in full quite often to avoid confusion for the reader

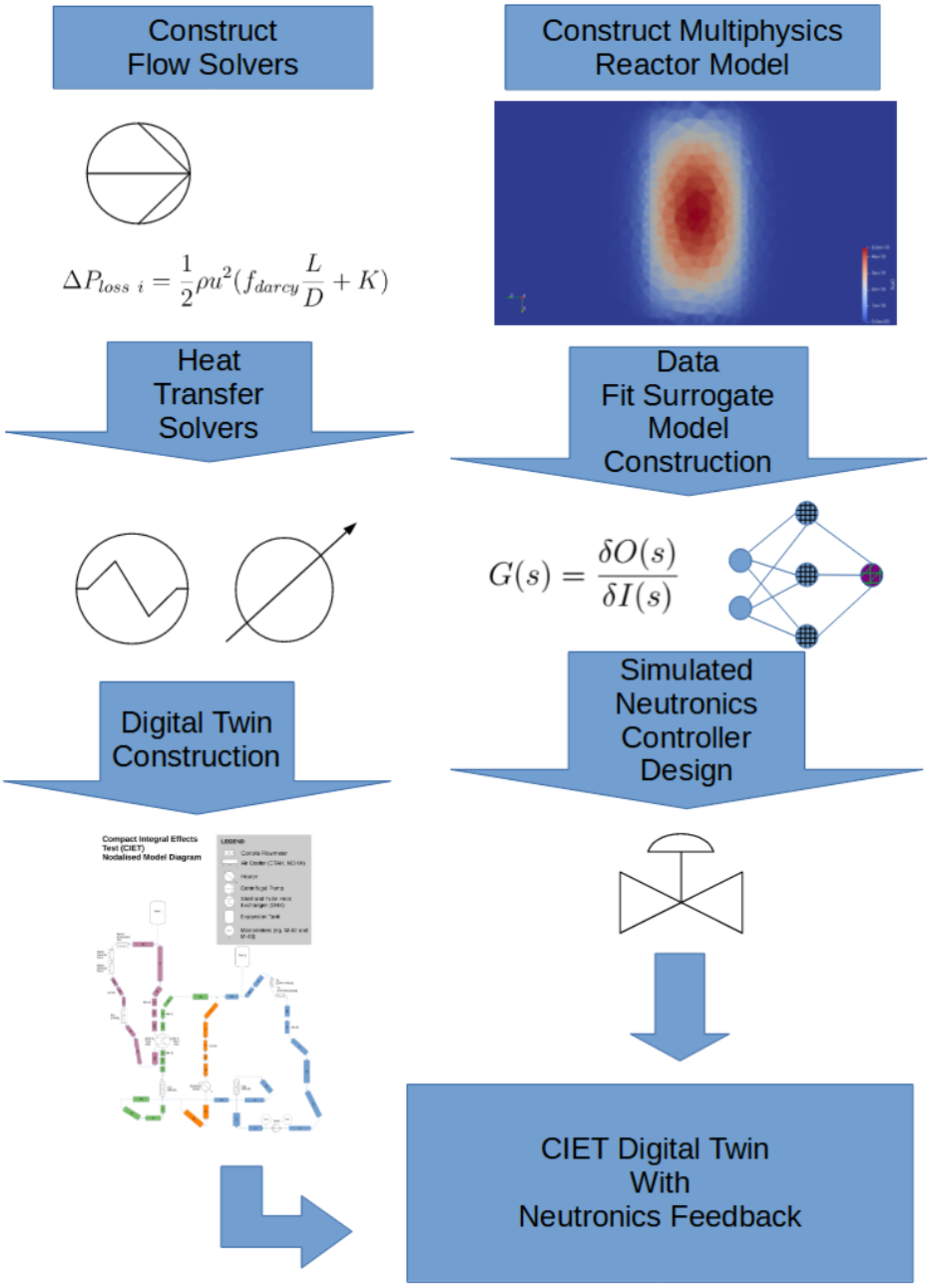


Figure 4.1: Overarching Plan for Constructing Type I Digital Twin with Simulated Neutronics Feedback(Note that the CIET diagram here was originally from my master's thesis [Ong, 2023])

For now, the plan is to construct the Type I digital twin of CIET's heater and use it to

test a reactor feedback controller. Of course, in future, I hope to test the SNF controller in a full Digital Twin of CIET with both forced and natural circulation loops. This effort is done in the hope that we can expedite construction of an actual SNF facility. For Digital Twin construction, the steady state flow isothermal solvers were already completed in my master's thesis [Ong, 2023]. We also covered heat transfer and construction of digital twin in the previous chapters.

For reactor feedback, I wish to construct a multiphysics model and reduce it to a data fit surrogate model which can run in real-time. This could be something as simple as a transfer function or something as complicated as an artificial neural network. For this first iteration, I hope to design a SNF controller into the heater of CIET so that some form of fuel temperature feedback can be simulated[3].

Of course, a fully capable SNF controller can be quite complicated to construct and it is too much content for this one thesis to go through. Here, we only want to demonstrate the usefulness of the constructed Digital Twin in expediting SNF controller design[4]. Of course, actual development of a simulated neutronics feedback controller might require many iterations to achieve the desired result. For the purposes of this demonstration, we will only run through one iteration.

The first part of this individual iteration, covered in this chapter, involves exploring the motivation and general principles of constructing a multiphysics reactor model and data fit surrogate model through a literature review and some discussion. In subsequent chapters, we use these principles in constructing a reactor multiphysics simulation which is subsequently transformed into a simple data fit surrogate model. The subsequent chapters, involve implementing the desired control behaviour in the type I Digital Twin of CIET.We then use some anecdotal estimation based on previous experience with CIET to estimate the time saved for one controller iteration.

Chapter Introduction

In this chapter, we wish to cover the basic motivation behind simulated neutronics facilities, and discuss how one may go about constructing one. We also discuss a method based on data fit surrogate modelling in contrast to the traditional point reactor kinetics equations approach. We then explore possible considerations and software options which can help us construct these data based surrogate models. We also explore some options for constructing multiphysics reactor models to generate data for the surrogate model, and then we shall narrow down the options to fit the goal of this work.

Here, we only seek to demonstrate two concepts. Firstly that digital twins expedite construction of simulated neutronics facilities in general, and secondly, that simulated neu-

[3]Do note I am controlling the heater rather than a valve mostly, it is just that a flow control valve illustrates the idea a little better than a heater

[4]To do so without conducting actual experiments which can compare time taken to develop controllers fully on CIET versus time taken to develop controllers on the digital twin first and then CIET, assumptions using anecdotal knowledge could be used to estimate potential time savings.

tronics facilities can be constructed using data fit surrogate models based on high fidelity multiphysics models. After demonstrating these two concepts, it is hoped that future simulated neutronics facilities can be constructed and used for simulating some Fluoride Salt Cooled High Temperature Reactor (FHR) transients and design basis events. For using simulated neutronics facilities in studies of FHR transients in a scaled integral effects test such as CIET, a whole study needs to be performed. In fact, even constructing the simulated neutronics feedback facility required for studying all transients is beyond the scope of this dissertation. These endeavours are reserved for future work.

For now, we will only demonstrate the concept for how to construct of a simulated neutronics feedback controller using high fidelity multiphysics models, and how this process is sped up with digital twins. Hence, we will not even be constructing a controller fully capable of simulating a transient, we will only demonstrate a plausible process for doing so. In other words, we wish to just perform a "Dry Run" here of how to prepare a simulated neutronics controller using a surrogate modelling method based on high fidelity models. We then time the "Dry Run" development time for a controller and use that to estimate the time saved by using the constructed Digital Twin of CIET.

Given the scope of the dissertation, we will first be exploring in this chapter simulated neutronics facilities and their role in FHR development through a literature review. We then explore the viability of various surrogate models in constructing simulated neutronics facilities. Once the viability of surrogate models is assessed for this work, we then consider high fidelity modelling options for the FHR to generate data for the surrogate model. This includes discussion of underlying equations as well as software choices. We then lay down some general principles in constructing a suitable multiphysics model a first iteration of simulated neutronics controller. For this work, we are using two group diffusion neutronics and porous media equations to describe an arbitrary FHR core used for data generation. Lastly, we consider frequency response testing and transfer functions as a data fit surrogate model and how one would construct this model by testing the two group diffusion multiphysics model.

4.2 Simulated Neutronics Facilities

Role of Test Reactors in Licensing

We have established previously that integral effects tests (IETs) such as CIET play an important role in licensing Gen IV reactors such as the fluoride salt cooled high temperature reactor (FHR). Expediting licensing is important so that we can build more Gen IV reactors such as the FHR in order to help us achieve the 2050 net zero goals [Bouckaert et al., 2021].

Licensing such a reactor design does pose great challenge when it comes to regulation as there are many steps needed before the license is granted for a power reactor [David E Holcomb et al., 2013] such as the FHR.This is because licensing involves building test reactors, such as Oak Ridge National Lab's (ORNL's) molten salt reactor experiment (MSRE)

[Paul N Haubenreich and Engel, 1970], which is in itself an expensive and potentially time consuming endeavour. We may be tempted to use a test reactor similar to the FHR such as the MSRE so that we don't have to spend the extra time and money.However, we cannot use MSRE data for FHR licensing because FHRs are solid fuelled [Andreades et al., 2016] while the MSRE was a liquid fuelled molten salt reactor. As such, we may still need to build test reactors for each Gen IV reactor design we wish to license such as the FHR.Building these test reactors is important because they would help us study certain transients and Design Basis Accidents. These studies are critical to the licensing process.

Common Transients to be addressed in Licensing FHRs

For the FHR, some of these transients, such as unprotected loss of forced cooling (ULOFC), unprotected loss of heat sink (ULOHS) and, in general, anticipated transients without SCRAM (ATWS), are important to study because they may cause damage to the core of the FHR or its structural supports. A particular case of interest is the ULOHS transient, where the primary salt pump (PSP) trips and the reactor is not automatically shut down. In this scenario, while there is convection of heat from the core, the fuel and coolant salts would conceivably rise to very high temperatures. Thankfully, fuel pebbles within FHRs are generally quite resilient as TRISO particles within the fuel pebble can withstand temperatures of 1400 to 1600 °C. Nevertheless, we still have to consider the structural supports and pipes for the FHR since they are most vulnerable to local rises in temperature. This happens because there is relatively high thermal resistance between the metallic structures and air, and relatively low thermal resistance between hot salt and metal. Should the salt pump continue to run, core temperatures will be lower as compared to the case where the PSP trips. However, they are still high enough such that coolant salt exiting the core may damage the primary loop. Even if the PSP were to trip in a ULOFC scenario, there would be natural circulation which may bring a lower flowrate of extremely hot salt from the core. Nevertheless, the reactor core would be kept at a higher temperature than that of the ULOHS, and due to fuel temperature feedback mechanisms, the reactor power output would be low for ULOFC as compared to ULOHS.While, average temperatures throughout the loop would be higher in ULOHS compared to ULOFC, the metallic structures in the primary loop are the only intermediate heat sink available in both these scenarios. In both scenarios, the structural temperatures will be subject to heightened temperatures. Therefore, we will need to consider how their temperatures evolve over time for the case of the PSP tripping and the PSP not tripping. In particular, we should ensure that these temperatures do not exceed the temperature limits structural materials in the primary loop.

To better understand what these temperature limits are, we may want to review some material properties of typical materials used for structural supports. These supports are usually made from metallic materials such as Hastelloy N, Inconel 617, 316 L and 316 H stainless steel. Of the 316 Stainless Steels, 316 H with higher carbon content than 316 L has been shown to have better corrosion resistance in molten salts such as FLiNaK [Doniger et al., 2023], and therefore tend to be favoured over 316 L stainless steels. The typical upper

limits of operation temperature for metallic materials such as Hastelloy N and Inconel 617 range from $700°C$ to $850°C$ [D. Jiang et al., 2022]. This makes these materials suitable for FHRs, such as the Mk 1 PB-FHR and KP-FHR, which typically operate around 550 $°C$ to 700 $°C$ [D. Jiang et al., 2022].

At temperatures higher than these, we risk the structures experiencing accelerated creep and deformities [T. Allen et al., 2013]. Therefore, we must be aware of the temperatures of the coolant which come into contact with these pipes and structural supports. This may be one of the reasons why the peak bulk coolant temperature, or the maximum temperature of FLiBe coolant, is listed as a thermal hydraulic figure of merit for the FHR [Nicolas Zweibaum, 2015].Peak bulk coolant temperatures usually do not usually reach concerning levels except in the case of transients and accidents. Therefore, to ensure that the FHR is designed safely, we need to subject the test reactor to such transients and monitor the peak bulk coolant temperatures so that they do not exceed the safety limits of the metallic materials. If they happen to do so, the FHR should be redesigned such that the safety limits on peak bulk coolant temperature are not exceeded.

To find out if the peak bulk coolant temperature is exceeded during a transient, we would subject a test reactor to such a transient and perform design improvements based on the information gathered to ensure that the reactor performs within these safety limits. However, building multiple tests reactors for iterative design in this fashion can be impractical. This is because constructing test reactors comes with its own set of challenges, particularly when it comes to additional costs of obtaining permits to handle radioactive material and dispose of spent fuel. Furthermore, performing large numbers of transients on actual reactors could result in significant structural damage to the reactor vessel and fuel [Nicolas Zweibaum, 2015]. Such complications could make testing of these transients too economically, socially and ecologically costly if improperly managed. Proper management of these risks would cost more time and monetary resources. It is also likely that obtaining a license for such a test reactor involves demonstrating that these risks are properly mitigated. This process further adds to the time needed for reactor research and development. For all intents and purposes, using test reactors in the early stages of FHR development may be too costly in terms of time. If we wish to expedite FHR development, using only test reactors in the licensing process may be impractical. Moreover, building and operating the test reactors also requires a licensing process. While this process is less stringent than for a full power reactor, we may still need to ascertain if the reactor itself can undergo transients safely.

Role of Integral Effects Tests (IETs) in Licensing

To ascertain if a test reactor design is safe, we might use IET such as CIET to provide us with test data.This test data can in turn be used to show that the reactor design is reasonably safe to operate. Additionally, using non nuclear integral effects tests (IETs) to better understand thermal hydraulic phenomena is advantageous because we do not have the added complexity and cost of handling radioactive material.

However, having an IET which deals with reactor coolants such as molten salt can still prove financially challenging due to costs of the salt [Nicolas Zweibaum, 2015][5] and materials needed to withstand its high temperatures and salt corrosion. These costs can be prohibitive for early research into novel reactor designs. As such, scaled IETs such as CIET takes help to ease the technical difficulty and financial burden in the early stages by using simulant fluids instead of molten salts. Therefore, one can investigate thermal hydraulic phenomena in the simulant fluid IET in earlier research and development stages so that one does not have to deal with the high temperature environment or Beryllium toxicity in the case of FLiBe.

How Simulated Neutronics Facilities can address some shortfalls of IETs

Unfortunately, integral effects tests (IET) in general do not replicate reactor physics or thermal feedback behaviour of a reactor core. If we want to investigate how the coupled thermal hydraulics and neutronics phenomena affects the peak bulk fluid temperature in real time, we will need to artificially implement a more realistic reactor behaviour in our IET.

One method to introduce the realism of reactor physics in test facilities is to program controllers which can simulate reactor behaviour in transient studies. These facilities are known as simulated neutronics facilities. Simulated neutronics facilities have been constructed using point reactor kinetics for space reactors [Shannon M Bragg-Sitton, Godfroy, and K. Webster, 2010], direct drive gas cooled fast reactors [S M Bragg-Sitton and K. L. Webster, 2007] using NASA's Safe Affordable Fission Engine (SAFE)-100a reactor [Shannon M Bragg-Sitton and Forsbacka, 2004]. These are not limited to space reactors as simulated neutronics facilities have also been used for boiling water reactors [Kok and Van der Hagen, 1999; Furuya, Fukahori, and Mizokami, 2007; H. Chen et al., 2017; Shi et al., 2015; Marcel, Rohde, and Van Der Hagen, 2017]. The Delft Simulated Reactor (DESIRE) loop was one such example where simulated neutronics was used in an IET using simulant fluids (Freon-12) in place of the actual coolant, pressurised water [Kok and Van der Hagen, 1999]. This shows that that augmenting simulant fluid IETs, such as CIET, with simulated neutronics capabilities can be considered to be an established method.

Most of these simulated neutronics facilities use the point reactor kinetics equations, because they can be solved in real-time.

$$\frac{\partial}{\partial t}P(\vec{r},t) = \frac{\rho(t)-\beta}{\Lambda}P(\vec{r},t) + \sum_{i}^{n}\lambda_{decay,i}C_i(\vec{r},t) \tag{4.1}$$

$$\frac{dC_i(\vec{r},t)}{dt} = \frac{\beta_i}{\Lambda}P(r,t) - \lambda_{decay,i}C_i(\vec{r},t) \tag{4.2}$$

[5]For Li_2BeF_4 (FLiBe), costs include money needed for lithium enrichment [Nicolas Zweibaum, 2015]

$P(\vec{r},t)$ is the power distribution of the reactor. $\rho(t)$ is the reactivity, C_i is the delayed neutron precursor concentration for group i, with appropriate conversion factors multiplied in to have it in units of power. Λ is the mean neutron generation time, β_i is the delayed neutron fraction for group i, and β is the total delayed neutron fraction. $\lambda_{decay,i}$ is the decay constant for group i.

$$\beta = \sum_{i}^{n} \beta_i \tag{4.3}$$

In its simplest form, the PRKE does not usually consider spatial variations in flux or power. However, it can partly account for neutron flux distribution provided its shape does not change with time.

4.3 Using Surrogate Modelling in Simulated Neutronics Facilities

Issues with the Status Quo Simulated Neutronics which use Point Reactor Kinetics

In literature, real time simulated neutronics feedback often works well with point reactor kinetics equations (PRKE) due to the fast calculation times of such a model. Nevertheless, despite the relatively wide use of point kinetics based simulated neutronics feedback in literature, one must note that PRKE does not consider variations in the shape of the neutron flux over time. In general, PRKE is incapable of predicting detailed reactor transients caused by rapid localised changes in reactivity [Duderstadt and Hamilton, 1976].

Temporal variations in neutron flux shape are quite important in FHR core because these will ultimately impact reactor stability as in the case of Xenon Oscillations. Moreover, as control rods are inserted, the shape of the flux distribution in the core will change such that the flux is higher in regions further from the control rods and lower in regions closer to the control rods [Duderstadt and Hamilton, 1976] as seen in Figure 4.2.

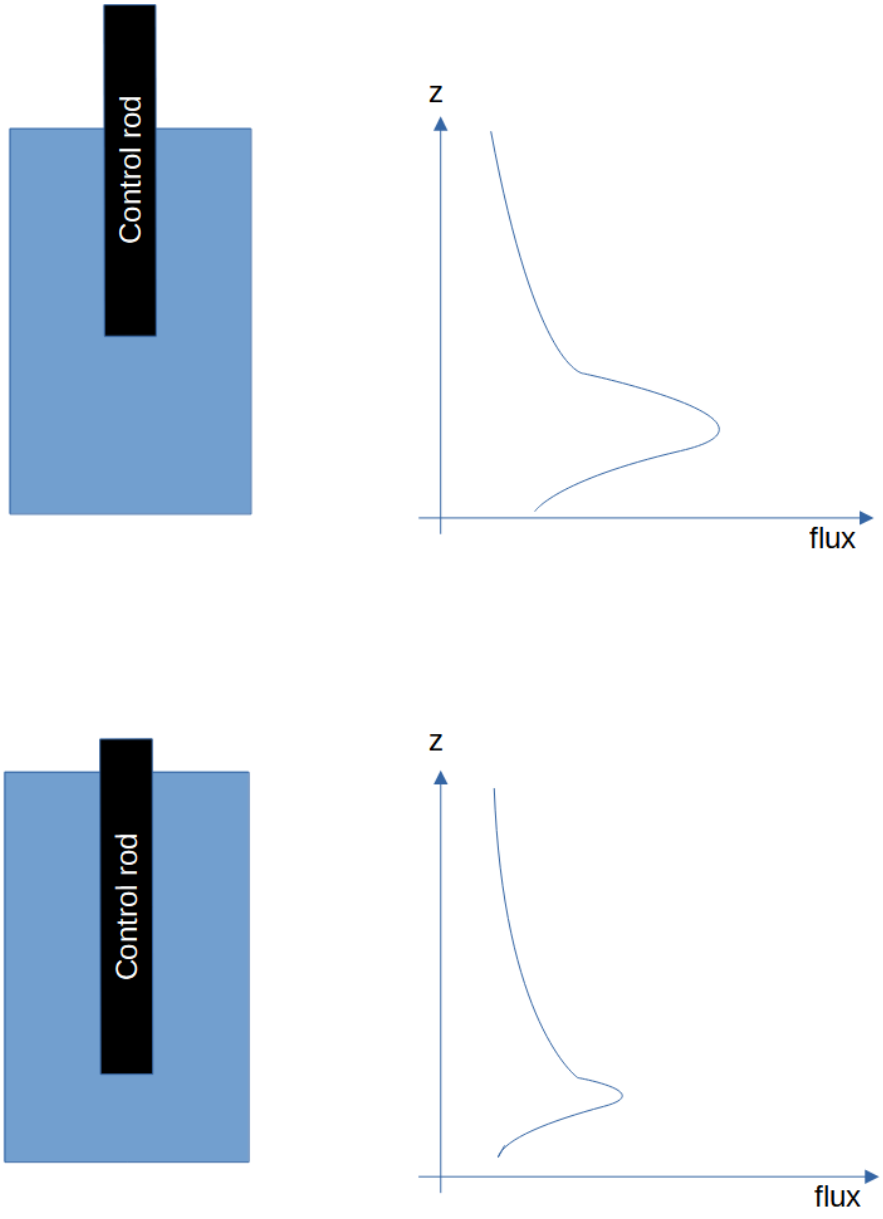


Figure 4.2: positions of control rods change flux shape within the reactor core, flux not drawn to scale

In a similar manner, the fuel temperature feedback coefficients make it such that as the fuel temperature increases, the resonance absorption cross sections over the fuel also increase.

Thus, the effect is as if reactor poisons or control rods were "inserted" into the fuel where temperature increases. If there is a situation where the rate at which core experiences a time varying spatial temperature distribution, then this will cause the core to also have a time varying neutron flux distribution. In this case, the neutron flux will be higher near colder fuel pebbles and lower near hotter fuel pebbles.

Such time varying temperature and flux profiles are bound to happen during transients. One possible transient where this can play an important role is unprotected loss of forced cooling (unprotected LOFC) where reduced coolant flow would cause core temperatures in general to rise. Figure 4.3 shows that during normal operations, the fuel temperature nearer the exit region of the core is generally hotter than the fuel temperature near the entrance region.

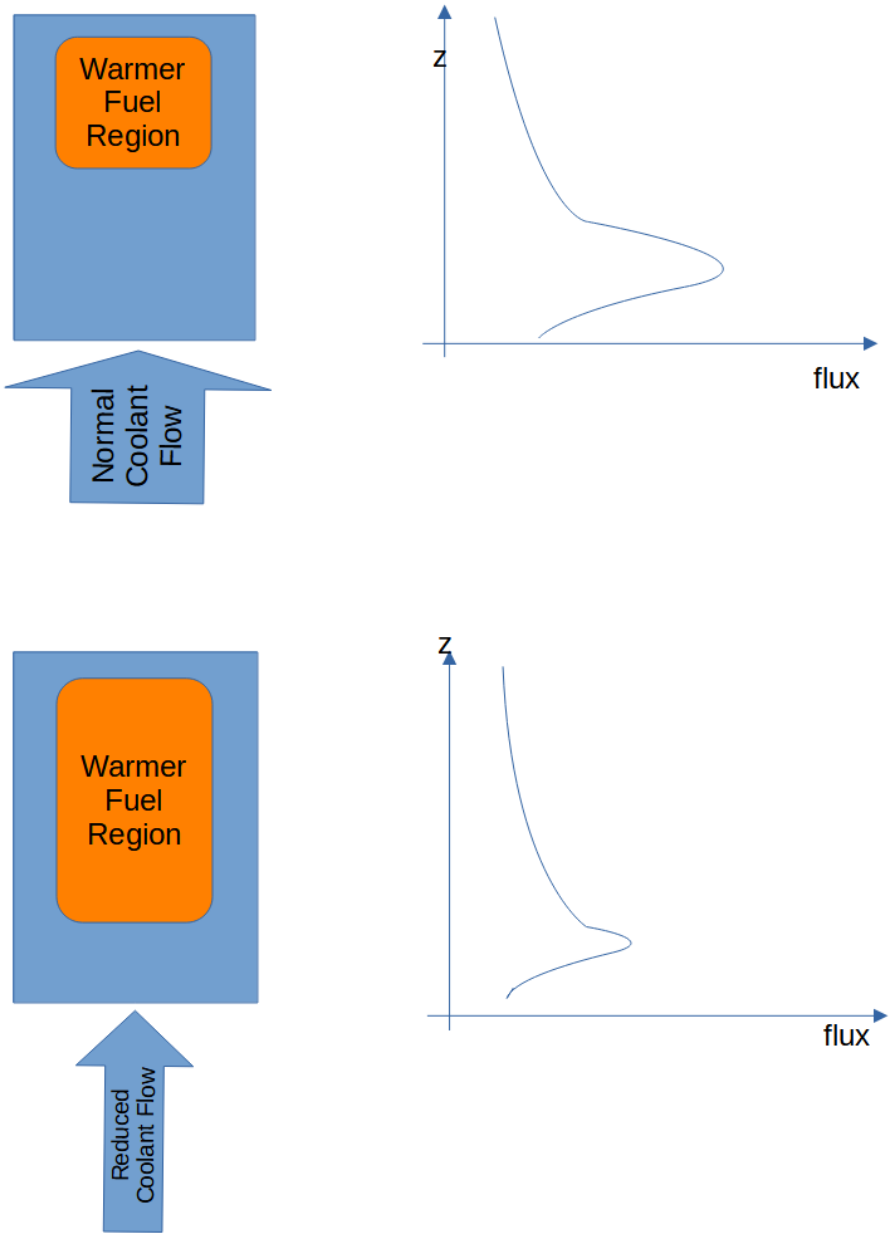


Figure 4.3: Rate of Coolant Flow Change Flux Shape within the Reactor Core, flux not drawn to scale

During ULOFC transients, the core temperatures rise in general. However, due to the reduced flow of coolant, the region of warmer fuel expands. We could of course define warm

as fuel exceeding a certain temperature, but I'm only interested to give a general idea here. As the region of warmer fuel expands, the flux shape would also change. This would render the assumptions behind the PRKE to be quite invalid.

If we were to insist on using PRKE for modelling reactor dynamics during ULOFC, then we must consider the question of how an average fuel temperature, moderator temperature or coolant temperature is obtained. This is quite important because as the flux shapes change, then different regions of the core become more important to account for. This is because the reactor's power as a whole would be more sensitive to temperature changes in regions with higher neutron fluxes. Let us consider first that reaction rates are equal to the integral of neutron macroscopic cross sections and neutron flux over the volume of the reactor. For a change in macroscopic cross section induced by a given temperature change, a region with higher neutron flux would experience a greater power difference compared to a region of lower neutron flux. This simple thought experiment assumes that the flux profile is roughly constant with cross section changes. In reactor physics, the picture is often more complicated as flux, cross sections and temperature profiles are often independent. I assume here that neutron flux does not change appreciably due to temperature changes. However, a thought experiment with this simplifying assumption is still useful in that one can see that not all temperature changes within a reactor are created equal. In the fuller picture of reactor feedback, one should also note that the spatial distribution of cross sections, power distributions, temperature and flux are all interdependent. This makes certain regions of the core more critical or important to consider for the flux and cross section profile. This concept is explored upon heavily in literature in a concept known as "neutron importance" frequently discussed in the context of adjoint neutron transport equations [Lewins, 1960]. Such detailed discussion is considered out of scope of this dissertation. However, the reader should consider that obtaining a reasonable way of averaging the fuel temperature of the fuel, coolant and moderator is no trivial task. Obviously, volume and mass averaged power would not make sense since temperature changes would be more important in regions of high neutron flux or high power density. However, power profiles and temperature profiles are somewhat dependent on each other as, flux shapes and spatial distribution of cross sections are quite dependent on each other. Therefore, we cannot assume the same power profile in every scenario. These problems complicate the use of PRKE for accurately simulating such transients and make the use of PRKE, in its traditional form, too cumbersome or too inaccurate to use.

These problems are not unique to ULOFC because any unprotected transient changing the spatial temperature distribution $T(x, y, z)$ of the fuel, coolant and moderator in the core would also change the flux shape, thus invalidating the PRKE assumptions. These temperature distribution changes would ultimately affect cross sections in a non-uniform manner over time, and thus the flux shape may differ over time when transients occur. This leads to a heterogeneous distribution of reactivity, heat generation rate and temperature distribution [Stewart et al., 2022]. Thus, we can see that anything that affects the thermal hydraulics behaviour of the core would ultimately impact flux shape of the core as well. This means we may want to take into account any entity that can affect the thermal hydraulics

behaviour of the core. For example, the reflector structure would also heat and cool at different rates compared to the core due to the differences in thermal inertia, this would most definitely impact the rate at which the warm region grows during a ULOFC transient, and also the shape of the warm region over time. It is evident that in each of these transients, flux shape of the core changes over time. Therefore, PRKE equations may not describe these transients with sufficient detail in real-time. Even if we were to insist on using PRKE, then we have to consider how the feedback coefficients and average temperatures are calculated such that overall reaction rates are preserved. As discussed earlier, this is no trivial matter. If we want to account for time varying spatial distributions of flux and cross sections, we may want to use another model for simulated neutronics feedback rather than PRKE.

So far, we have highlighted some deficiencies of PRKE mainly in the context of coupled neutronics and thermal hydraulics phenomena. If we were to introduce neutron poisons, heterogeneous burnup distributions and insertion of single control rods, then this phenomena heterogeneous power distribution would become even more important to address and it render PRKE even more unsuitable.

One final consideration is that PRKE equations are often treated using only one energy group. Thus, any effect due to cross section resonances and thermalisation might not be simulated accurately. This is an important effect to simulate in thermal spectrum FHRs since the TRISO fuel,FLiBe and reflector blocks in FHRs would heavily moderate neutrons to the thermal spectrum. In general, neutron moderation effects in FHRs are heavily dependent upon the moderator temperature and density. These would, of course, include FLiBe and any form of graphite or carbon material in the vicinity of the fissile material. Temperature and density are not spatially uniform in the core, and we would once more have to concern ourselves with how to find an average FLiBe density and temperature for reactivity feedback calculations. Moreover, the neutron spectrum in the core is heavily dependent on burnup and reactor poisons, and we wish to investigate how heavily these impact the evolution of unprotected transients in the core. When neutron spectrum changes, we may wish to use multi energy group equations such as multigroup diffusion to account for these changes. Of course, two energy group PRKE equations have been developed in literature [Aboanber, Nahla, and Al-Muhiameed, 2014], and this would mean that multigroup PRKE can be used for modelling as well. We could add as many energy groups as we want to the PRKE.In any phenomena where the PRKE becomes deficient in terms of fidelity, we could then, of course, continue adding more levels to fidelity to improve upon the realism of PRKE equations. However, computational burdens increase with each level of fidelity added and we may not always be sure that these can be calculated in real-time.

We have discussed so far that PRKE may be deficient for simulating reactor feedback mechanisms with spatial variations in flux, cross section, power and temperature. While these spatial variations may not always be important, it is difficult to quantify how important these effects are unless we have a simulation which can account for these spatial variations. For reactors larger than test reactors in the lab (in the order of magnitude of a few kilowatts), it may be reasonable to just assume that these spatial variations are important. Therefore, we require higher fidelity models than the PRKE, which can potentially discretise the spatial

variation and energy distribution of neutrons over time. Unfortunately, high fidelity models by nature cannot produce results for reactor transients available in real time. Computational Time and Power required scales exponentially with number of elements in a simulation. High fidelity simulations with 1 million or more nodes may require several core hours or core days to simulate a minute worth of transients. Therefore, using high fidelity models to simulate reactor transients and calculate heater power in real time is just impractical.

Data Based Surrogate Modelling as Potential Solution to Address lack of Spatial Discretisation capabilities for PRKE based Simulated Neutronics Facilities

To capture higher order fidelity effects while retaining real time calculation capability, a low fidelity empirical model can be used to capture the important transient effects of the high fidelity multiphysics model. In literature, point kinetics modelling parameters were calibrated to a high fidelity coupled thermal hydraulics and neutronics code [Stewart et al., 2022]. However, the low fidelity model is not necessarily a point reactor kinetics model, but could be a range of other models as well which can be calculated in real-time. Sometimes a black box transfer function approach is used to characterise system behaviour [Chinesta et al., 2016] so that it is modelled purely based on an input output basis [Schilders, Van der Vorst, and Rommes, 2008]. Such a technique was used in the DESIRE Loop [Kok and Van der Hagen, 1999; Van De Graaf, Van Der Hagen, and Mudde, 1994]. Hence, we can see that surrogate models look promising in capturing the transient effects of a high fidelity multiphysics models.

4.4 Surrogate Modelling Methods

The act of simplifying a more complex dynamic model into a simpler approximate model for the purpose of speeding up computations can is sometimes known surrogate modelling [Frangos et al., 2010]. Surrogate modelling would include methods such as data fit modelling [Frangos et al., 2010], hierarchical surrogate modelling [Frangos et al., 2010; Asher et al., 2015], Model Order Reduction [Schilders, Van der Vorst, and Rommes, 2008; Asher et al., 2015] and even artificial intelligence [Alizadeh, J. K. Allen, and Mistree, 2020; Asher et al., 2015].

Hierarchical Surrogate Modelling

Hierarchical surrogate modelling makes use of physical models and simplifying assumptions to derive a lower fidelity model from a higher fidelity model [Frangos et al., 2010]. Additionally, hierarchical surrogate models can also be derived from reducing numerical resolution [Asher et al., 2015]. For thermal hydrualics, we could use this method for passive reactor

safety to simplify the governing mass, momentum and energy transport equations to first-order forms. The hierarchical, two-tiered scaling analysis (HTTSA) was used to validate physical assumptions to reduce three dimensional governing equations to first order forms [Per F Peterson, 1994]. This scaling approach based on examining time scales and length scales in order to justify simplifying assumptions is also prominently featured in boundary layer analysis [Bejan, 2013]. These simplifying assumptions greatly reduced the computational burden required to solve the equations.

Hierarchical Surrogate Modelling in Neutronics

In the context of neutronics, the PRKE, diffusion and simplified spherical harmonics (SPN) methods would fall into this category because it applies simplifying assumptions to the neutron transport equation in order to derive a model for the reactor. To better understand this, let us first consider the example of the neutron transport equation [Duderstadt and Hamilton, 1976]:

$$\begin{aligned}\frac{\partial}{\partial t}n(\vec{r},\hat{\Omega},E,t) = \frac{\chi(E)}{4\pi}\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\nu(E')\Sigma_f(\vec{r},E')vn(\vec{r},E',\hat{\Omega}',t) \\ +\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\Sigma_s(\vec{r},E'\to E,\hat{\Omega}'\to\hat{\Omega})vn(\vec{r},E',\hat{\Omega}',t) \\ +Q_{ex}(\vec{r},\hat{\Omega},E,t) - v\hat{\Omega}\bullet\nabla n(\vec{r},\hat{\Omega},E,t) - \Sigma_t vn(\vec{r},\hat{\Omega},E,t)\end{aligned} \tag{4.4}$$

Here, $n(\vec{r},\hat{\Omega},E,t)$ represents the number of neutrons per unit volume and point $\vec{r}$, per unit solid angle at solid angle $\hat{\Omega}$, per unit energy at energy E at time t.

In a reactor, this is equal to the contributions of the fission source ($\chi(E)$ term) to place neutrons in a particular $(\vec{r},\hat{\Omega},E,t)$, external sources Q_{ex} that places neutrons in a particular $(\vec{r},\hat{\Omega},E,t)$, plus inscattering from neutrons at other energies, positions and angles to this particular $(\vec{r},\hat{\Omega},E,t)$, minus the absorption and outscattering of neutrons from a particular $(\vec{r},\hat{\Omega},E,t)$, which is the Σ_t term, minus the neutrons leaking out of a particular volume ($\hat{\Omega}\bullet\nabla n$ term).

Note that $\Sigma_S(\vec{r},E'\to E,\hat{\Omega}'\to\hat{\Omega})$ is known as the double differential cross section since it is the scattering cross section differentiated by both energy and angle.

Here, we may not keep using the term neutrons in a particular $(\vec{r},\hat{\Omega},E,t)$. We shall instead define and use the term phase space. Phase space is a convenient way to show all the possible $(\vec{r},\hat{\Omega},E)$ of a neutron at any time. Neutrons at a point in this phase space would mean that these neutrons correspond to a particular $(\vec{r},\hat{\Omega},E)$.

In neutronics for reactor physics, it is common to use the term neutron flux where we define angular neutron flux: $\psi(\vec{r},\hat{\Omega}',E,t) = vn(\vec{r},\hat{\Omega}',E,t)$. For neutrons interacting with nuclei, density of neutrons does not matter for reaction rate as much as the collision frequency of neutrons. The latter quantity is more closely related to flux than neutron density.

Therefore, we like to write our equations in terms of flux rather than neutron density.

$$\begin{aligned}\frac{1}{v}\frac{\partial}{\partial t}\psi(\vec{r},\hat{\Omega},E,t) = \frac{\chi(E)}{4\pi}\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\nu(E')\Sigma_f(\vec{r},E')\psi(\vec{r},E',\hat{\Omega}',t) \\ + \int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\Sigma_s(\vec{r},E'\to E,\hat{\Omega}'\to\hat{\Omega})\psi(\vec{r},E',\hat{\Omega}',t) \\ + Q_{ex}(\vec{r},\hat{\Omega},E,t) - \hat{\Omega}\bullet\nabla\psi(\vec{r},\hat{\Omega},E,t) - \Sigma_t\psi(\vec{r},\hat{\Omega},E,t)\end{aligned} \tag{4.5}$$

Note, in both equations Q_{ex} refers to external source, in equation 4.4, this is in units of neutron density, but in equation 4.5, this is in terms of neutron flux. We note that in this form of the neutron transport equation, we assume that there is multiplying media that produces neutrons isotropically when fissions occur.

This equation is difficult to solve in its entirety, hence simplifying assumptions are usually applied. Otherwise, Monte Carlo methods such as Serpent, MCNP or OpenMC [Romano and Forget, 2013; Romano, Horelik, et al., 2015] are used to solve the neutron transport equation. This can be quite time and resource intensive for computers. It makes it impossible to solve the equation for reactor transients in real time.

Applying hierarchical surrogate modelling here would mean applying simplifying assumptions to make the neutron transport equation easier to solve. For example, the PRKE equations often assume that we consider neutrons are all of the same energy [Duderstadt and Hamilton, 1976] and that cross section dependence on energy and incident neutron angle is ignored. Additionally, we also ignore changes in spatial distribution of neutrons over time.

Of course, we can choose varying levels of simplifications. We could have a multigroup energy discretisation method for neutrons, consider time varying spatial distribution of neutron flux, and consider the angular anisotropy of cross sections to some degree. The diffusion equations and simplified spherical harmonics equations are derived by applying some of these assumptions to simplify the solution to the NTE.

It is worthwhile noting that hierarchical surrogate modelling is not generally applicable since the assumptions in deriving these models do not always hold well for every situation. For example, the diffusion approximation does not work well around highly absorbing regions [X. Wang, 2018], hence, other hierarchical surrogate models such as the SP3 equations to better account for flux anisotropy in these regions. In fact, the SP3 equations can be shown to be an asymptotic correction to the diffusion equations [X. Wang, 2018; E. W. Larsen, J. E. Morel, and J. M. McGhee, 1993; M. Modest and Lei, 2012]. Hence, all of these are hierarchical surrogate models.

Hierarchical models can also involve solving the discretised form of the equation on a coarser mesh [Frangos et al., 2010] than what otherwise would reflect the physics exactly. For example, in the neutron transport equation, the discrete ordinates method (SN) equations would discretise solid angle by splitting it into a fixed number of domains rather than have angles simualted over a contiuum just like for Monte Carlo. One other example is to use a two energy group diffusion equation in place of a continuous energy spectrum.

Strengths and Weaknesses of Hierarchical Surrogate Modelling

Hierarchical surrogate models allow for simplifying assumptions to be made so that a complex equation becomes solveable. For example, the integrodifferential neutron transport equation can prove extremely difficult to solve using Monte Carlo methods within a reasonable timeframe for large reactors. Hence, simplifying assumptions such as the diffusion approximation are made so that the surrogate model is solveable using less computational power. Even desktop computers will be able to solve these surrogate modelling equations [X. Wang, 2018]. Furthermore, while each of these models can prove to be somewhat lengthy to derive, having knowledge of the simplifying assumptions used will allow the user to know where the surrogate model is applicable.

The problem in using hierarchical surrogate modelling for simulated neutronics feedback is that there is no guarantee that the models can be run in real-time just like for the diffusion equation. One would then be tempted to introduce additional simplifications to ensure the model can be solved in real-time.

Yet, with every simplifying assumption added to simplify the model, the surrogate model would become less generally applicable. These models may only prove to be useful for a small set of reactor physics applications. The PRKE equation for example, will be useful for fast real-time calculations, but not account for spatial variations in flux over time. Remove a few assumptions and we are back at diffusion. The diffusion equation can account for these time varying spatial variations in flux over time, but would be impractical to calculate in real-time despite being a surrogate model.

With hierarchical modelling, it can be difficult to find the "sweet spot" where reactor physics are sufficiently accounted for in real-time. To see how these dynamics are at work, we can look an example of hierarchical surrogate modelling in a neutronics context.

Overview of Deriving Multigroup Diffusion in Homogeneous Media as an example of Hierarchical Surrogate Modelling

An overview of deriving diffusion neutronics from the NTE is presented to show an example of how hierarchical surrogate modelling works. Hierarchical surrogate modelling can involve physics based assumptions to simplify the model [Frangos et al., 2010]. This is done in the diffusion equation where the treatment of cross section angular dependence and anisotropic scattering is heavily simplified. The second example is where a coarser grid may be used to solve equations [Frangos et al., 2010]. Spatially, we can see this as using a coarser mesh in finite volume simulation. In terms of energy, we can see this in play when we use a few energy groups in multigroup diffusion rather than treat the neutron energy spectrum as a continuum. We may observe that despite these assumptions and even with the assumption of homogeneous media, the multigroup diffusion approximation to the NTE cannot be solved in real-time. Thus, its use in real-time simulated neutronics facilities is quite limited in its present form.

Not all the steps are presented in detail since that would be too lengthy for this discus-

sion. Interested readers can read neutronics textbooks such as those written by Duderstadt [Duderstadt and Hamilton, 1976]. To demonstrate how hierarchical modelling works, let us sequentially apply these simplifications to the NTE so that we obtain the homogeneous medium diffusion equations.

Let's consider the NTE again.

$$\begin{aligned}\frac{1}{v}\frac{\partial}{\partial t}\psi(\vec{r},\hat{\Omega},E,t) = \frac{\chi(E)}{4\pi}\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\nu(E')\Sigma_f(\vec{r},E')\psi(\vec{r},E',\hat{\Omega}',t)\\ +\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\Sigma_s(\vec{r},E'\to E,\hat{\Omega}'\to\hat{\Omega})\psi(\vec{r},E',\hat{\Omega}',t)\\ +Q_{ex}(\vec{r},\hat{\Omega},E,t) - \hat{\Omega}\bullet\nabla\psi(\vec{r},\hat{\Omega},E,t) - \Sigma_t\psi(\vec{r},\hat{\Omega},E,t)\end{aligned}$$

Zeroth Moment Equations For reactor applications, only want to care about the angle integrated flux as these most directly impact fission rates $\phi(\vec{r},E,t)$:

$$\phi(\vec{r},E,t) = \int_{4\pi} d\hat{\Omega}\psi(\vec{r},E',\hat{\Omega}',t) \tag{4.6}$$

Integrating out the angles, we obtain the zeroth moment equations:

$$\begin{aligned}\frac{1}{v}\frac{\partial}{\partial t}\phi(\vec{r},E,t) = \chi(E)\int_0^\infty dE'\nu(E')\Sigma_f(\vec{r},E')\phi(\vec{r},E',t)\\ +\int_0^\infty dE'\Sigma_s(\vec{r},E'\to E)\phi(\vec{r},E',t) + Q_{ex}(\vec{r},E,t) - \nabla\bullet\vec{J}(\vec{r},E,t) - \Sigma_t\phi(\vec{r},E,t)\end{aligned} \tag{4.7}$$

Where we define neutron current $\vec{J}$ as:

$$\int_{4\pi} d\hat{\Omega}\hat{\Omega}\psi(\vec{r},\hat{\Omega},E,t) = \vec{J}(\vec{r},E,t) \tag{4.8}$$

First Moment Equation The next step to deriving the diffusion equation is the first moment equation. In the first moment equation, we multiply $\hat{\Omega}$ in first and then perform the integration again. This is important because with a few assumptions, we will get the diffusion coefficient from the first moment equation. Applying these assumptions is what makes the diffusion equation a hierarchical surrogate model of the neutron transport equation (NTE).

For clarity, $\hat{\Omega}$ is [Duderstadt and Hamilton, 1976]:

$$\hat{\Omega} = \hat{e}_x\sin\theta\cos\phi + \hat{e}_y\sin\theta\sin\phi + \hat{e}_z\cos\theta = \Omega_x + \Omega_y + \Omega_z \tag{4.9}$$

This is not to be confused with $d\hat{\Omega}$ as

$$d\hat{\Omega} = \sin\theta d\theta d\phi = -d\mu d\phi \tag{4.10}$$

Therefore, multiplying $\hat{\Omega}$ into each equation is the same as multiplying the vector components into each equation and we integrate separately. We can then recombine each equation into what we can represent as a vector equation. Overall, the process looks like this:

$$\begin{aligned}\int_{4\pi} d\hat{\Omega}\hat{\Omega}\frac{1}{v}\frac{\partial}{\partial t}\psi(\vec{r},\hat{\Omega},E,t) = \int_{4\pi} d\hat{\Omega}\hat{\Omega}\frac{\chi(E)}{4\pi}\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\nu(E')\Sigma_f(\vec{r},E')\psi(\vec{r},E',\hat{\Omega}',t) \\ + \int_{4\pi} d\hat{\Omega}\hat{\Omega}\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\Sigma_s(\vec{r},E'\to E,\hat{\Omega}'\to\hat{\Omega})\psi(\vec{r},E',\hat{\Omega}',t) \\ + \int_{4\pi} d\hat{\Omega}\hat{\Omega}Q_{ex}(\vec{r},\hat{\Omega},E,t) - \int_{4\pi} d\hat{\Omega}\hat{\Omega}\hat{\Omega}\bullet\nabla\psi(\vec{r},\hat{\Omega},E,t) - \int_{4\pi} d\hat{\Omega}\hat{\Omega}\Sigma_t\psi(\vec{r},\hat{\Omega},E,t)\end{aligned} \tag{4.11}$$

Now, to spare the reader from lengthy derivations, I will skip to the end result.

$$\begin{aligned}\frac{1}{v}\frac{\partial}{\partial t}\vec{J}(\vec{r},E,t) = \int_0^\infty dE'\Sigma_{s1}(\vec{r},E',t)\vec{J}(\vec{r},E',t) + \vec{Q}_{ex1} - \frac{1}{3}\nabla\phi(\vec{r},E,t) \\ -\Sigma_t(\vec{r},E,t)\vec{J}(\vec{r},E,t)\end{aligned} \tag{4.12}$$

Where,

$$\Sigma_{s1}(\vec{r},E',t) = \int_{4\pi} d\hat{\Omega}\hat{\Omega}\bullet\hat{\Omega}'\Sigma_s(\vec{r},E'\to E,\hat{\Omega}'\bullet\hat{\Omega}) \tag{4.13}$$

$$\vec{Q}_{ex1} = \int_{4\pi} d\hat{\Omega}\hat{\Omega}Q_{ex}(\vec{r},\hat{\Omega},E,t) \tag{4.14}$$

This is done assuming a azimuthal symmetry in flux and scattering cross section, linearly anisotropic angular flux and linearly anisotropic scattering cross section, expanded using the P1 approximation. When external neutron sources are considered isotropic, then we can consider the external sources in the first moment equation to go to zero as well. We also assume that our fission source and total cross section are isotropic.

To further simplify the first moment equation, we assume the source is isotropic so $\vec{Q}_{ex1} = 0$ and that the variation of neutron current with time is small. Do note that the neutron speed v, even for thermal neutrons is $v \approx 2200\frac{m}{s}$ at 0.0253 eV. Thus, it may be reasonable to assume that the variation of neutron current with time is small compared to the other terms. This of course assumes that $\frac{\partial}{\partial t}\vec{J}(\vec{r},E,t)$ is comparable in order of magnitude to the other terms in the equation.

We can also assume that sources within the medium are isotropic (if they even exist at all), so:

$$Q_{ex}(\vec{r},\hat{\Omega},E,t) = 0 \tag{4.15}$$

$$\frac{1}{3}(\nabla)\phi(\vec{r},E,t) = \int_0^\infty dE'\Sigma_{s1}(\vec{r},E'\rightarrow E,t)\vec{J}(\vec{r},E',t) - \Sigma_t(\vec{r},E,t)\vec{J}(\vec{r},E,t) \tag{4.16}$$

It should be noted that the first moment equations are vector equations, whereas the zeroth moment equations are scalar equations.

Multigroup Equations as a Discretisation of Energy The next step is to discretise the energy groups. Within these energy groups, the cross sections are assumed to be the same regardless of energy. We can have as few as one or two energy groups, or even more than forty [Duderstadt and Hamilton, 1976]. For FHRs, we might use around eight energy groups to represent the entire spectrum [X. Wang, 2018]. A hierarchical surrogate model may use coarse grids to simplify a model [Frangos et al., 2010] Here, treating the energy dependence using multiple energy groups simplifies the calculation. This is an example of coarsening the energy grid, and therefore, this also makes the multigroup diffusion equation a hierarchical surrogate model.

Multigroup Zeroth Momment Equations Let us define an energy group g with an lower bound energy of E_g and upper bound energy of E_{g-1}. In this definition, the highest energy group is group 1. To obtain the neutron balance over energy group g for the zeroth moment equations,

$$\begin{aligned}\int_{E_g}^{E_{g-1}} dE\frac{1}{v_g}\frac{\partial}{\partial t}\phi(\vec{r},E,t) = \int_{E_g}^{E_{g-1}} dE\chi(E)\int_0^\infty dE'\nu(E')\Sigma_f(\vec{r},E')\phi(\vec{r},E',t)\\ + \int_{E_g}^{E_{g-1}} dE\int_0^\infty dE'\Sigma_s(\vec{r},E'\rightarrow E)\phi(\vec{r},E',t) + Q_{ex}(\vec{r},E,t) - \int_{E_g}^{E_{g-1}} dE\nabla\bullet\vec{J}(\vec{r},E,t)\\ - \int_{E_g}^{E_{g-1}} dE\Sigma_t\phi(\vec{r},E,t)\end{aligned} \tag{4.17}$$

$$\phi_g = \int_{E_g}^{E_{g-1}} dE\phi(\vec{r},E,t) \tag{4.18}$$

$$Q_{ex,g} = \int_{E_g}^{E_{g-1}} dEQ_{ex}(\vec{r},E,t) \tag{4.19}$$

$$\chi_g = \int_{E_g}^{E_{g-1}} dE\chi(E) \tag{4.20}$$

Now, the flux averaged cross sections of type i, moment n, and group g are defined as:

$$\Sigma_{i,n,g}\phi_g = \int_{E_g}^{E_{g-1}} dE\phi(\vec{r},E,t)\Sigma_{i,n}(\vec{r},E,t) \tag{4.21}$$

$$\phi_{g'}\Sigma_{s,g'\to g} = \int_{E_g}^{E_{g-1}} dE \int_{E_{g'}}^{E_{g'-1}} dE'\Sigma_s(\vec{r},E'\to E)\phi(\vec{r},E',t) \tag{4.22}$$

We also write the divergence of neutron current, which can be interpreted as a leakage term from the control volume, for energy group g to be:

$$\nabla \bullet \vec{J}_g(\vec{r},t) = \int_{E_g}^{E_{g-1}} dE\nabla \bullet \vec{J}(\vec{r},E,t) \tag{4.23}$$

After some steps and substitutions, we arrive at our multigroup version of the zeroth moment equations:

$$\frac{1}{v_g}\frac{\partial}{\partial t}\phi_g(\vec{r},t) = \chi_g \sum_{g'=1}^{G} \nu_{g'}\Sigma_{f,g'}(\vec{r},E')\phi_{g'}(\vec{r},t) + \sum_{g'=1}^{G} \phi_{g'}\Sigma_{s,g'\to g} - \nabla \bullet \vec{J}_g(\vec{r},t) - \Sigma_{t,g}\phi_g(\vec{r},t) \tag{4.24}$$

Here, in the context of reactor physics, external sources are neglected, this is another simplification.

Multigroup First Moment Equations Now let's apply the same multigroup treatment to the first moment equation. The divergence operator $\nabla\bullet$ is used on the first moment equations to convert it into a scalar equation. However, the implicit assumption in this operation is that the material is homogeneous so that cross sections do not vary with respect to distance position $\vec{r}$. This makes the diffusion equation in this form not as applicable with respect to variations in cross sections with space.

$$\int_{E_g}^{E_{g-1}} dE\frac{1}{3}(\nabla \bullet \nabla)\phi(\vec{r},E,t) = \int_{E_g}^{E_{g-1}} dE \int_0^{\infty} dE'\Sigma_{s1}(E'\to E,t)\nabla \bullet \vec{J}(\vec{r},E',t) - \int_{E_g}^{E_{g-1}} dE\Sigma_t(E,t)\nabla \bullet \vec{J}(\vec{r},E,t) \tag{4.25}$$

After several simplifying assumptions, we arrive at a form which is our diffusion equation.

$$-\frac{1}{3(\Sigma_{t,g} - \Sigma_{s1,g\to g})}(\nabla \bullet \nabla)\phi_g(\vec{r},t) = \nabla \bullet \vec{J}_g(\vec{r},t) \tag{4.26}$$

Where:

$$\phi_g(\vec{r},t)\Sigma_{s1,g'\to g} = \int_{E_g}^{E_{g-1}} dE \int_{E_{g'}}^{E_{g'-1}} dE'\Sigma_{s1}(E'\to E)\phi(\vec{r},E,t) \tag{4.27}$$

Our diffusion coefficient is:

$$D_g = \frac{1}{3(\Sigma_{t,g} - \Sigma_{s1,g\to g})} \tag{4.28}$$

Here, besides assuming spatially homogeneous media, we also had to assume the following:

- Isotropic scattering between energy groups
- Leakage ($\nabla \bullet \vec{J}_g$) weighted Σ is approximately equal to flux weighted Σ, this implies a non strongly absorbing media

Multigroup Diffusion Equations in Homogeneous Media We substitute in the simplified first moment equations back to our zeroth moment equations to obtain the multigroup diffusion equation in homogeneous media:

$$\begin{aligned} \frac{1}{v_g}\frac{\partial}{\partial t}\phi_g(\vec{r},t) &= \chi_g \sum_{g'=1}^{G} \nu_{g'}\Sigma_{f,g'}\phi_{g'}(\vec{r},t) \\ + \sum_{g'=1}^{G} \phi_{g'}\Sigma_{s,g'\to g} + Q_{ex,g}(\vec{r},t) &+ D_g\nabla^2\phi_g(\vec{r},t) - \Sigma_{t,g}\phi_g(\vec{r},t) \end{aligned} \tag{4.29}$$

Conclusion on Hierarchical Surrogate Modelling for Diffusion Now, hierarchical surrogate modelling has allowed us to simplify our equations so that we can solve faster than with Monte Carlo methods. This makes it somewhat usable for simulating reactor transients. We also know the limitations of these equations since we explicitly used physical assumptions to simplify the mathematical equations. For example, cross sections and group diffusion coefficients change with temperature. In that sense, the reactor medium is not homogeneous. Therefore, the homogeneous material assumption limits the use of this model in simulating transients when coupled with thermal hydraulics.

The multigroup diffusion equations when homogeneous materials are not assumed would look like:

$$\begin{aligned} \frac{1}{v_g}\frac{\partial}{\partial t}\phi_g(\vec{r},t) = \chi_g \sum_{g'=1}^{G} \nu_{g'}\Sigma_{f,g'}\phi_{g'}(\vec{r},t) + \sum_{g'=1}^{G} \phi_{g'}(\vec{r},t)\Sigma_{s,g'\to g} + Q_{ex,g}(\vec{r},t) \\ + \nabla \bullet D_g\nabla\phi_g(\vec{r},t) - \Sigma_{t,g}\phi_g(\vec{r},t) \end{aligned} \tag{4.30}$$

Only in this form can we use the multigroup diffusion equation to account for changing cross sections and diffusion coefficients with temperature.

In any case, the problem with this is that these equations cannot be solved in real-time. Hence, we could be tempted to go with more simplifying assumptions, and we would then arrive at PRKE which can be solved in real-time. Unfortunately, this would land us back in square one as the PRKE cannot account for spatial reactor kinetics which can be important in large reactors [Duderstadt and Hamilton, 1976].

A second problem is that the multigroup diffusion equation is not "accurate enough" in that it fares poorly in regions of discontinuous cross sections and strongly absorbing regions such as near control rods [X. Wang, 2018]. As a result, other hierarchical surrogate models such as the simplified spherical harmonics (SPN) equations have been used because of improved performance. Yet with higher fidelity required, more computational resources and time will be needed to simulate the NTE with a higher fidelity surrogate model. This makes it impractical for evaluating and simulating reactor feedback in real-time.

Therefore, using hierarchical surrogate modelling alone may not be suitable for use in simulated neutronics feedback facilities.

Data Fit Surrogate Modelling

Data fit modelling is more of a "black box" technique where we extract data and try to map the inputs to outputs. One common example well known in control theory is the transfer function [Asher et al., 2015]. Data fit surrogate modelling would also include methods used in deep learning such as Gaussian Processes [Frangos et al., 2010; Asher et al., 2015]. In fact, neural networks used in artificial intelligence are considered to be a data driven surrogate model [Asher et al., 2015].

Transfer Functions as an Example of Data Fit Surrogate Modelling

In this section, we outline transfer functions as one important method of data fit surrogate modelling. This is because of their well known historical use in stability analysis and controller development of nuclear reactors. For example,transfer functions have been used in literature to describe the transient behaviour of small test reactors such as Iowa University's UTR-10 research reactor [T.-C. Chan, 1971] and the molten salt reactor experiment (MSRE) [Ball and TW Kerlin, 1965]. Due to their use in controller development, it was thought that using them to develop controllers for simulated neutronics feedback would be suitable as well.

Transfer functions $G(s)$ are describe the relationship between system inputs and outputs in the frequency domain. We traditionally use a Laplace transform to convert these inputs and outputs to the frequency domain [T.-C. Chan, 1971]. These inputs and outputs specifically refer to the deviation of a variable from its steady state value. Therefore, the inputs and outputs are denoted $\delta I(s)$ and $\delta O(s)$ respectively where s is the complex frequency.

The transfer function can be described as [De Wet and Per F Peterson, 2020]:

$$G(s) = \frac{\delta O(s)}{\delta I(s)} \tag{4.31}$$

$$s = \sigma + j\omega$$

An example of an input could be control rod position as it was for the MSRE [Ball and TW Kerlin, 1965] and UTR-10 [T.-C. Chan, 1971] experiments. The corresponding output could be reactor temperature or reactor power.

Obtaining Transfer Functions from Experimental Data To obtain transfer functions, one would usually perturb the system, in this case the nuclear reactor, using some sort of input signal $\delta I(t)$ after it has reached steady state. This input signal could be a control rod insertion or some variation of pump speed to change coolant flowrate through the reactor.

The experimentalist would then note the change in the system's output $\delta O(t)$. These inputs and outputs can be converted into the frequency domain using a Laplace Transform to obtain $\delta I(s)$ and $\delta O(s)$. Two common examples of inputs that are used to obtain the transfer function of the system are the step functions and sinusoidal inputs.

Step Functions and Time Domain Tests One of the most common ways to obtain a transfer function is with the use of step inputs in time domain testing. The step function is also known as the heaviside function [Legua, Morales, and Sánchez Ruiz, 2008]. The graph of a step response in the time domain is shown in Figure 4.4:

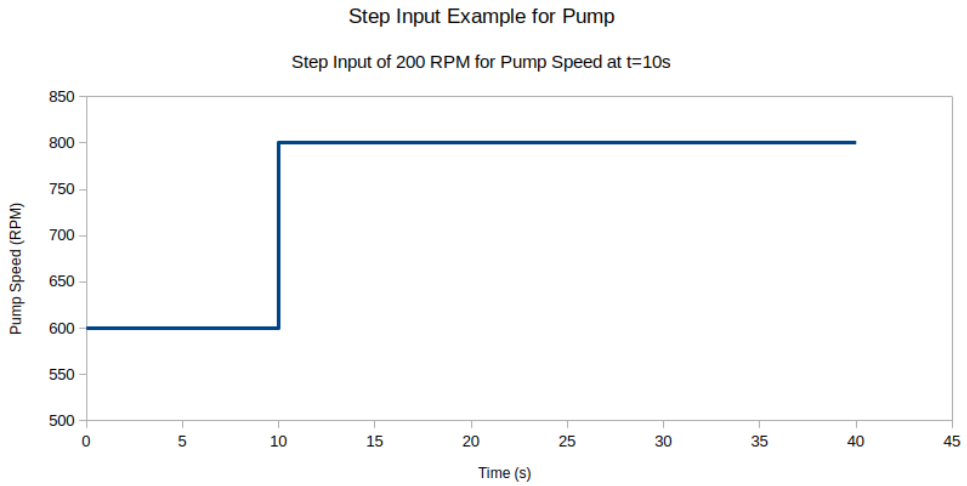


Figure 4.4: Example of Heaviside Function

This test is relatively simple, but it perturbs all frequencies at once. In step responses and pulse response tests, obtaining a good signal to noise ratio can be challenging [Kerlin,

2012]. For a step response of amplitude A after some delay time c, the frequency domain representation found in equation 4.32:

$$\delta I(s) = \frac{Ae^{-cs}}{s} \tag{4.32}$$

We present an Fourier Transform Plot using $A = 5$ and $c = 0.1$, in Figure 4.5:

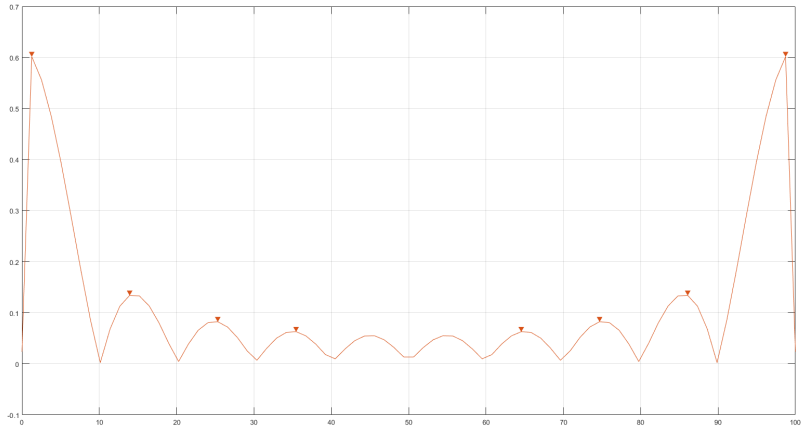


Figure 4.5: Example of Heaviside Function in Frequency Domain

Step inputs have been used in reactivity measurements [Schmid, 1957] and to simulate on/off switches for dynamic reactor simulations [Amendola et al., 1982]. Step increases in reactivity were also used to determine open loop transfer functions of the miniature neutron source reactor (MNSR) of Syria [Hainoun and Khamis, 2000]. This shows that the process of using step functions for making measurements of reactor properties and transfer functions has already been established in literature.

Sinusoids and Frequency Response Tests It is more common, however, to see literature for the use of repetitive forcing compared to single step changes for nuclear reactors [Kerlin, 2012; Sanathanan and Tsukui, 1974]. This is because we can obtain a better signal to noise ratios with repetitive forcing methods [Kerlin, 2012]. This is because in repetitive forcing, we amplify the signal strength in the frequency domain by repeating it periodically [Kerlin, 2012]. Whereas for single step changes, the signals are not periodic, and therefore weaker compared to the background noise [Kerlin, 2012]. To analyse the system response to a periodically oscillating function, we normally look at the frequency domain rather than the time domain. This involves plotting the ratio of the amplitudes of the input and output signals (gain) and phase shift of the input and output signals of various frequencies. This process of perturbing a system and characterising it by the gains and phase shifts over various frequencies is known as frequency response testing.

In theoretical frequency response, sinusoids (sines and cosines) have been used to characterise a system [Kerlin, 2012] in frequency response tests. This is because it is easy to observe a phase shift and gain of the system using sinusoids. In practice, sinusoids are difficult to generate using hardware as compared to binary pulses. Therefore, binary pulses are used more often in practical frequency response for physical systems. As sinusoidal frequency response testing is theoretically easier to discuss, let us start with this first before moving onto more complex signals.

When we use sinusoids in frequency response testing, we only perturb one frequency at a time. The characteristic frequency s we perturb becomes:

$$s = j\omega$$

A possible frequency domain representation of the sinusoidal inputs, in this case sines, can be written as [Oberhettinger and Badii, 2012]:

$$\delta I(s) = \frac{\omega}{s^2 + \omega^2} \tag{4.33}$$

In the time domain, a hypothetical example of the sinusoidal forcing function and the system response is plotted in Figure 4.6:

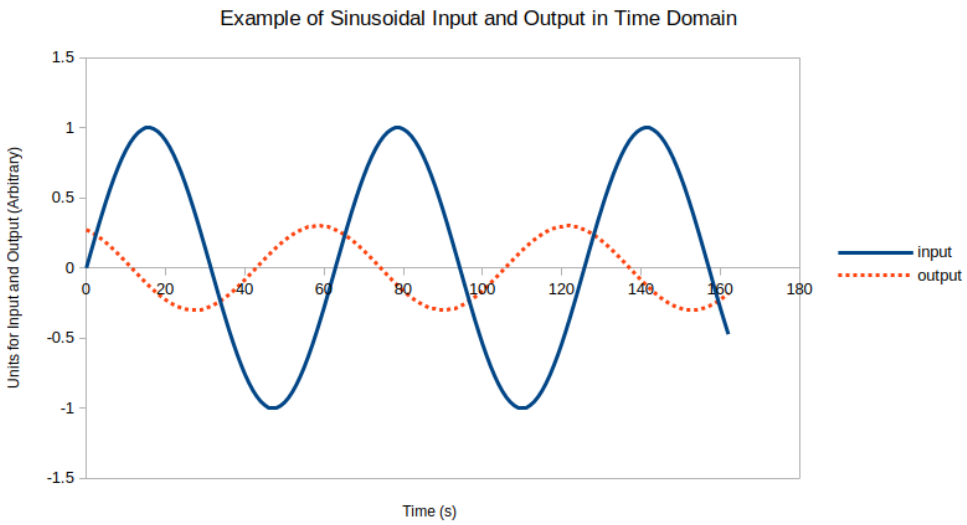


Figure 4.6: Example of Sinusoidal Function Inputs and Outputs

For a typical sinusoidal input, we see that the output is also a sinusoid. The oscillation frequency remains the same, but the amplitude and phase of the oscillation is changed. For an input sinusoid of amplitude a and angular frequency ω, the input would look like:

$$\delta I(t) = a \sin(\omega t) \tag{4.34}$$

The output can be represented as:

$$\delta O(t) = a|G(j\omega)| \sin(\omega t + \psi) = b \sin(\omega t + \psi) \tag{4.35}$$

The amplitude of the output is b and the phase shift is ψ. The ratio between the input and output amplitudes $|G(j\omega)|$ is known as the system gain.

$$|G(j\omega)| = \frac{b}{a} \tag{4.36}$$

To obtain a transfer function, one repeats the procedure of perturbing the system with sinusoids of various frequencies noting the phase shifts and magnitude gains at different input frequencies. The gains and phase shifts at different frequencies are plotted in graphs known as Bode plots. This procedure is known as frequency response testing. A Bode plot for the system with a transfer function $G(s) = \frac{1}{s+1}$ is provided in Figure 4.7 for the reader's reference.

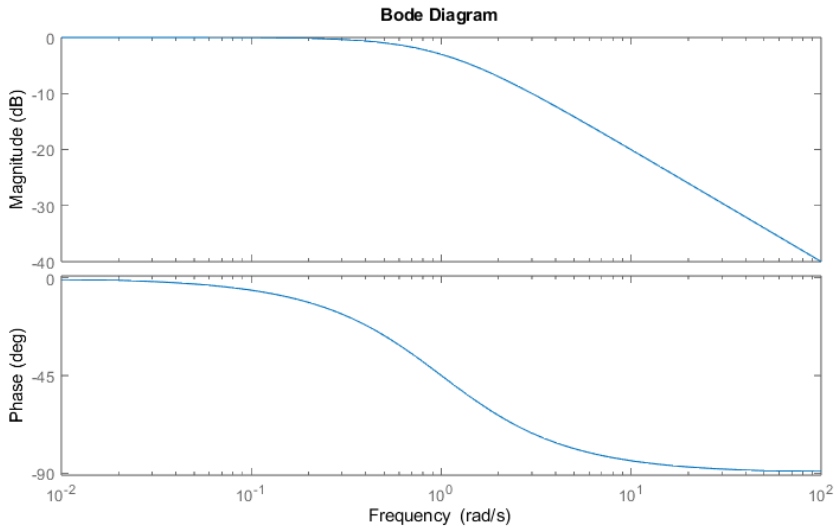


Figure 4.7: Example of Bode Plot for $G(s) = \frac{1}{s+1}$

For more background information on frequency response testing and frequency response testing in CIET, one can explore work done by DeWet and Poresky [De Wet and Per F Peterson, 2020; Poresky, 2017]. Frequency response methods have been used in test reactors such as the molten salt reactor experiment (MSRE) [Robinson and Fry, 1970] and Experimental Breeder Reactor II (EBR-II) [Rhodes, Furstenau, and Larson, 2000].

Practical Considerations for Frequency Response In practice, transfer functions can be estimated from Bode plots by hand [Sanathanan and Tsukui, 1974], or they can be fitted using programs such as Matlab's system identification toolbox with the function tfest

[Ljung and Singh, 2012]. Matlab's tfest is short for transfer function estimation. It can be used to estimate transfer functions from both time domain and frequency domain data. This often involves some form of least squares regression of the Frequency Response Plots [Valério, Ortigueira, and Sá da Costa, 2008]. This method of least squares regression is already well established in MATLAB's system identification toolbox [Kollár, Pintelon, and Schoukens, 1991]. The fitted transfer function can then be subjected to a step response test in the time domain to check if the transfer function has been correctly fitted [Sanathanan and Tsukui, 1974].

To obtain the frequency domain data, we may traditionally use sinusoids. While perturbing a system one sinusoid at a time produces a very clean signal (it concentrates the energy at one frequency), this procedure can be very slow. For lower frequencies such as a 0.001 Hz sine wave, we require about 10000s for 10 oscillations. This excludes the time required for a system to reach steady state after startup. Therefore, we may need alternate input signals to speed up the testing procedure [De Wet and Per F Peterson, 2020].

Now presuming the system is linear, this being the simplest case, the laws of superposition will hold. Where one sine wave is able to perturb the system at one frequency, multiple sine waves can test it at multiple frequencies. Therefore, one could construct a signal to be a summation of sine waves in order to perturb the system at multiple frequencies at once [De Wet and Per F Peterson, 2020]. These are known as broadband inputs because a summation of sine waves with a band of different frequencies is used to perturb the system rather than a single sine wave at a time [De Wet and Per F Peterson, 2020].

If more time is required to be saved, then one could use pseudorandom noise to perturb the system. This would, after a very long time, perturb the system at several frequencies, thus saving time on the number of tests needed. We use pseudorandom noise rather than truly random noise because we wish for such tests to be repeatable [Anderson, Finnie, and Roberts, 1967].

Nevertheless, using pseudorandom noise or even sum of sinusoid inputs may prove difficult experimentally as in the reactor system [Kerlin, 2012], the control rod may have difficulty moving to various precise positions within a very short span of time [De Wet and Per F Peterson, 2020]. Therefore, binary sequences are favoured as inputs over their continuous counterparts where on or off type signals can be used in place of continuous signals. For example, square waves can be considered the binary approximation of sine waves and pseudorandom binary signals (PRBS) can be considered the binary approximation of continuous background noise [Kerlin, 2012]. We can visualise this by plotting power spectrum (P_k) of a square wave of amplitude A is given by [Kerlin, 2012]:

$$P_k = \begin{cases} \frac{A^2}{k^2} 0.81 & \text{if k odd} \\ 0 & \text{if k even} \end{cases} \tag{4.37}$$

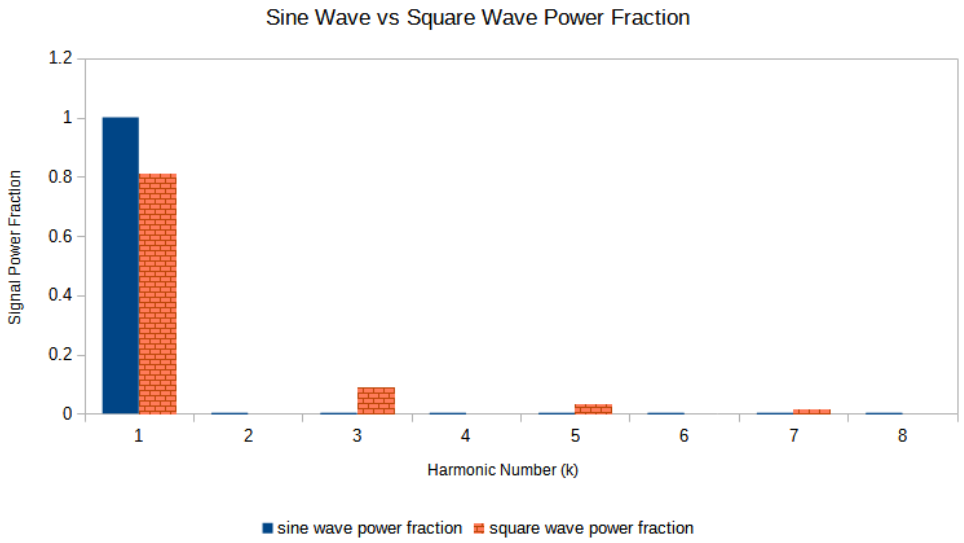


Figure 4.8: Power Spectrum Comparison for Sine Wave and Square Wave with same Fundamental Frequency

We can see that 81% of the signal energy is contained within the first harmonic [De Wet and Per F Peterson, 2020; Kerlin, 2012], provided this first harmonic would be in the same frequency as the sine wave that the square wave approximates. Therefore, binary sequences are quite suitable for substituting their sinusoidal counterparts.

Advantages of using Transfer Functions for Simulated Neutronics Facilities Transfer functions are rather simple as they are 1D models that can be computed in real-time. Furthermore, provided the oscillations are small, transfer functions obtained using frequency response will be able to capture system dynamics at all relevant timescales [De Wet and Per F Peterson, 2020]. Transfer functions obtained using frequency response are also able to capture some of the spatial effects for reactor dynamics especially in higher frequency regions [Loewe, 1965]. Finally, transfer functions usually exist in the natural language of controller design because they are derived using Laplace Transforms. Therefore, it would be relatively easy to design controllers that can simulate neutronics feedback once the transfer functions of the reactor is known.

Extension to Multiple Input Multiple Output Systems So far, we see transfer functions being used to correlate one input to one output in a single input single output (SISO) system. If one wished to use transfer functions in multiple input multiple output (MIMO) systems, then networks of transfer functions can be used to correlate multiple inputs to multiple outputs in what is known as a state space model. These state space models have been used for control applications for Pressurised Water Reactors (PWRs) [G. Wang et al.,

2017]. For this discussion, we constrain ourselves to SISO systems for the sake of brevity of literature review.

However, it is important for the reader to know that the use of transfer functions to model a system is not strictly limited to SISO systems.

Limitations of Transfer Functions Despite these benefits, it must emphasised that such transfer functions replicate system or reactor behaviour only under certain conditions and assumptions. Transfer functions are only well suited for characterising linear time invariant (LTI) systems. Should the system be inherently nonlinear, the system can still be approximated as linear provided the amplitudes of the oscillation are small enough not to cause nonlinear behaviour to occur [De Wet and Per F Peterson, 2020]. In terms of using these transfer functions for simulated neutronics feedback facilities, the utility will be limited as the reactor system and feedback mechanisms are inherently nonlinear.

To characterise nonlinear systems using data fit surrogate modelling, we may need to move beyond transfer functions for LTI systems. For this purpose, we can discuss neural networks and artifical intelligence.

Artificial Intelligence, Neural Networks and Deep Learning

In this day and age (2023), the use of artificial intelligence (AI) methods has permeated all of society. This includes the use of the well known ChatGPT [OpenAI, 2023]. For nuclear reactors, artificial neural networks, and deep learning have also been used as a data fit surrogate model. These neural networks are suitable for characterising system nonlinearities [Asher et al., 2015] and have been used before in neutronics calculations [Gong, S. Cheng, Z. Chen, and Q. Li, 2022; Gong, S. Cheng, Z. Chen, Q. Li, et al., 2022].

Machine learning has been used to model the transportable FHR (TFHR) for real-time reactor control [Zeng et al., 2018] and thus shows promise for implementing simulated neutronics feedback in IETs such as CIET.Nevertheless, given the complexity and breadth of what AI has to offer, it will not be used in this dissertation. However, AI based simulated neutronics feedback seems to hold promise for simulating non linear reactor phenomena in real-time provided it has been appropriately trained.

Reduced Order Modelling and Model Order Reduction

Reduced order modelling, or model order reduction belong to a category of projection based surrogate modelling [Asher et al., 2015]. Model Order reduction (MOR) was originally developed in the field of control theory [Schilders, Van der Vorst, and Rommes, 2008] and is therefore its methods are potentially well suited for controller development. A quick internet search using Google Scholar and ChatGPT [OpenAI, 2023] shows that there are several methods of MOR including proper orthogonal decomposition (POD) [Chinesta et al., 2016], balanced truncation [Schilders, Van der Vorst, and Rommes, 2008] and Krylov Subspace Methods [Schilders, Van der Vorst, and Rommes, 2008]. The goal of such methods

is to attain an accurate enough mapping of model inputs and outputs [Chinesta et al., 2016] that can be computed within a reasonable amount of time depending on the application.

Constrasting MOR to Data Driven Surrogate Modelling

Model Order Reduction (MOR) also aims to reduce computational complexity, but in contrast to data driven methods, there is often added emphasis on preserving important system properties as compared to data driven methods of surrogate modelling. In the context of control systems, the goal of MOR is to reduce the complexity of the system while preserving passivity[6] and stability [Baur, Benner, and Feng, 2014; Schilders, Van der Vorst, and Rommes, 2008]. Sometimes it is intended that the block structure of a dynamic system be preserved [Quarteroni, Rozza, et al., 2014]. In the case of turbulence in fluid mechanics, perhaps the quantity that we want to preserve is turbulent kinetic energy [Weiss, 2019].

Constrasting MOR with Hierarchical Surrogate Modelling

We also wish to contrast MOR with hierarchical surrogate modelling to avoid confusion. A crude analogy that may help the reader understand MOR more intuitively is a low pass filter. In low pass filter, a time varying signal is received by the low pass filter and decomposed into various orthogonal modes or frequencies often using a Fast Fourier Transform (FFT). The user would then decide a suitable cutoff frequency for which modes with frequencies above this cutoff frequency are ignored. The output is then reconstructed by recombining the remaining modes of oscillation to reconstruct a de-noised signal which takes fewer modes to represent and yet captures the dominant and desirable dynamics of the signal. Now, MOR often involves decomposition [Asher et al., 2015] analogous to FFT.For example, we have the POD method and balanced truncation method [Gugercin and Antoulas, 2004]. In fact, balanced truncation is quite conceptually similar to how the low pass filter works. First, we have the system of equations in the form of a matrix. We then decompose this matrix using Singular Value Decomposition (SVD) into component states [Gugercin and Antoulas, 2004]. These components have "amplitudes" which are characterised using Hankel singular values [Gonzalez, 2018] which are analogous to frequency in the low pass filter. These singular values characterise how reachable, observable and controllable a state is [Antoulas, 2005; Gonzalez, 2018]. Only states above a threshold Hankel singular value are kept while the rest are discarded as they are deemed to be "noise". When constructing the reduced order model, only the reachable and observable states with the highest Hankel singular values are used. This would reduce the model while capturing the most important system dynamics.

In contrast, hierarchical surrogate modelling requires some knowledge of the system and physical assumptions to perform this model reduction. For passive reactor safety systems, scaling using HTTSA of the governing equations can result in simplified first-order forms, that can be used to normalize experimental or numerical simulation data [Per F Peterson,

[6]Passivity can be understood as the ability of a system to remain stable even if non-linear components are added to it [Schilders, Van der Vorst, and Rommes, 2008]

1994]. Another prominent example of using some physical assumptions with scaling and nondimensionalisation to simplify governing equations is in deriving the analytical solution for boundary layer flow equations [Bejan, 2013]. In these equations, one would presume that flow velocities perpendicular to the flat plate would be negligible compared to flow velocities parallel to the flat plate near the boundary layer in analysis of system length scales and time scales. This scaling analysis is one of the main simplifying assumptions specific to boundary layer flow. Using these assumptions, we could remove some terms in the governing equations. Thereafter, the governing equations would be nondimensionalised and solved like an Ordinary Differential Equation (ODE) rather than a Partial Differential Equation (PDE), thus lowering computational burden. This approach extends beyond boundary layer flow and can be used similarly in other flow regimes. For example, the relative residence times of fluid in different regions of the system for stratified flow can be analysed in similar scaling analysis [Per F Peterson, 1994] to reduce the complex governing equations to simplified first order forms.

MOR does not use such assumptions and scaling analysis in general, but rather construes the problem as a system of matrices and vectors which can be approximated using matrix decomposition methods. Some of these resulting components from matrix decomposition can be neglected, thus simplifying the model. The basis of this truncation process, such as in balanced truncation [Gugercin and Antoulas, 2004], is considerably more abstract compared to the scaling analysis and does not require specific knowledge about the system. One such method is the Lyapunov balancing method [Gugercin and Antoulas, 2004], which involves matrix decomposition and characterising decomposed states based on abstract mathematical constructs such as observability controllability and reachability which are quantified using their respective Grammians [Gugercin and Antoulas, 2004; Gonzalez, 2018].

While both hierarchical surrogate modelling and MOR result in simplified systems of equations, the basis of the former is based on specialised knowledge of the physical system while the latter relies more on characterising the system using matrix decomposition and control theory.

Examples of MOR

Besides balanced truncation MOR, there are examples of MOR in literature which we shall briefly cover and discuss. Firstly, let's explore Proper Orthogonal Decomposition (POD). POD is a projection technique [Frangos et al., 2010; Asher et al., 2015] which involves decomposing a vector which contains variables of interest into orthogonal functions similar to how periodic functions can be decomposed using a Fourier Transform [Weiss, 2019]. These methods originated from turbulence which were intended to reconstruct the fluctuating parts of the velocity fields using these orthogonal functions with less computational power [Weiss, 2019]. We can use POD to reduce complex systems of Partial Differential Equations (PDEs) into Ordinary Differential Equations (ODEs). This is known as Galerkin projection and it has been used in both fluid mechanics [Weiss, 2019] and neutronics [Elzohery, 2022]. Of course, the applicability of POD went past the domain of multiphysics calculation and was

used to describe dynamic systems of input and output [Chinesta et al., 2016]. For example, for linear time invariant (LTI) systems, POD was used to decompose the LTI systems using frequency domain snapshots as opposed to time domain snapshots [Chinesta et al., 2016].

In the nucelar engineering context, POD was used to generate a reduced order model transfer functions for a couple point reactor kinetics and thermal hydraulics one dimensional model of the multi application small light water reactor (MASLWR) [Zarei, 2021]. Furthermore, it has been used in context of multiphysics simulation codes such as GeN-Foam for neutronics feedback [German, Ragusa, and Fiorina, 2019]. However, it was known that POD generated transfer functions could not capture the high frequency behaviour of the system as it was a lumped system [Zarei, 2021].

Besides POD, another popular method in MOR are Krylov Subspace methods. The Krylov Subspace methods, which are frequently used for solving linear systems of equations $\mathbf{A}\vec{x} = \vec{b}$ [Simoncini and Szyld, 2007] can also be used in the context of MOR to simplify systems of equations.Interested readers can look into literature by Ipsen [Ipsen and C. D. Meyer, 1998] which gives a primer into some of these methods. Krylov subspace methods have also been used for model order reduction of bilinear control systems [Breiten and Damm, 2010; Benner, Breiten, and Damm, 2010]. The Krylov subspace of a matrix $\mathbf{A}$ and vector $\vec{b}$ is:

$$\mathcal{K}_r(\mathbf{A}, \vec{b}) = \text{span}\left(\vec{b}, \mathbf{A}\vec{b}, \mathbf{A}^2\vec{b}, \ldots, \mathbf{A}^{r-1}\vec{b}\right) \tag{4.38}$$

Similar to POD methods, these also involve projection onto an span of basis vectors. These methods were invented to increase the computational speed of iteratively solving large linear systems of equations. Hence, Krylov methods are quite well suited for simplifying high order linear time invariant (LTI) systems [Salimbahrami and Lohmann, 2002; Bai, 2002] where multiple transfer are interconnected. However, where generalised nonlinear problems are concerned, there may be some difficulty in applying Krylov Subspace methods [Asher et al., 2015; Frangos et al., 2010; Feng, 2005].

For nonlinear MOR, there has been some success in using quadratic polynomials in the context of electrical circuits [Feng, 2005; Y. Chen, 1999], and for in reactor multiphysics, POD has been used [German, Ragusa, and Fiorina, 2019]. These methods are not as simple to use as a data fit surrogate model and therefore will be out of scope for this work. Furthermore, their emphasis seems to be on preserving model fidelity more so than increasing speed to enable real-time simulation for reactor feedback. Hence, I will not be exploring use of these methods for this dissertation. However, they could be explored for use in future iterations of simulated neutronics feedback facilities.

4.5 Hybrid Methods

While the surrogate modelling methods are described in three different categories [Asher et al., 2015] for the sake of understanding their working principles, it is important to note that

in practice, these models can be used used together rather than separately. For example, Gong uses physics informed machine learning for a reduced order modelling (ROM) of the HPR1000 reactor [Gong, S. Cheng, Z. Chen, and Q. Li, 2022]. This was done to solve for flux distribution of the HPR1000. A similar approach of using both ROM and AI was used to drive thermal hydrualics design for reactor cores [Popov et al., 2022].

One other notable use was that in the DESIRE Boiling water reactor (BWR) simulated neutronics facility, both hierarchical surrogate models and data driven models are used as PRKE and transfer functions were used in giving the DESIRE loop coupled thermal hydraulics and void reactivity feedback [Kok and Van der Hagen, 1999]. Thus, it is likely that we will use a combination of methods for constructing simulated neutronics facilties.

4.6 Evaluation of Surrogate Models for Simulated Neutronics Facility Development in this Dissertation

Now that we have explored some surrogate modelling classes and methods, we now want to evaluate how practical they are for simulated neutronics facilities. Hierarchical methods based on physics principles [Frangos et al., 2010] alone cannot work for producing simulated neutronics feedback in real-time. We saw that diffusion neutronics often taken too long to solve and that the modelling capability of PRKE is limited.

A simpler method such as determining a transfer function from frequency response data would be better suited because only need to simple model to demonstrate a proof of concept that data-driven surrogate modelling works for simulated neutronics facilities. We might take a high fidelity model such as the NTE, and use a hierarchical surrogate such as the diffusion to obtain reasonably more detailed simulation than the PRKE, and then use data driven surrogate models such as transfer functions to model the hierarchical surrogate. This is a hybrid approach for surrogate modelling. Here, both diffusion and transfer functions have already been established as useful methods in their own right. Frequency response testing is an already established method to probe reactors such as the molten salt reactor experiment (MSRE) [P N Haubenreich, 1969; Robinson and Fry, 1970]. It has favourable characteristics such as its high signal-to-noise ratio [De Wet and Per F Peterson, 2020; Kerlin, 2012]. Given these advantages, these simpler data driven surrogate modelling methods are preferred for generating surrogate models as compared to MOR methods, machine learning methods and other methods at least for this current iteration.

In future, AI and MOR can be explored for the purpose of constructing simulated neutronics feedback facilities. These methods are promising because since ROM [Elzohery, 2022; German, Ragusa, and Fiorina, 2019] and AI [Gong, S. Cheng, Z. Chen, and Q. Li, 2022; Popov et al., 2022] have been used in reactor neutronics and thermal hydraulics applications.

4.7 Modelling FHRs with Multiphysics code

So far, we have determined that using a combination of hierarchical surrogate modelling and data driven surrogate modelling would be suitable for development of simulated neutronics facilities. Specifically, we are developing the controller that gives the electrical heater in the IET some capability to emulate reactivity feedback.

The hierarchical surrogate models to be used here consist coupled high fidelity (high compared to point kinetics models) multiphysics models. These could be diffusion, SPN or even discrete ordinates (SN) based models coupled with thermal hydraulics. The thermal hydraulics models by themselves are usually hierarchical surrogates because we make physical assumptions in order to make the Navier Stokes and heat transfer equations easier to solve. For example, we use closure models in turbulence modelling to practically simulate the effects of turbulence on heat and momentum transfer (e.g. large eddy simulation (LES) [Chen Qingyan, 2000]). The way we choose a surrogate model depends heavily on the phenomena we want to simulate. The number of phenomena we simulate in turn impacts the levels of simulation fidelity. To address these different phenomena, different codes have been developed for simulating reactor multiphysics at these different levels of fidelity. Given the variety of phenomena present in the core and the simulation choices available, we now want to survey some of the multiphysics phenomena which occur in FHR cores. Additionally, we want to survey some methods and codes available for use for modelling FHRs and evalute which of them is most suitable for this work.

The reader should note that material phenomena are assumed to happen on timescales much larger than unprotected transients. Corrosion, metal creep and other material phenomena may lower the threshold temperature for which structural damage over the course of several hours, weeks, months or years, and not in seconds [Franklin, Lucas, and Bement, 1983]. Therefore, these are not directly simulated. Any such discussion is out of scope for this dissertation.

Neutronics Modelling Equations

Now for neutronics modelling, we have discussed earlier that hierarchical surrogate modelling is used to simplify the NTE:

$$\begin{aligned}\frac{1}{v}\frac{\partial}{\partial t}\psi(\vec{r},\hat{\Omega},E,t) = \frac{\chi(E)}{4\pi}\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\nu(E')\Sigma_f(\vec{r},E')\psi(\vec{r},E',\hat{\Omega}',t)\\ +\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}'\Sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\rightarrow\hat{\Omega})\psi(\vec{r},E',\hat{\Omega}',t)\\ +Q_{ex}(\vec{r},\hat{\Omega},E,t)-\hat{\Omega}\bullet\nabla\psi(\vec{r},\hat{\Omega},E,t)-\Sigma_t\psi(\vec{r},\hat{\Omega},E,t)\end{aligned}$$

Into surrogate models such as the multigroup diffusion equation:

$$\frac{1}{v_g}\frac{\partial}{\partial t}\phi_g(\vec{r},t) = \chi_g \sum_{g'=1}^{G} \nu_{g'}\Sigma_{f,g'}\phi_{g'}(\vec{r},t) + \sum_{g'=1}^{G} \phi_{g'}\Sigma_{s,g'\to g} + Q_{ex,g}(\vec{r},t) + D_g\nabla^2\phi_g(\vec{r},t) - \Sigma_{t,g}\phi_g(\vec{r},t) \tag{4.39}$$

Or PRKE:

$$\frac{\partial}{\partial t}P(\vec{r},t) = \frac{\rho(t)-\beta}{\Lambda}P(\vec{r},t) + \sum_{i}^{n}\lambda_{decay,i}C_i(\vec{r},t) \tag{4.40}$$

$$\frac{dC_i(\vec{r},t)}{dt} = \frac{\beta_i}{\Lambda}P(r,t) - \lambda_{decay,i}C_i(\vec{r},t) \tag{4.41}$$

In this case, we want a model which has higher fidelity than PRKE and one that can account for spatial reactor kinetics. There are an assortment of methods we can use for this high fidelity model.

Monte Carlo Methods

The first method to tackle this difficult to solve equation is in essence to use brute force. Firstly, the geometry of the core is set up so that there will be different zones of the core filled with different materials. A neutron will then be simulated in the core independent of other neutrons, and it will travel through the different zones. As the neutron travels , the program will calculate the distance before an interaction occurs. And when an interaction occurs, the computer will reference the probability of a fission reaction, scattering reaction absorption reaction etc based on the neutron phase space and material type.

The Monte Carlo software would simulate thousands and ten thousands of these neutrons to calculate the number of neutrons generated in the next generation along with other quantities of interest. The collection of such neutron histories is known as the Monte Carlo method.

Software such as Monte Carlo N Particle (MCNP), Serpent and OpenMC are common Monte Carlo codes used for neutron transport. OpenMC is an open source implementation of Monte Carlo code and will be used in this work [Romano and Forget, 2013; Romano, Horelik, et al., 2015].

Deterministic Methods

As Monte Carlo methods can get computationally expensive and exceedingly long for complex geometries such as full reactor calculations, there is motivation to develop an alternate

class of methods to solve the NTE.These are known as deterministic methods. The basic idea is to discretise the integrodifferential equations to obtain a set of coupled equations which can then be solved iteratively.

Now there are at least four categories of discretisation that needs to be performed. One is time discretisation, most simply shown in point reactor kinetics. If time and space are discretised, the diffusion approximation comes about , and it accounts for linear anisotropy of flux [Duderstadt and Hamilton, 1976]. This would be a similar level of complexity for scalar transport such as heat transport as well as mass and momentum transport seen in thermal hydraulics. The third and fourth category of discretisation required is solid angle and neutron energy. Angle is also usually discretised further in regions of the reactor where linear anisotropy (diffusion model) is insufficient to describe flux [X. Wang, 2018].

Finally, energy groups also require judicious discretisation to account for spatial self shielding [X. Wang, 2018],[Duderstadt and Hamilton, 1976]. Discretisation results in what we call the multigroup equations, with multigroup diffusion equations being commonly used as an approximation for the neutron transport equation.

Multigroup Diffusion Methods

Angular Treatment For treatment of angular dependence, have already discussed the simplest of these methods: the diffusion approximation. The diffusion approximation assumes that angular flux is linearly anisotropic and that the scattering cross sections are also linearly anisotropic. In a multi energy group setting, we assume that there is isotropic inter-energy group scattering, which is a common assumption [Brantley and E. W. Larsen, 2000]. Fick's diffusion law is applied to relate flux gradients to neutron diffusion rates in what is known as the first moment equation. These methods are commonly used and can be found in multiphysics reactor codes such as GeN-FOAM [Fiorina, I. Clifford, et al., 2015].

Multi-Group Energy Discretisation and Structure For Multi-Group Diffusion, we must carefully consider how the energy groups are structured. For most LWR calculations, about 2 to 4 groups are sufficient whereas for fast reactors, anywhere from 20 to 1000 groups are used [Duderstadt and Hamilton, 1976].

In FHRs and HTGRs, the spectrum is heavily thermalised. Here, as many as eight or nine groups are used [Duderstadt and Hamilton, 1976; X. Wang, 2018]. The main rationale for these groupings are to capture the large cross section differences between fast and thermal groups. However, resonances also have a large cross section difference compared to the cross sections of similar energy ranges in its vicinity, hence for more detailed models, some of these resonances are captured in energy groupings.

These groups can be determined using a few principles. First, let us can consider the lethargy normalised neutron spectrum of the FHR.Flux spectrum data from the generic Fluoride Salt Cooled High Temperature Reactor simulated using KENO and NEWT [Kile

et al., 2022] was read using GraphReader software [K. P. Larsen, 2022] and is presented here in Figure 4.9:

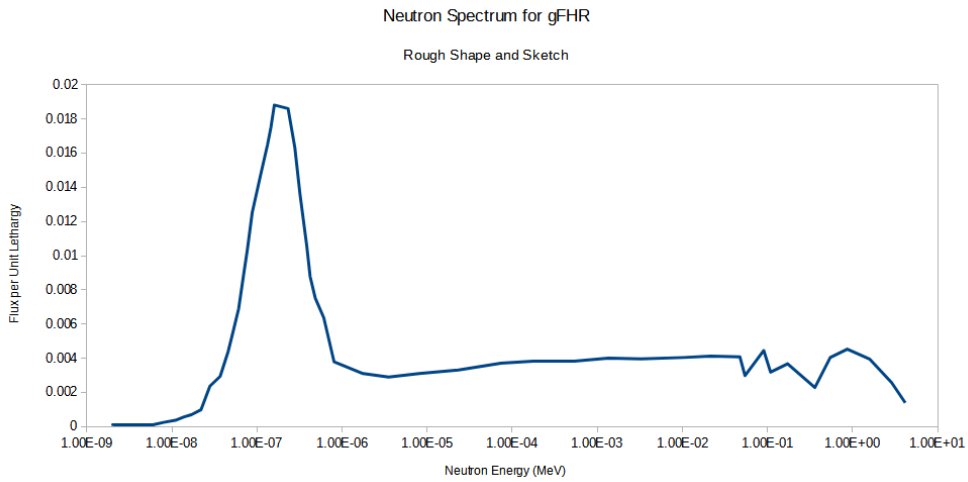


Figure 4.9: Rough Sketch of gFHR Flux Spectrum [Kile et al., 2022] based on GraphReader [K. P. Larsen, 2022]

One could create a two group spectrum to separate the maxwell boltzmann distribution of thermal neutrons from the fast spectrum. In fact, the Maxwell Boltzmann distribution features prominently in the FHR spectrum [X. Wang, 2018]. Hence, the Maxwell Boltzmann peak can be considered one group. Neutrons with energy below the bell curve can be considered one group and neutrons with energy above the bell curve can be considered one group. Thus we can already have three groups considering the Maxwell Boltzmann Peak.

Secondly, we can consider cross section resonances as another basis for grouping fluxes. However, there are many nuclides with resonances. Therefore, we shall only capture resonances pertaining to important nuclides such as U-235 fission and parasitic absorption. Let us consider U-235 resonances obtained from ENDF/B-VII [Chadwick et al., 2006] and overlay them with the gFHR flux spectrum as seen in Figure 4.10:

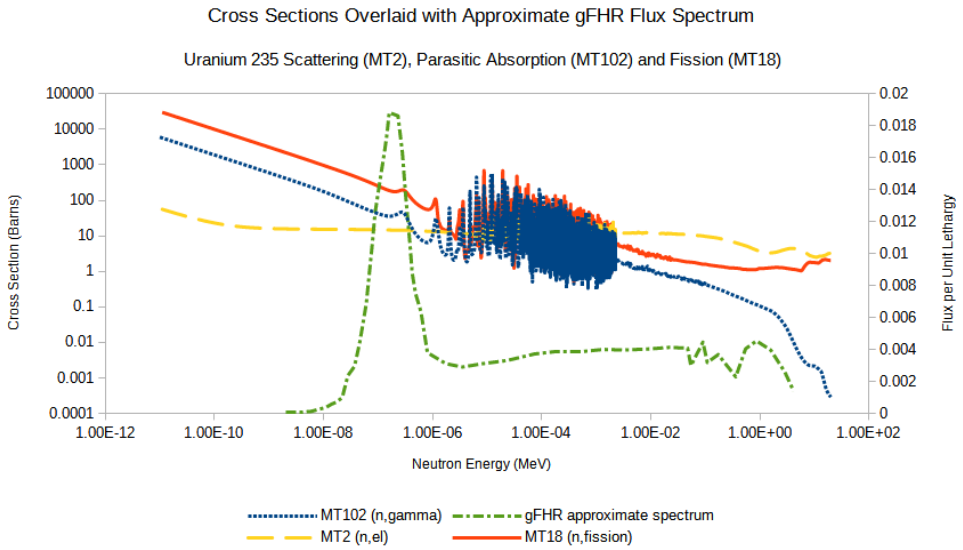


Figure 4.10: gFHR Flux Spectrum with U-235 Cross Sections Overlaid [Chadwick et al., 2006; Kile et al., 2022]

U-235 has have several resonances in the 10^{-5} to 10^{-2} MeV range. We may be tempted to assign one group for each resonance here, and this would be quite troublesome for there are many resonances. Thankfully, the flux at these energies is typically low especially within moderated reactors as seen from the flux spectrum shape of the gFHR plot in Figure 4.10. Hence, we do not necessarily need to use multiple groups to account for each of these resonances. However, low energy resonances play a larger role in overall reaction rates (and hence reactor kinetics) than high energy resonances [Duderstadt and Hamilton, 1976]. This is because fluxes in the thermal region are much higher than in the resonance region of 10^{-5} to 10^{-2} MeV. The relative importance of groups can be visualised by the product of the area under the curves of the neutron spectrum and cross section energy dependence graphs respectively. By this logic, the FHR neutron spectrum shows that most neutrons reside in the thermal region [Fratoni, 2008; X. Wang, 2018]. Therefore, more energy groups can be assigned to lower energy resonances (10^{-7} to 10^{-5} MeV) and one group assigned to cover all high energy resonances (10^{-5} to 10^{-2} MeV).

In a similar manner, we can assign groups for resonances present in U-238, especially low lying ones. One group can be assigned to the three low lying resonances [X. Wang, 2018] for MT102 (n, γ) reactions for U-238 and one group for the rest of the resonance region in Figure 4.11:

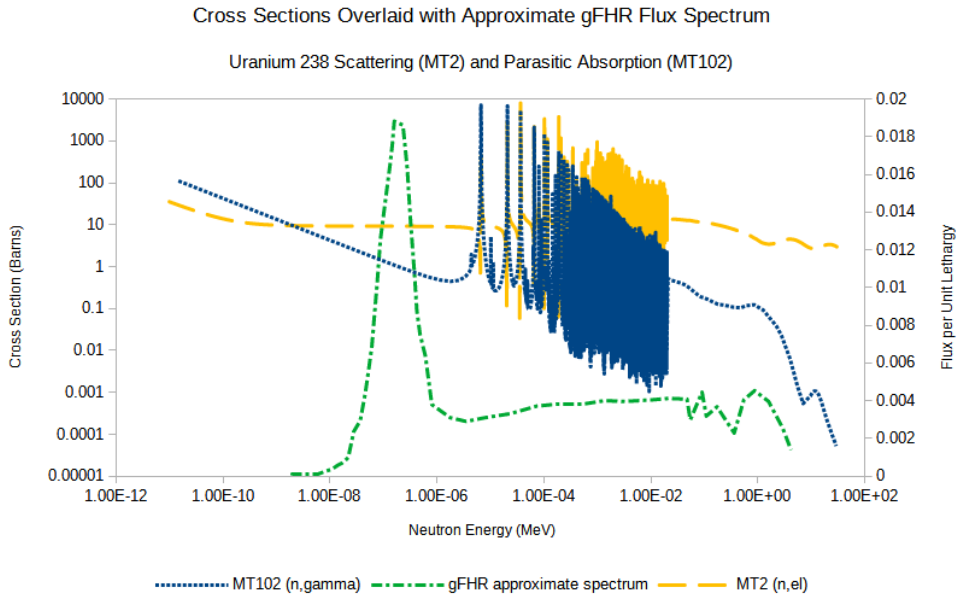


Figure 4.11: gFHR Flux Spectrum with U-238 Cross Sections Overlaid [Chadwick et al., 2006; Kile et al., 2022]

We could do the same other nuclides present when sufficiently high burnup occurs.

Last but not least, FLiBe resonances in the high energy (fast neutron) region are also important to capture. We pay special attention to the scattering and absorption resonances because they are important to FLiBe's role as a moderator. However, (n,2n) reactions are also considered, and these can be another group in itself [X. Wang, 2018]. Some of these resonances can be seen in Fluorine-19 and Lithium-7 cross sections which I have once more overlaid with the gFHR flux in Figure 4.12:

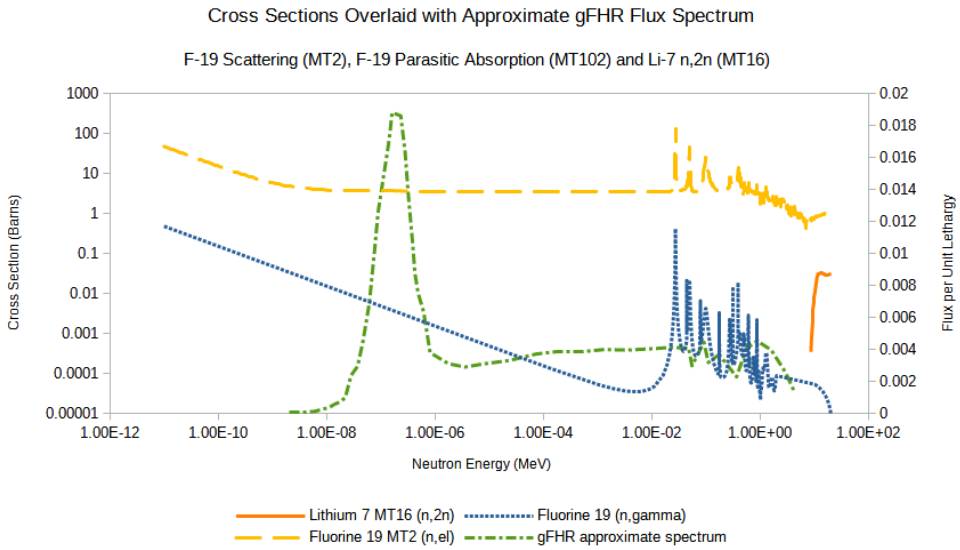


Figure 4.12: gFHR Flux Spectrum with Li-7 and F-19 Cross Sections Overlaid [Chadwick et al., 2006; Kile et al., 2022]

Lithium-7 and Fluorine-19 are two of several nuclides that make up FLiBe, but we can already see that these high level resonances for scattering and absorption are important for simulating the moderation process. As such, these would warrant one group on their own, and the (n,2n) reaction would warrant another energy group.

Spherical Harmonics Methods Supposing that angular discretisation is desired, the spherical harmonics method can be used. In the spherical harmonics (PN) method, where the angular flux ψ is periodic in polar and azimuthal angle, the flux is generally decomposed into a series of spherical harmonics [Duderstadt and Hamilton, 1976] usually denoted as Y_l^m. This is analogous to how periodic functions can often be broken down into sines and cosines using a Fourier Transform.

$$\psi(\vec{r}, \hat{\Omega}, E, t) = \sum_{l=0}^{\infty} \sum_{m=-l}^{m=1} \sigma_l^m(\vec{r}, E, t) Y_l^m(\hat{\Omega}) \tag{4.42}$$

Where σ_l^m are constants to be determined. One would substitute this expression for angular flux into the NTE and use the property of orthogonality to obtain a set of equations known as the spherical harmonics (PN) equations. Often, one might take only the first three terms of the expansion to form the P3 equations or even the first term to form the P1 equation. In fact, the diffusion approximation is also known as the P1 spherical harmonics approximation because the form of the P1 equations forms the basis for diffusion equations. More specifically, it shows gives a mathematical statement for linearly anisotropic flux and scattering cross sections. Of course, there are other assumptions required to convert P1 equations into the diffusion equations. We shall not cover this at the moment. While P1

equations, especially in their diffusion equation form, are applicable to general geometries, higher order equations such as P3 Spherical Harmonics equations are difficult to use for all but the simplest of geometries [Duerigen et al., 2010], eg. slab geometry.

Practically speaking, spherical harmonics can be used for complex geometries only if it were simplified in the form of the simplified spherical harmonics equations (SPN) methods. The SPN methods can be thought to be like the PN methods used for slab geometries, but extended to 3D in an ad hoc manner. One of the most commonly used forms of the SPN equations are the SP3 equations [Bahabadi, Pazirandeh, and Athari, 2015; Brantley and E. W. Larsen, 2000]. These SP3 equations can be shown to be an asymptotic correction to P1 [E. W. Larsen, J. E. Morel, and J. M. McGhee, 1993; E. Larsen, J. Morel, and J. McGhee, 1996; M. Modest and Lei, 2012]. Compared to multigroup diffusion, multigroup SP3 results in improved behaviour near highly absorbing regions, such as control rods [X. Wang, 2018]. Due to its popularity, these equations have been implemented in codes such as GeN-Foam [Fiorina, Hursin, and Pautz, 2017].

Fidelity for Angular and Energy Discretisation Given that the objective of this dissertation is to formulate a methodology to compress high fidelity CFD data into a transfer function for the purpose of controlling a heater in a simulated neutronics facility, the fidelity levels of the CFD models need not be very high. This is because the same transfer function methodology should be applicable to any CFD model so long as the inputs and outputs of the system are well monitored.

Hence, for a first iteration, the reactor model can be arbitrary and the future models of such a reactor can be subsequently refined to fit specific molten salt reactors, FHRs or even liquid metal cooled reactors (as long as there is minimal phase change).

However, the minimum requirement is that the modelling technique should be readily extensible to capture complex geometries, thermal inertia, and all other manner of phenomena related to anticipated transients without scram (ATWS).

Therefore, minimally, the neutronics model must incorporate spatial dependence, delayed precursors, show capability to incorporate multigroup models. However, while control rods are an integral part of the reactor, control rod oscillations are not necessary yet. Nevertheless should this work be complete, a model taking into account nonlinear anisotropy of flux such as SP3 should be the immediate next step. To facilitate this, a two group diffusion model must be used at minimum as it can handle these variations. For angular flux dependence, an SP3 model could be used in the immediate next steps but will be out of scope of this work.

Burnup, Reactor Poisons and Decay Heat

Now, these diffusion, SPN, SN and Monte Carlo modelling techniques do simulate energy, spatial and angular dependence of neutrons to differing degrees. However, it is often assumed that the reactor isotopic and chemical composition does not change significantly over the timescale of a transient provided the transient is short enough.

Should the transient occur over prolonged durations (hours, days or weeks), reactor poisons and fission products may build up over time. These can cause transients of their own. For example, reactor poisons such as Xenon-135 oscillations may be important when power transients are done on the timescale of several hours. Xe-135 is produced from the direct fission yield of a fissile isotope and the decay of I-135 [Lamarsh and Baratta, 2001]. Often, the fission yield of I-135 is several times the yield of Xe-135, it is about 4 times more for U-233, about 6 times more for Pu-239 and about 30 times more for U-235 [Lamarsh and Baratta, 2001]. Hence, the production of Xe-135 is largely governed by I-135 decay, which has a half life of 6.7 hrs [Lamarsh and Baratta, 2001]. Transients on such timescales, while important, do not threaten reactor safety as much as an unprotected reactivity transient, which can be on the timescale of seconds. This gives the operator very little time to react, and therefore, it is more important for operators to know how to handle these transients. Moreover, GeN-Foam does not have modelling capabilities for Xe-135 built in. While it is open source code and can be modified, the work required to develop such capabilities is too much for this dissertation. Furthermore, the methods for converting the high fidelity reactor models to low fidelity surrogate models would be similar regardless of whether Xe-135 concentrations are simulated. Hence, while Xe-135 transients are important they will not be included in the first few iterations of the reactor models.

Xe-135 is just one of many fission products which can affect core reactivity. The nuclear fission reactions can both breed new fuel and increase concentration of reactor poisons and transuranics. These are bound to be present with increasing burnup of the pebble fuel. Fuel burnup not only impacts reactivity, but doppler feedback coeffcients as well. In fact fuel temperature feedback coeffcients will also become more negative with burnup because of the buildup of transuranics [Fratoni, 2008].[7].

As explained previously, the aim of this demonstration is to investigate the possibility of reducing a high complexity multiphsyics model to a low fidelity data fit surrogate model for use in simulated neutronics facilities. While burnup and reactor poisons are an important consideration for reactor design, they may not be as specifically important for this endeavour. Hence, I have chosen to relegate investigation of these effects to future work. This would include any analysis of the effect of uneven heterogenous pebble burnup around the core and the effect of pebble hotspots. For simplicity, I will be working with fresh cores. The fresh core assumption also means that simulating decay heat is out of scope for the first iteration of this work. It will be worthwhile investigating the impacts of burnup and decay heat in future work however.

[7]At 425 μ m diameter at C/HM ratio of 360, -2.75 pcm/K of doppler feedback is expected at 0 GWd/tHM (gigawatt-day per ton heavy metal), but -4.5 pcm/K doppler feedback $\frac{\Delta\rho}{\Delta T_{fuel}}$ is expected at about 120 GWd/tHM burnup. The prevailing enrichment here is about 10%.[Fratoni, 2008]

Structural Expansion

Structural expansion can be considered a material and thermal hydraulics phenomena since it deals with expansion of solid structures, and this depends on the material used to construct the reactor core as well as the temperatures locally at each point of the core containment structure and supports. This affects neutronics [T. Allen et al., 2013] as well since an expanded core means more leakage would occur, thus reactivity decreases. This is potentially concerning because graphite, used as a key structural material in FHR cores, has a thermal coefficient of expansion of about $5.2 * 10^{-7}/K$ in the a direction and $2.8 * 10^{-5}/K$ in the c direction at 873 K [Tsang et al., 2005]. Depending on the direction of thermal expansion, graphite has a comparable thermal expansion coefficient to steel which used for fast reactor structural material [Caro et al., 2013]. Steel has an estimated thermal expansion coefficient of $1.8 * 10^{-5}/K$ at about 873K [Fu, X. D. Li, and Hwang, 2011]. Therefore, at typical FHR operating temperatures, graphite potentially has a coefficient of thermal expansion in a similar order of magnitude to steel. Of course, the alloying elements and composition of the steel would affect this coefficient, but it should not be nought to change this order of magnitude. Hence, thermal expansion in FHR cores and fast reactor cores could be of a similar order of magnitude. This may make thermal expansion a potentially significiant phenomena to address.

Nevertheless, neutronics and structural mechanics coupling for short timescales was considered to be not as critical as neutronics and thermal hydraulics coupling and thus not considered for FHR modelling [T. Allen et al., 2013]. This is because the key factors reducing reactivity during a spike in core power would be the expansion of FLiBe, otherwise known as coolant void feedback [X. Wang, 2018; T. Allen et al., 2013] and fuel temperature feedback [X. Wang, 2018]. Reactivity feedback due to structural expansion could be neglected in comparison to these other feedback mechanisms. We could of course verify this in another study, but this is beyond the scope of this dissertation and therefore, not included.

Over time, of course, irradiation of graphite and materials can cause expansion, and this too would change the structural configuration of the reactor. This would in turn affect the thermal hydrualics of the core. However, these effects occur over several weeks, months and years [Franklin, Lucas, and Bement, 1983] and not on the same timescale as ATWS and other transients. Therefore, these effects are neglected for this dissertation.

Coolant Expansion and Void Feedbacks

When FLiBe heats up, its density is reduced. The reduced salt density results in lower parasitic absorption from the salt, thus increasing reactivity. This is not a desirable result as we want to have increasing coolant temperatures result in a net negative coolant temperature feedback. To achieve this, FHRs are designed to be undermoderrated [Greene et al., 2010]. If the core is undermoderated, then FLiBe expansion would result in negative reactivity feedback [T. Allen et al., 2013]. In undermoderated cores, bubbles or voids in FLiBe would also result in the same negative reactivity feedback due to decreased moderation. The degree

of undermoderration would depend on the carbon to fuel (heavy metal) ratio. The carbon to fuel ratio should be adjusted in FHRs such that the negative reactivity feedback from undermoderration exceeds the positive reactivity feedback from reduced parasitic absorption [Greene et al., 2010]. Given that coolant temperature feedback plays such a large role in FHR reactivity feedback mechanism, it is quite important to model this phenomena [X. Wang, 2018]. Therefore, for this work, we aim to model reactivity effects of the FLiBe thermal expansion.

However, any two phase flow phenomena due to gas entrainment will not be modelled in this work as multiphase flow adds extra complications. As the study of this effect is not key to this present dissertation, study of gas bubble and other such multiphase effects on neutronics is relegated to future work.

Double Heterogeneity of TRISO Pebble Bed

One other important neutronics problem to solve for pebble bed FHRs would include simulating the doubly heterogeneous geometry of pebble beds with TRISO fuel [Lou, Yao, et al., 2020; Fratoni, 2008]. This is challenging because the simulation requires too much computational resources to be practical if a brute force Monte Carlo approach is used. Of course, if one has enough computational resources, one can simulate full cores with doubly heterogeneous geometry fully preserved as was the case for the High Temperature Reactor 10 (HTR-10) test reactor MCNP simulation [Abedi and Vosoughi, 2012]. This HTR-10 simulation only took into account neutronics for the HTR-10 and did not couple it with thermal hydraulics. When thermal hydraulics is involved, there is additional computational burden that is added.

For coupled thermal hydraulics (porous media) and neutronics calculations for doubly heterogenous geometry such as those found in the Thorium Molten Salt Reactor Solid Fuel (TMSR-SF) [Aufiero and Fratoni, 2016] and the generic FHR (gFHR) developed by Kairos Power [Satvat et al., 2021]. For these calculations such as those in the TMSR-SF, high fidelity calculations were performed where the each pebble with its TRISO particles was fully simulated in Monte Carlo Code [Aufiero and Fratoni, 2016]. Of course, the added computational burden necessitated a supercomputer and techniques to speed up calculations in doubly heterogeneous geometries. For Serpent, an algorithm known as Woodcock Delta Tracking was used for its Monte Carlo simulations [Aufiero and Fratoni, 2016]. This Delta Tracking is exceptionally important for doubly heterogenous geometries where the conventional ray tracing method would require copious amounts of computational time to calculate distances between surfaces [Leppänen, 2010]. While Delta tracking speed up the TMSR-SF simulation, calculations still required 20 cores in an Intel(R) Xeon(R) CPU E5-2670 v2 with clock frequencies of 2.50GHz [Aufiero and Fratoni, 2016]. Essentially, we have to use supercomputers for such simulations. If one desires to save computing power, then we may want to use deterministic methods to speed up the neutronics simulations.

Even if we were to use deterministic methods to simplify the problem, we would normally have to use Monte Carlo simulations to generate multi group cross sections (MGXS) so that

these deterministic calculations[8] can be performed. And for PRKE, we would need Monte Carlo methods to help generate some form of reactivity feedback. Either way, Monte Carlo methods are going to be used. Hence, we will focus the bulk of the discussion on Monte Carlo rather than deterministic methods.

For Monte Carlo methods for pebble beds, additional techniques besides delta tracking are often used to speed up calculations. One such technique was used by Kairos Power in their simulation of the gFHR using the Kairos Power Advanced Core Simulator (KPACS) [Satvat et al., 2021]. In KPACS simulations of the gFHR, each pebble is explicitly simulated. However, taking inspiration from the Very Superior Old Programs (VSOP) [Satvat et al., 2021] used for pebble bed high temperature reactors (HTRs) [Rütten et al., 2005], the core is split into spectral zones where the spatial variations in neutron spectrum and temperature are small within these zones [Satvat et al., 2021]. Each zone has a representative pebble simulated using Serpent 2, and coupled to the thermal hydraulics StarCCM+ code using an in house wrapper called KPATH.As each representative pebble is explicitly modelled with its TRISO particles, the self shielding effects within each pebble are preserved. At the same time, the entire pebble core is not explicitly simulated, thus saving plenty of simulation time while coupling Monte Carlo solvers with the appropriate thermal hydraulics solver. This zoning method for FHRs bears conceptual resemblance to mesh coarsening. In this case, the zones of the core can be thought of as an extremely coarse mesh. As such, this method falls under the category of hierarchical surrogate modelling. This method was specifically used for burnup calculations [Satvat et al., 2021] but has the potential to be used for MGXS calculation as well.

If an alternate form of simplification is desired for Monte Carlo simulation, then some form of homogenisation can be made. One should note though, that information will be lost during homogenisation, and it is generally not suitable for high fidelity or ultra high fidelity simulation. However, for the purposes of this dissertation, it does suffice since we are only constructing an arbitrary reactor with which to extract prototypical reactor feedback behaviour. Now, there are several methods of hierarchical surrogate modelling which can save time required for Monte Carlo simulation. The simplest case involves full homogenisation of the TRISO fuel kernel, with its layers, into the surrounding carbon matrix [Fratoni, 2008], this "full homogenisation" is sometimes known in literature as the volumetric homogenisation model (VHM) [Lou, Yao, et al., 2020]. VHM brings about a significant amount of deviation for k_∞ of the homogenised pebble compared to the actual TRISO fuel pebble chiefly due to the reduction in spatial self shielding. A ballpark figure for some models is that deviations in k_∞ can be as much as 6% [Fratoni, 2008].

Secondly, we can also perform homogenisation such that the coatings of the TRISO particles are mixed with the surrounding carbon matrix, and the fuel kernels are left as they are [Fratoni, 2008]. The k_∞ of these models in literature are sometimes not statistically different

[8] For deterministic codes, one must use Dancoff Factors [Fratoni, 2008; Bende et al., 1999] for double Heterogeneity problems, but they are not relevant for use in Monte Carlo simulations. They are relevant if one wanted to use deterministic codes to simulate the fuel and moderator regions separately rather than lumping them together [Kloosterman and Ougouag, 2005].

from the original TRISO fuel pebble model being within 2 σ[Fratoni, 2008], where σ in this context is the standard deviation of the Monte Carlo k_∞ calculation. Generally speaking, if we allow complete homogenisation of TRISO fuel kernels and graphite pebble materials, k_{eff} would be underestimated by about 6% [Fratoni, 2008]. However, if we simulate homogenisation of TRISO fuel shells with the pebble and leave the uranium fuel intact, k_{eff} would only be underestimated by 0.2% [Fratoni, 2008]. This is mainly due to the self shielding effects of the fuel within the pebble [Fratoni, 2008]. The former method may produce results too far (>1%) from the benchmark k_{eff} while the latter result may still take excessively long for a full core simulation as it was reported to save about 25% of Monte Carlo simulation time [Fratoni, 2008]. Hence, we may want to look for another method for faster simulation.

A third method of homogenisation is known as the Reactivity Equivalent Physical Transform (RPT) [Y. Kim and Baek, 2005; Lou, Yao, et al., 2020]. This modifies the VHM approach by first concentrating the fuel kernels into a smaller area within the pebble. Only after this process will we then homogenise the section within the pebble where TRISO fuel kernels are. Compared to VHM, RPT was reported to have better captured the spatial self shielding effects of TRISO particles [Y. Kim and Baek, 2005]. The RPT process is illustrated in Figure 4.13.

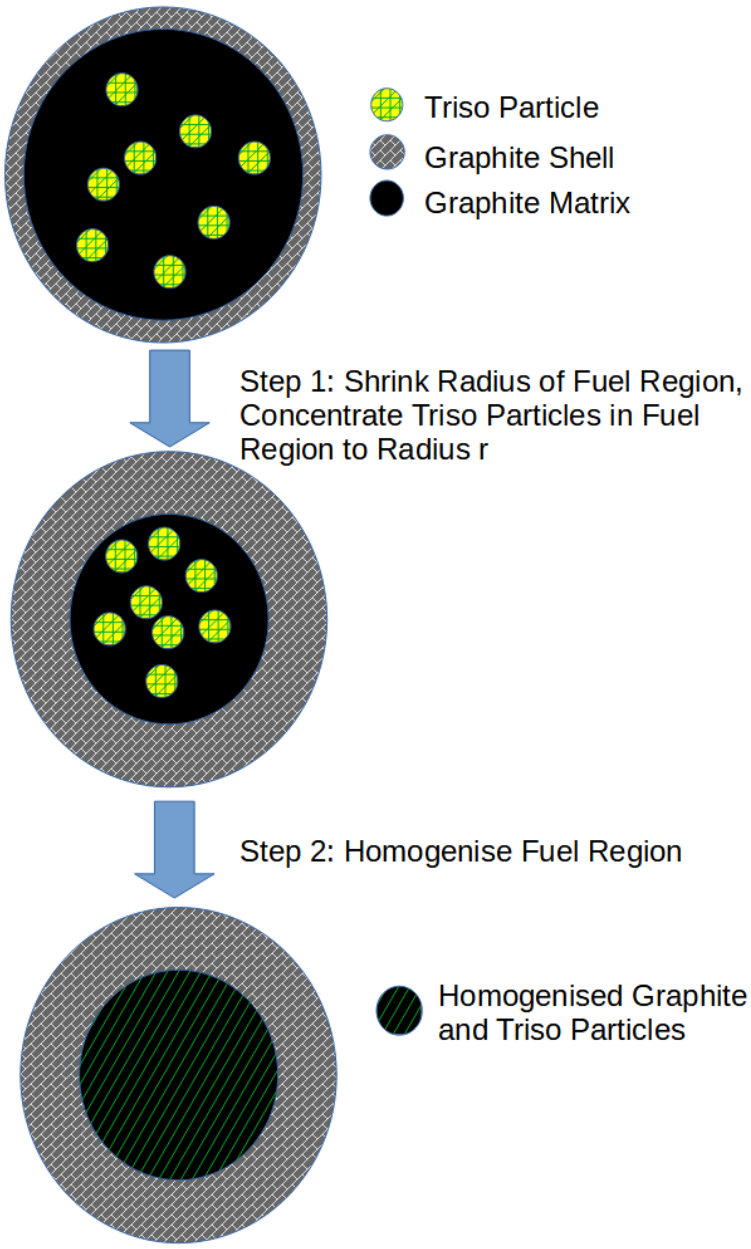


Figure 4.13: Classical RPT Method

In Figure 4.13, the graphite shell and graphite matrix are essentially the same material. I just differentiate it to distinguish the fuel region from the non fuel region. Secondly, TRISO particles are not drawn to scale, and their respective layers are also not drawn for simplicity.

The radius of this concentrated cylinder or sphere of homogenised TRISO fuel or matrix is adjusted until the k_∞ matches that of the original pebble or geometry. This process allows the user to control the amount of self-shielding experienced by the fuel so that the neutron spectrum more closely represents that of explicitly simulated doubly heterogeneous geometry [Y. Kim and Baek, 2005]. In particular, the self-shielding effect is increased by reducing the radius of this smaller area [Y. Kim and Baek, 2005]. The self-shielding effect is strongest if all the fuel is concentrated into a single sphere (or cylinder) in its centre [Lou, Chai, et al., 2020]. As the fuel gets more dispersed, the self shielding decreases. The self shielding effects in these TRISO particles are somewhere between the case of complete dispersion (VHM) and no dispersion (where all fuel is concentrated in the center of the pebble). After homogenisation using RPT, we can verify whether the self shielding is accurately replicated by inspecting the neutron spectrum [Y. Kim and Baek, 2005] or even important terms within the six factor formula such as resonance escape probability [Y. Kim and Baek, 2005; Duderstadt and Hamilton, 1976; Lou, Chai, et al., 2020]. This technique was used successfully for pebble beds within the very high temperature reactor (VHTR) [Noh et al., 2008].

Despite this success, RPT methods were found to be inaccurate for certain burnup calculations involving dispersed burnable fuel particles within the fuel pebble [Lou, Yao, et al., 2020]. As a result, the RPT method was heavily modified to give rise to the Ring RPT method [Lou, Yao, et al., 2020]. Here, the TRISO particles are compressed into a ring (for cylindrical geometry) or shell (for spherical geometry). A rough schematic can be shown in Figure 4.14:

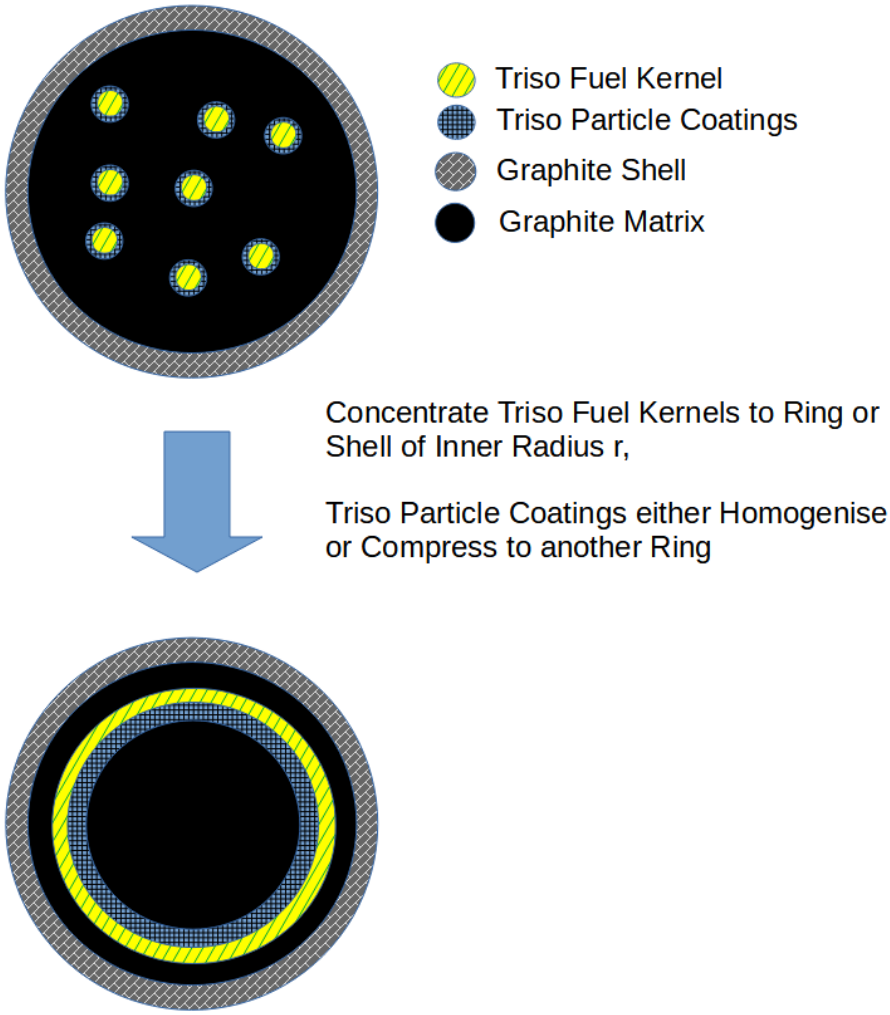


Figure 4.14: Ring RPT Method Rough Schematic

Similar to RPT, the user is able to adjust the level of self-shielding by changing the radius of the ring or shell [Lou, Chai, et al., 2020]. Spatial self shielding effects are strongest if the fuel is concentrated in the centre of the fuel pin or fuel pebble, and these effects will

weaken as the user increases the radius of the ring or shell so as to more evenly disperse the fuel within the homogenised pebble. In Ring RPT, the inner radius of this ring or shell is adjusted until k_∞ of the Ring RPT pebble matches that of the original pebble [Lou, Yao, et al., 2020]. Once more, we can check if the self-shielding is adequately replicated by inspecting the neutron spectrum of the Ring RPT pebble and the explicitly modelled pebble.

For the shell layers of the TRISO kernel, it may matter little whether it is homogenised with the matrix or compressed into its own shell layers. This is because the homogenisation of TRISO shells with the pebble graphite matrix would only change k_{eff} of the pebble by 0.2% [Fratoni, 2008]. One can of course compensate for this small change by adjusting the radius until k_∞ is the same as the reference TRISO pebble. Therefore, one may not need to worry about The Ring RPT method was found to be more practical for cylindrical geometries compared to spherical geometries of small TRISO fuel sizes with burnable poisons [Lou, Chai, et al., 2020].

For Ring RPT, however, the degree of fuel dispersion achievable is largely limited by the outer radius of the pebble. Therefore, there is a lower limit for the amount of spatial self shielding the Ring RPT can replicate [Lou, Chai, et al., 2020]. When the Ring RPT radius is maximised, the degree of spatial self shielding is still higher than that of fully homogenised fuel. For doubly heterogeneous fuel, the degree of self shielding may be lower than that of the lowest self shielding the Ring RPT can replicate. This problem is especially apparent for pebble fuel [Lou, Chai, et al., 2020]. An ad hoc solution for this is to partially homogenise the TRISO fuel kernels with the surrounding layers of pyrolytic carbon, silicon carbide and buffer layers and perform Ring RPT on this homogenised fuel. Doing so would lower the threshold of spatial self shielding that can be simulated using Ring RPT such that k_∞ matches that of the explicitly simulated TRISO pebble. This could perhaps ensure that Ring RPT is usable for pebble fuel.

Now, supposing one wanted to use MGXS generated from these Monte Carlo simulations for deterministic multiphysics simulations, we could either opt to homogenise the entire core, homogenise across fuel elements or not homogenise at all. If we were not to homogenise between fuel and moderator regions, Dancoff factors become important [Bende et al., 1999; Fratoni, 2008; Kloosterman and Ougouag, 2005]. However, for simplicity, I will homogenise temperature dependent cross sections across the core in this work. I would calculate the cross sections such that reaction rates over the whole core are preserved. This way, the heat generation outputs are preserved, which is most important for the thermal hydrualics and neutronics coupling.

Thermal Hydraulics Modelling Equations

For thermal hydraulics, we are primarily interested in mass, energy and momentum balances in the fluid and conjugate heat transfer between fluids and solids. The Navier Stokes equations describes mass and momentum balance in the fluid (assuming it is Newtonian). The incompressible Navier Stokes equations can be written as [Tu et al., 2023]:

$$\mu\nabla^2 u - \frac{\partial P}{\partial x} + \rho g_x = \rho(\frac{D}{Dt}u) \tag{4.43}$$

$$\mu\nabla^2 v - \frac{\partial P}{\partial y} + \rho g_y = \rho(\frac{D}{Dt}v) \tag{4.44}$$

$$\mu\nabla^2 w - \frac{\partial P}{\partial z} + \rho g_z = \rho(\frac{D}{Dt}w) \tag{4.45}$$

$$\nabla \bullet (\vec{v}) = 0 \tag{4.46}$$

Where u, v and w are x, y and z components of fluid velocity, P is fluid pressure, g_i is gravitational forces or body forces in direction i. $\frac{D}{Dt}$ represents the total derivative which includes time dependent and advection terms. ρ is fluid density. μ is fluid dynamic viscosity which is assumed constant.

The form of the Navier Stokes equations 4.43, 4.44, 4.45 is for incompressible Newtonian fluids. Hence, incompressibility is generally assumed, and the mass conservation equation is represented by the continuity equation, equation 4.46. The incompressibility assumption is generally true for molten salts except in the case where natural convection comes into play. Also the molten salt in this case can be assumed to be Newtonian in nature. This assumption can be supported since it has been shown that the salt BeF_2 behaves like a Newtonian fluid [Moynihan and Cantor, 1968].

For thermal transport, energy is also transported by the fluid. Therefore, an energy transport equation is also given [Bejan, 2013].

$$\rho\frac{De}{Dt} + e\left(\frac{D\rho}{Dt} + \rho\nabla \bullet \vec{v}\right) = -\nabla \bullet \vec{q''} + q''' - P\nabla \bullet \vec{v} + \mu\Phi \tag{4.47}$$

Where $\mu\Phi$ is the dissipation term, where mechanical work converts into heat, $P\nabla \bullet \vec{v}$ constitutes part of flow work and $\frac{D}{Dt}$ represents the substantial or total derivative. q''' represents a heat source term which can represent the combined effects of heat generation within the fluid due to nuclear or chemical interactions, and also due to radiation heat transfer. e is the internal energy and q'' is the heat flux.

If in terms of enthalpy h,

$$\rho\frac{Dh}{Dt} = -\nabla \bullet \vec{q''} + q''' + \mu\Phi + \frac{DP}{Dt} \tag{4.48}$$

If one wishes to have an equation explicitly in temperature,

$$\rho c_p\frac{DT}{Dt} = -\nabla \bullet \vec{q''} + q''' + \mu\Phi + \beta T\frac{DP}{Dt} \tag{4.49}$$

In equation 4.49 β represents coefficient of thermal expansion [Bejan, 2013], not delayed neutron fraction.

Simplifications can then be made to this equation using Fourier's law, and assuming the dissipation term contributes negligibly to the rise in temperature of the fluid. Fourier's law of conduction can be written as [Bejan, 2013]:

$$\vec{q''} = k\nabla T \tag{4.50}$$

Where k in equation 4.50 is thermal conductivity of the medium, which can be fluid in this case or solid in a solid medium. Furthermore, we can also assume that temperature rises negligibly due to compressive forces in FHRs as the pressure variations are negligible compared to the heat fluxes and the compressibility of the fluid is negligible [Bejan, 2013] .

$$\rho c_p \frac{DT}{Dt} = -\nabla \bullet (k\nabla T) + q''' \tag{4.51}$$

In the solid, only usually heat conduction dominates.

$$\rho c_p \frac{dT}{dt} = -\nabla \bullet (k\nabla T) + q''' \tag{4.52}$$

Turbulence Modelling

In FHRs and FHR cores, there are a few key phenomena we need to take into account. First of which is turbulence modelling if the Reynold's Number (Re) exceeds a threshold value. Re represents the ratio of inertial forces to viscous forces within a fluid. To account for turbulence, there are several approaches.

Firstly, if we were to use brute force, then the Navier Stokes equations are solved directly in what is known as Direct Numerical Simulation (DNS). For DNS, it is common to use spectral decomposition methods which break the velocity and temperature fields down into orthogonal polynomials such as the Chebyshev Polynomials [Kasagi and Nishimura, 1997]. Of course, modelling turbulence in such a manner is extremely intensive on computational resource and time, and is usually impractical for reactor simulation.

Reynolds Averaged Navier Stokes (RANS) To simplify this, the turbulence is usually modelled using closure models. There are three classes of models. Firstly, the Reynold's Averaged Navier Stokes (RANS) methods [Tu et al., 2023; Pope, 2000]. For this, we consider the instantaneous velocity u equal to a time averaged velocity $\bar{u}$ and the fluctuating velocity u'. Taking the x component of velocity u:

$$u = \bar{u} + u' \tag{4.53}$$

Substituting this into the incompressible Navier Stokes equation, we can show that we get:

$$\mu\nabla^2\bar{u} - \frac{\partial\bar{P}}{\partial x} + \rho g_x = \rho\left(\frac{D}{Dt}\bar{u}\right) + \rho\left(\frac{\partial\overline{(u'u')}}{\partial x} + \frac{\partial\overline{(u'v')}}{\partial y} + \frac{\partial\overline{(u'w')}}{\partial z}\right) \tag{4.54}$$

Closure Models The $\left(\frac{\overline{\partial(u')^2}}{\partial x} + \frac{\overline{\partial(u'v')}}{\partial y} + \frac{\overline{\partial(u'w')}}{\partial z}\right)$ are called the Reynold's stress terms. One cannot solve the turbulent Navier Stokes equations without knowledge of these Reynold's stresses. This is known as the closure problem [Bejan, 2013].

RANS methods such as $k-\varepsilon$ model solve the closure problem by assuming the Reynold's stresses contribute to a time-averged, effective turbulent kinematic viscosity ν_t [Pope, 2000; Bejan, 2013]. Thus, the Reynold's stresses for velocity in direction i u_i and direction j u_j can be expressed as [Tu et al., 2023]:

$$-\overline{u_i' u_j'} = \nu_t \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i}\right) - \frac{2}{3} k \delta_{ij} \tag{4.55}$$

Where x_i and x_j are i and j components of length. δ_{ij} is the Kronecker delta and k is turbulent kinetic energy. It can be expressed in Cartesian coordinates as:

$$k = \frac{1}{2}\left((u')^2 + (v')^2 + (w')^2\right) \tag{4.56}$$

And the turbulent kinematic viscosity ν_t (assumed isotropic) can be modelled as:

$$\nu_t = C_\mu \frac{k^2}{\varepsilon} \tag{4.57}$$

C_μ is tested with DNS of turbulent channel flow to be almost constant at 0.09 [Pope, 2000] everywhere except for the boundary layer. ε is the rate of dissipation of k which can be written as [Tu et al., 2023]:

$$\varepsilon = \nu_t \overline{\left(\frac{\partial u_i'}{\partial x_j} + \frac{\partial u_j'}{\partial x_i}\right)} \tag{4.58}$$

$k-\varepsilon$ equation The values k and ε are modelled using the equations [Tu et al., 2023]:

$$\frac{Dk}{dt} = \nabla \frac{\nu_t}{\sigma_k} \bullet \nabla k + P - \varepsilon \tag{4.59}$$

$$\frac{D\varepsilon}{dt} = \nabla \frac{\nu_t}{\sigma_\varepsilon} \bullet \nabla + \frac{\varepsilon}{k}\left(C_{\varepsilon 1} \varepsilon P - C_{\varepsilon 2} \varepsilon\right) \tag{4.60}$$

Where σ_k has been data fitted to be 1.0, σ_ε has been fitted to be 1.3, $C_{\varepsilon 1}$ is fitted to be 1.44 and $C_{\varepsilon 2}$ is fitted to 1.92 [Tu et al., 2023].

P is the production term for k denoted as [Tu et al., 2023] [9]:

$$P = 2\nu_t \nabla \vec{\bar{u}} \bullet \nabla \vec{\bar{u}} + \nu_t \left[\left(\frac{\partial \bar{u}}{\partial y} + \frac{\partial \bar{v}}{\partial x}\right)^2 + \left(\frac{\partial \bar{v}}{\partial z} + \frac{\partial \bar{w}}{\partial y}\right)^2 + \left(\frac{\partial \bar{w}}{\partial x} + \frac{\partial \bar{u}}{\partial z}\right)^2\right] \tag{4.61}$$

[9] Tu wrote this in his textbook for 2D coordinates [Tu et al., 2023], I just extended it to 3D cartesian coordinates

Thus, the RANS $(k-\varepsilon)$ equations can be rewritten as:

$$\nabla(\mu+\mu_t)\bullet\nabla\bar{u}-\frac{\partial\bar{P}}{\partial x}+\rho g_x=\rho\left(\frac{\bar{D}}{\bar{D}t}\bar{u}\right) \tag{4.62}$$

$$\nabla(\mu+\mu_t)\bullet\nabla\bar{v}-\frac{\partial\bar{P}}{\partial x}+\rho g_y=\rho\left(\frac{\bar{D}}{\bar{D}t}\bar{v}\right) \tag{4.63}$$

$$\nabla(\mu+\mu_t)\bullet\nabla\bar{w}-\frac{\partial\bar{P}}{\partial x}+\rho g_z=\rho\left(\frac{\bar{D}}{\bar{D}t}\bar{w}\right) \tag{4.64}$$

Where

$$\mu_t=0.09\rho\frac{k^2}{\varepsilon} \tag{4.65}$$

And we find the equations for k and ε in equation 4.59, 4.60 and 4.61. The substantial derivative $\frac{\bar{D}}{\bar{D}t}$ uses averaged velocities to account for advection rather than instantaneous velocities.

Law of the Wall Of course, nearer the wall, different models must be used to model the boundary layer. Here, empirical correlations are devised called the law of the wall [Bejan, 2013] which can be written as:

$$u^+=\begin{cases}y^+, & \text{if } y^+<5\\ 5\ln y^+-3.05 & \text{if } 5<y^+<30\\ 2.5\ln y^++5.5 & \text{if } y^+>30\end{cases} \tag{4.66}$$

Where

$$u^+=\frac{u}{u_*} \tag{4.67}$$

$$y^+=\frac{yu_*}{\nu} \tag{4.68}$$

u_* is known as the friction velocity [Bejan, 2013]:

$$u_*=\left(\frac{\tau_{wall}}{\rho}\right)^{0.5} \tag{4.69}$$

The wall shear stress in 2D flows can be shown to be:

$$\tau_{wall}=\rho\nu\frac{\partial\bar{u}}{\partial y} \tag{4.70}$$

Heat Equations For fluids with heat transfer, a similar time averaging approach can done for RANS [Bejan, 2013]:

$$\nabla k \bullet \nabla \bar{T} = \rho c_p \left(\frac{\bar{D}}{\bar{D}t} \bar{T} \right) + \rho c_p \left(\frac{\partial \overline{(T'u')}}{\partial x} + \frac{\partial \overline{(T'v')}}{\partial y} + \frac{\partial \overline{(T'w')}}{\partial z} \right) \tag{4.71}$$

A similar approach can be used for closure models, to the effect that turbulence apparently increases thermal diffusivity α rather than momentum diffusivity ν.

$$\nabla(\alpha + \alpha_t) \bullet \nabla \bar{T} = \left(\frac{\bar{D}}{\bar{D}t} \bar{T} \right) \tag{4.72}$$

Where $\alpha = \frac{k}{\rho c_p}$ and α_t is known as turbulent thermal diffusivity or eddy thermal diffusivity [Bejan, 2013]. To solve the RANS equations, it is common to assume a constant turbulent Prandtl Number Pr_t [Bejan, 2013]:

$$\mathrm{Pr}_t = \frac{\nu}{\alpha} \approx 0.9 \tag{4.73}$$

Similar to the velocity fields, thermal energy transport also requires an empirical law of the wall. This can be written as [Bejan, 2013]:

$$T^+ = \begin{cases} \mathrm{Pr}\, y^+, & \text{if } y^+ < 13.2 \\ 2.195 \ln y^+ - 13.2\,\mathrm{Pr} - 5.66, & \text{if } y^+ > 13.2 \end{cases} \tag{4.74}$$

$$T^+ = (\bar{T} - T_{wall}) \frac{\rho c_p u_*}{q''_{wall}} \tag{4.75}$$

Where Pr is the Prandtl Number, $\mathrm{Pr} = \frac{\nu}{\alpha}$, T_{wall} is wall temperature, q''_{wall} is wall heat flux. These only hold for $0.5 < \mathrm{Pr} < 5$.

For molten salts such as FLiBe, in the FHR, the Prandtl Number is much higher, around $12 < \mathrm{Pr} < 24$ [Nicolas Zweibaum, 2015; Nguyen and Merzari, 2023]. For these ranges, a different law of the wall would hold [Gowen and J. Smith, 1967]. Usually, experiments or DNS [Nguyen and Merzari, 2023] can be done to determine a law of the wall in this range. They would then have to be fitted to empirical correlations such as [Bejan, 2013]:

$$T^+ = \begin{cases} \mathrm{Pr}\, y^+, & \text{if } y^+ < y^+_{csl} \\ \mathrm{Pr} \ln y^+_{csl} + \frac{\mathrm{Pr}_t}{\kappa} \ln \frac{y^+}{y^+_{csl}}\,\mathrm{Pr}, & \text{if } y^+ > y^+_{csl} \end{cases} \tag{4.76}$$

y^+_{csl} is dimensionless thickness of the conduction sublayer (CSL) and Pr_t is the turbulent Prandtl number $\mathrm{Pr}_t = \frac{\nu_t}{\alpha_t}$. κ is an empirical constant to be fitted.

Now, these near wall models are useful but cannot apply in all situations. We developed these equations without accounting for natural convection. Natural convection can interact with forced convection to give rise to mixed convection which will dampen or increase local turbulence. These mixed convection effects are important in the downcomer of FHRs

[Nguyen and Merzari, 2023]. Therefore, DNS studies were performed to study these effects [Nguyen and Merzari, 2023].

Situations like these show that RANS cannot always be trusted to give accurate results.

Large Eddy Simulations (LES) RANS is often not able to perform well in regions of adverse pressure gradients, regions of separated flow [Alam, Thompson, and Walters, 2017], where vortex shedding occurs [Pope, 2000], or as we previously discussed, regions of natural convection [Nguyen and Merzari, 2023]. In these regions, should we want to use a higher fidelity method but want to avoid DNS, then the prospect of Large Eddy Simulations (LES) would become attractive [Alam, Thompson, and Walters, 2017]. In LES, the larger turbulent eddy flows directly simulated, whereas the smaller eddies are treated using constituitive models [Pope, 2000]. For LES, it is common to set the simulation such that about 80% of the turbulent kinetic energy is directly resolved [H. Isaksson, 2019].

As opposed to RANS, LES does not average out all the turbulent eddies, instead, the common approach is to decompose spatial oscillations into various frequencies similar again to what a Fourier Transform does [Sagaut, 2005]. We then assign a cutoff frequency based on the attempting to directly 80% of the turbulent kinetic energy, and we decide a mesh or grid size based on them. In effect, we are applying a low pass filter to the Navier Stokes equations [Meyers, Geurts, and Sagaut, 2007]. The frequencies that are modelled, also known as subgrid modes [Sagaut, 2005], can modelled using subgrid viscosity models [Sagaut, 2005]. These include the Smagorinsky model [Meyers, Geurts, and Sagaut, 2007] among many others [Sagaut, 2005]. Unlike RANS, wall models or wall functions are only employed sometimes, some LES models do not contain wall functions. One example is the Wall Adapting Eddy Viscosity (WALE) model which does not require wall functions to model the transition region between bulk fluid and near wall viscous boundary layers [Nicoud and Ducros, 1999; Weickert et al., 2010].

LES models have been used in gas liquid separators for molten salt reactors (MSRs) [J.-J. Li et al., 2018] and molten salt flow around graphite blocks [Merzari et al., 2020], thus showing that they are well established for MSR research in literature. LES models may prove to work well provided the grid resolution is fine enough, however the computational cost is still high compared to RANS.

Hybrid RANS-LES Methods Hybrid RANS-LES models were developed to leverage the low cost of RANS near boundary layers and the suitability of LES for flow separation [Spalart, 1997]. These were known as Detached Eddy Simulations (DES) [Spalart, 1997]. A common problem was controlling when and how the RANS to LES transition occurred as this could impact accuracy [Spalart et al., 2006]. The second issue was improving the wall modelling [Shur et al., 2008]. Tweaks were made to DES simulations over time to solve these problems and this resulted in Delayed Detached Eddy Simulation (DDES) [Spalart et al., 2006] and Improved Delayed Detached Eddy Simulation (IDDES) [Shur et al., 2008]. IDDES

methods were used to simulate flow around pebble beds in the gas cooled reactors [Shams et al., 2015].

Unfortunately, LES and hybrid RANS LES methods can still be time consuming and computational resource heavy. These can take away computational resources required for simulating other phenomena such as neutron transport. Nevertheless, they can be used to verify or validate empirical correlations used for porous media such as the Wakao Correlation [Dave, Sun, and L. Hu, 2020].

Porous Media Modelling

Fluid Equations

Using turbulence modelling to directly simulate fluid flow for heat transfer within the core is computationally expensive. This is why we often resort to use a simpler model such as the porous medium equations [C. Wu et al., 2010]. Porous media equations regard that within each volume element, there exists two separate phases of solid and fluid. The solid is often referred to as the subscale structure [Fiorina, I. Clifford, et al., 2015]. The porous media Navier stokes equations as found in multiphysics code such as GeN-Foam can be written as [Fiorina, I. Clifford, et al., 2015]:

$$\frac{\partial \gamma \rho}{\partial t} + \nabla \bullet (\gamma \rho \mathbf{u}) = 0 \tag{4.77}$$

$$\frac{\partial \gamma \rho \mathbf{u}}{\partial t} + \nabla \bullet (\gamma \rho \mathbf{u} \otimes \mathbf{u}) = \nabla \bullet (\mu_T \nabla \mathbf{u}) - \nabla \gamma p + p_i \nabla \gamma + \gamma \mathbf{F_g} + \gamma \mathbf{F_{SS}} \tag{4.78}$$

The energy equation of the fluid in GeN-Foam is written as [Fiorina, I. Clifford, et al., 2015]:

$$\frac{\partial \gamma \rho e}{\partial t} + \nabla \bullet (\mathbf{u} \gamma (\rho e + p)) = \nabla \bullet (\gamma k_T \nabla T) + \gamma \mathbf{F_{SS}} \bullet \mathbf{u} + \gamma \dot{Q}_{SS} \tag{4.79}$$

Where $\otimes$ is the tensor product for column vectors $\mathbf{u}$ and $\mathbf{v}$:

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^T$$

Where ρ is fluid density, γ is porosity of the medium otherwise known as volume fraction. $\mathbf{u}$ represents the interstitial velocity of the fluid in the porous medium. p represents fluid pressure, μ_T represents the turbulent dynamic viscosity $Pa \bullet s$ and μ represents the dynamic viscosity[10]. p_i is the interfacial pressure. One may note the lack of a dynamic viscosity term μ in the equations for momentum diffusion because momentum more often diffuses into the

[10]There seem to be some possible errata in Fiorina's paper as read in May 2023. First, μ_T was stated as turbulent kinematic viscosity but had units of $Pa \bullet s$, which is a unit for dynamic viscosity [Fiorina, I. Clifford, et al., 2015]. The second issue is that the equations are claimed to reduce to traditional RANS when porosity is 1 [Fiorina, I. Clifford, et al., 2015]. Unfortunately, the lack of a dynamic viscosity μ term in the momentum equations means that the resulting RANS equation would describe inviscid flow. Upon examining

subscale structure rather than to adjacent volume elements in the fluid medium. It is not uncommon for μ to be left out, for example Pronghorn [AJ Novak et al., 2018], another multiphysics code, also leaves the laplacian μ term out of its porous media equations. $\dot{Q}_{SS}$ represents the heat transferred from the porous solid in Wm^{-3}, also known as sub-scale structure, to the fluid , $\mathbf{F_g}$ represents the volumetric force vector due to gravity in Nm^{-3} and $\mathbf{F_{SS}}$ represents the volumetric force vector of the subscale structure (porous solid) on the fluid, also known as the drag. Of course , frictional losses contribute to the internal energy of the fluid as well, so that is accounted for within the energy equation.

A minor note on Thermal Conductivity in GeN-Foam for Fluid Energy Equations

k_T in GeN-Foam was stated as turbulent thermal conductivity in $Wm^{-1}k^{-1}$ as of May 2023. This is likely a publication errata because in energy equations, the fluid should be able to conduct heat between volume elements even in laminar flow whether or not there is a subscale structure. Thus, there should be an extra term accounting for conduction and not just turbulent thermal conduction. Pronghorn contains this in its code [AJ Novak et al., 2018] which shows that this term cannot be excluded in general.

However, for porous media, exclusion of fluid thermal conductivity could be a valid assumption depending on the Peclèt number. We discuss this more in detail in Chapter 4.9.

On Interfacial Pressure p_i

The interfacial pressure p_i by the solid on the fluid is assumed to be p [Fiorina, I. Clifford, et al., 2015]. So that:

$$-\nabla\gamma p + p_i\nabla\gamma = -\gamma\nabla p$$

Subscale Drag

Now, $\mathbf{F_{SS}}$ the term describing the subscale momentum sink, is a vector in three principal directions. This term can be estimated in each principal direction using empirical correlations. An empirical correlation for general porous media such as Darcy Forchheimer's equation [I. D. Clifford, 2013; Sobieski and Trykozko, 2014] is used for GeN-Foam's porous media solvers [Fiorina, I. Clifford, et al., 2015]. Let's consider Darcy Forchheimer's equation for the volume element in the x direction [I. D. Clifford, 2013]:

$$-\frac{\partial P}{\partial x} = \frac{\mu}{k}u_{D,x} + \beta\rho u_{D,x}^2 \tag{4.80}$$

the source code in "UEqn_1p.H", I also found this issue to be the present because the viscosity term "nuEff" is taken from the turbulence model rather than the fluid properties. Nevertheless, these equations are still useful for porous media flow.

In equation 4.80, k represents permeability of the porous medium, μ is fluid dynamic viscosity, ρ is fluid density and β is an empirical constant which shows the degree of resistance the pebble bed might give to flows in the inertial regime. Here, we assume k and β are scalars and that flow resistance coefficients are isotropic. In general, flow resistance coefficients in porous media are not isotropic. In fact, GeN-Foam porous media equations were developed for fast reactor calculations [Fiorina, I. Clifford, et al., 2015; Fiorina, Kerkar, et al., 2016; Fiorina, Hursin, and Pautz, 2017] where the porous media models were meant to model fuel pins with coolant. In such a medium, coefficients such as k are anisotropic and depend on flow direction. Hence, k is generally modelled using a vector rather than a scalar. For pebble bed porous media, we assume that k in such a system is isotropic, and is independent of flow direction. We therefore model it using a scalar. Of course, the flow velocity itself is still a vector. And in this case, $u_{D,x}$ is the x component of the Darcy Velocity $\mathbf{u_D}$ where Darcy velocity is defined [I. D. Clifford, 2013; Fiorina, I. Clifford, et al., 2015]:

$$\mathbf{u_D} = \gamma \mathbf{u} \tag{4.81}$$

Fiorina [Fiorina, I. Clifford, et al., 2015] considers these Darcy-Forchheimer equations in terms of Darcy velocity u_D because Darcy velocity does not change when the porosity changes. Darcy velocity is more closely related to superficial velocity since both measure flowrate per unit area [Vafai, 2015]. Now we can see a general form for $F_{SS\ x\ direction}$

$$F_{SS\ x\ direction} = \frac{\mu}{k\gamma} u_{Dx} + \frac{\beta\rho}{\gamma^2} u_{Dx}^2 = \kappa(U_{Dx}) U_{Dx} \tag{4.82}$$

These are extended to a 3D case and be substituted back into the momentum equations.

For packed beds such as pebble beds specifically, Ergun's equation applies [X. Wang, 2018], and it can be written as [Ergun and Orning, 1949; Hassan and Kang, 2012; Perry and Green, 2015]:

$$-\frac{\Delta P}{L} = \frac{150 q \mu}{\Phi_s^2 d^2} \frac{(1-\gamma)^2}{\gamma^3} + \frac{1.75 \rho q^2}{\Phi_s d} \frac{1-\gamma}{\gamma^3} \tag{4.83}$$

Ergun's equation is a more specific form of the Darcy-Forchheimer equation as it applies only to packed beds. Both are equivalent if:

$$\frac{1}{k} = \frac{150}{\Phi_s^2 d^2} \frac{(1-\gamma)^2}{\gamma^3} \tag{4.84}$$

$$\beta = \frac{1.75}{\Phi_s d} \frac{1-\gamma}{\gamma^3} \tag{4.85}$$

Φ_s is sphericity of the pebble, this is to enable Ergun's equation to be used in packed beds with non spherical packings. For spheres, Φ_s is simply 1 [Ozahi, Gundogdu, and Carpinlioglu, 2008]. d is the diameter of the particle or sphere, q is superficial velocity or can be interpreted to mean Darcy velocity in our context.

Subscale Heat Transfer

The subscale heat transfer term $\dot{Q}_{SS}$ which is the sub-scale heat sink or source for non reacting components [Fiorina, I. Clifford, et al., 2015]. It is written as:

$$\dot{Q}_{ss} = A_v h(T_{ss} - T_f) \tag{4.86}$$

Where A_v is the surface area per unit volume between subscale solid structure and the fluid. T_{ss} is the subscale structure temperature of that cell and T_f is the fluid temperature in the same cell. For heat transfer, energy increase due to drag terms is just the dot product of the drag force and local fluid velocity.

Now for heat transfer in FHRs, the Wakao correlation can be used [X. Wang, 2018] to estimate heat transfer coefficeint h. This can be written as [Wakao, Kaguei, and Funazkri, 1979]:

$$\mathrm{Nu}_{d_p} = 2 + 1.1\ \mathrm{Pr}^{1/3}\mathrm{Re}^{0.6} \tag{4.87}$$

Where Nu_{d_p} is the Nusselt Number based on pebble diameter, Re is the Reynold's number, and Pr is the Prandtl Number.

The Nusselt number is:

$$\mathrm{Nu}_{d_p} = \frac{h d_p}{k} \tag{4.88}$$

d_p in equation 4.88 is the pebble diameter. Readers should take note that this is not equal to hydraulic diameter d_h. k is fluid thermal conductivity and h is the heat transfer coefficient. The Reynold's number is based on the porosity and interstitial velocity magnitude, it can be written as [Raluca Olga Scarlat, 2012; Wakao, Kaguei, and Funazkri, 1979]:

$$\mathrm{Re}_{d_p} = \gamma \frac{\rho d_p}{\mu} u_{interstitial} \tag{4.89}$$

The Prandtl number is defined as the ratio of momentum diffusivity ν to thermal diffusivity α. It is also calculated using fluid viscosity μ constant pressure heat capacity c_p and fluid thermal conductivity k:

$$\mathrm{Pr} = \frac{\nu}{\alpha} = \frac{\mu c_p}{k} \tag{4.90}$$

Solid Phase Energy Equation

Within the subscale structure, conduction heat transfer is present, and the energy equation being solved for non heat generating media is [Fiorina, I. Clifford, et al., 2015]:

$$\rho_{ss} c_{p,ss} \frac{\partial T_{ss}}{\partial t} = \nabla \bullet (\gamma \mathbf{k_{ss}} \nabla T) + A_v h(T_{ss} - T_f) \tag{4.91}$$

Where in this case, $\mathbf{k_{ss}}$ is the conductivity tensor which accounts for anisotropic heat conduction within the subscale structure. If it were isotropic, replacing $\mathbf{k_{ss}}$ with a scalar would suffice.

In general, if the structure produces heat from a nuclear reaction, decay reaction or some other reaction eg. chemical, a heat generation term can be added to the equation here:

$$\rho_{ss} c_{p,ss} \frac{\partial T_{ss}}{\partial t} = \nabla \bullet (\gamma \mathbf{k_{ss}} \nabla T) + A_v h (T_{ss} - T_f) + \dot{Q}_{gen} \tag{4.92}$$

To calculate the heat transfer coefficient h, one can refer to an equation such as the Wakao correlation in equation 4.87 to estimate the local heat transfer coefficient.

In codes such as GeN-FOAM however, it is common to simplify these conduction equations in the subscale structure. For fuel pins, they are simplified to 1D heat conduction equations [Fiorina, I. Clifford, et al., 2015] where axial conduction is neglected. The conduction equation for fuel pins for example is written as [Fiorina, I. Clifford, et al., 2015]:

$$\rho_{fuel} c_{p,fuel} \frac{\partial T_{fuel}}{\partial t} = k_{fuel} \frac{\partial^2 T_{fuel}}{\partial r^2} + \frac{k_{fuel}}{r} \frac{\partial T_f}{\partial r} + \dot{Q}_f \tag{4.93}$$

In such a case, conduction between the solid fractions of adjacent volume elements is nonexistent. Conjugate heat transfer is calculated for each volume element as if a representative solitary structure represented the solid phase of the porous media. For pebble bed geometries, GeN-Foam also has capability to subscale structures based on a nodalised thermal resistance model [Robert et al., 2023].

Heterogenous Heat Transfer Media in TRISO Pebble

Now, the TRISO pebble fuels themselves, as we have mentioned, are quite heterogeneous in their geometry and material composition. This makes them a heterogeneous conduction media as well. In this case, we cannot use the always use 1D thermal conductivity equation or no aliased thermal resistance model directly since the heat generating material is heterogeneously dispersed. However, should we apply some homogenisation, these simple models can still be used [M. Liu et al., 2019].

We could fully homogenise the TRISO particles with the graphite matrix. In this model, the peak fuel temperatures tend to be under predicted [M. Liu et al., 2019; Oh, 2006]. In this approach, thermal conductivity of the homogenised pebble is the volume weighted average of the individual components:

$$k_{Thermal\ effective} = \frac{V_p k_p + V_m k_m}{V_m + V_p} \tag{4.94}$$

And the volumetric heat capacity of the pebble can be weighted by volume.

$$\rho_{pebble,T} c_{p,ave,pebble} = \sum_i^N \frac{V_{i,T}}{V_{pebble,T}} \rho_{i,T} c_{p,ave,i} \tag{4.95}$$

Where in equation 4.94 V_p and k_p are the volume and thermal conductivities of the TRISO particles and V_m and k_m are the volume and thermal conductivities of the matrix. In equation 4.95, $\rho_{i,T}$ represents density of pebble material i at temperature T. $c_{p,ave,i}$ is the specific heat capacity of a pebble material i averaged over a suitable range of temperatures. V_i stands for the volume. Where subscripts of the word "pebble" exist, these refer to the quantities averaged over the entire pebble. For example $\rho_{pebble,T}$ represents the volume averaged density of the entire pebble. While inaccuracies would exist, this is by far the simplest approach, and it has value for the first iterations of simulation.

We could correct for this by introducing a single heat generating TRISO particle within the pebble fuel and use that as the reference temperature for fuel [M. Liu et al., 2019; Oh, 2006]. Despite this correction, the peak fuel temperatures may still be underpredicted because the pebble matrix is homogenised. To further correct for homogenisation, Liu suggests to densely pack all the particles to the central region of the core, then perform homogenisation over the densely packed region and then the fuel matrix [M. Liu et al., 2019]. This packing fraction was set to 0.56 to 0.64 for the central closely packed region [M. Liu et al., 2019]. This is not too dissimilar from the traditional reactivity equivalent physical transform (RPT) method used to homogenise the TRISO regions for pebble with dispersed TRISO fuel [Lou, Yao, et al., 2020; Y. Kim and Baek, 2005]. This method served to better predict the peak fuel temperature given steady state conduction.

For this dissertation, we will just stick to the simplest model for expediency even though this is quite important for the purposes of modelling an FHR.This is because the focus of this work is more on deriving the data fitted surrogate model from the reactor than building the simulated reactor. In future, studies could be done to quantify how various homogenisation methods affect fuel temperature feedback and transient behaviour in general.

Radiation Heat Transfer

At operating temperatures of the FHR, radiation heat transfer (RHT) plays a significant role in heat transfer in addition to conduction and convection [I. M. B. Johnson, 2022; Derdeyn et al., 2018]. The underlying equation describing radiation heat transfer is known as the Radiative Transfer Equation (RTE) [M. F. Modest, 2013] and bears some similarity to the neutron transport equation (NTE). Therefore, similar methods such as Monte Carlo methods, SN methods, PN methods [M. F. Modest, 2013] and SPN methods [M. Modest and Lei, 2012] have been used to solve it. This can be very computationally expensive. Therefore, we will need to be judicious about using the limited computational resources available to simulate the most important phenomena.

To check if radiation is significant, we need to quantify the significance of RHT.Johnson did this by normalising blackbody emissive power E_b by convective heat flux q''_{conv} [I. M. B. Johnson, 2022].

$$\frac{E_b}{q''_{conv}} = \frac{\sigma_{Stefan-Boltzmann}(T^4_{hot} - T^4_{cold})}{h(T_{hot} - T_{cold})} \tag{4.96}$$

Where $\sigma_{Stefan-Boltzmann}$ is the Stefan-Boltzmann constant, [M. F. Modest, 2013], T_{hot} and T_{cold} are the temperatures of the hot and cold body respectively. h is the convection heat transfer coefficient.

Blackbody radiation represents the upper bound of how much heat can be absorbed by the cooler body, in this case, FLiBe. This is because while FLiBe is participates in RHT, it would not absorb all the radiation. In the core, Johnson determines this ratio to be about 0.04 given a h to be $4700 W/(m^2 \cdot K)$ as derived from the Wakao Correlation [I. M. B. Johnson, 2022]. Hence, we can see that convection tends to dominate heat transfer so that RHT in the core is not too significant. We might choose to neglect this if computational resources become constrained.

Nevertheless, it is possible that RHT could play an important role in heat exchangers [I. M. B. Johnson, 2022]. A ballpark figure is that for heat exchangers, the ratio of E_b/q''_{conv} may be around 0.40 [I. M. B. Johnson, 2022; Bardet and Per F Peterson, 2008]. This is outside the scope of simulation for this work however, since we are mostly interested in multiphysics in the core.

Interested readers in RHT may note that the interaction of RHT with convection in molten salt is well studied in concentrated solar power (CSP) [Amber and O'Donovan, 2017]. For FHR specific applications, I recommend readers to consider reading about simulated RHT with mixed convection heat transfer for FLiBe has been performed in laminar flow [Abou Dbai, Raluca O Scarlat, and Trujillo, 2020].

4.8 Examples of Codes Used in Reactor Multiphysics

Now that we have discussed some of the phenomena that are possibly important for FHR transients, we now want to examine how this multiphysics coupling has been done in literature.

The coupled multiphysics models we are interested in usually deal with thermal hydraulics and neutronics aspects of the reactor. These have been simulated with Computational Fluid Dynamics (CFD) codes which utilise the multigroup diffusion or simplified spherical harmonics (SPN) models coupled with thermal hydraulics models. Monte Carlo models were used to obtain the fuel temperature and coolant void feedback dependent Multi-Group Cross Sections (MGXS) for FHRs [X. Wang, 2018] in the context of Anticipated Transient without SCRAM (ATWS). These Monte Carlo codes include programs such as Serpent [Leppänen et al., 2014], MCNP or OpenMC [Romano and Forget, 2013; Romano, Horelik, et al., 2015]. The resultant Multi-Group Cross Sections (MGXS) were used in the Computational Fluid Dynamics (CFD) codes such as COMSOL [X. Wang, 2018] or OpenFOAM and GeN-Foam [Fiorina, I. Clifford, et al., 2015]. Deterministic neutronics equations such as diffusion and SP3 were run coupled with the porous media thermal hydraulics equations [X. Wang, 2018]. Of the multiphysics coupling methods, coupling porous media equations with deterministic multigroup neutronics equations is considered "lower fidelity" compared to coupling the CFD code to the Monte Carlo Code directly. Such endeavours were performed using Open Source

Software such as OpenMC and Nek5000 within the Multiphysics Object Oriented Simulation Environment (MOOSE) Framework [April Novak et al., 2018]. Closed Source Monte Carlo codes such as Serpent have been coupled to Open Source codes such as OpenFOAM as well for prompt criticality simulations [Aufiero, Fiorina, et al., 2015] and FHR simulations [Aufiero and Fratoni, 2016].

Usually the "higher fidelity" models use the Monte Carlo Code such as Serpent to calculate reactor power at each timestep for use in the CFD code such as OpenFOAM, which then updates the spatial temperature distribution. The temperature distributions are then fed back to the Monte Carlo Code so that reactor power is re-evaluated [Sorrell and Hawari, 2019]. This is in contrast to the "lower fidelity" coupling method where the Monte Carlo Code calculates the MGXS at different temperatures once [X. Wang, 2018] and then MGXS are interpolated at each timestep using a user-defined temperature correlation [X. Wang, 2018]. Usually these higher fidelity models are used for benchmarking purposes as was done for the Generic FHR (gFHR) [Satvat et al., 2021] or for the Transient Reactor Test Facility (TREAT)[Sorrell and Hawari, 2019]. High The benchmarks developed using higher fidelity models can become references for lower fidelity neutronics and thermal hydraulics coupled models or even surrogate models such as Kairos Power (KP) Advanced Gas REactor Evaluation (AGREE), also known as KP-AGREE [Satvat et al., 2021; Blandford et al., 2020][11].

For demonstration purposes, we may want to use simpler coupled multiphysics methods as the basis of our simulations. Once again, we are only interested in having a high fidelity model so that we can develop a data fit surrogate model. Hence, for this dissertation, I would favour using diffusion neutronics as opposed to Monte Carlo methods or other deterministic methods because of its simplicity.

In future, other high fidelity models can be used as a source of data for the data fit surrogate model. We may in future use a code to simulate multiphysics with SP3 neutronics or SN neutronics as a baseline for the higher fidelity model. We can fit the improved multiphysics model into a state space model, or even an AI model.

4.9 Justification for Software and Multi Physics Modelling Choices

For this demonstration case, we intend to simulate the use of CIET's digital twin (specifically one for the heater) in developing controllers for simulated neutronics experiments. Therefore, we are only interested in demonstrating how a data fitted surrogate model can be derived from a higher fidelity simulation and then have this surrogate model developed into a controller using the Type I digital twin. To achieve this goal, we are only going to do one iterative "dry run" of controller development to demonstrate how CIET's digital twin may

[11] Actually, KP-AGREE is possibly one example of a hierarchical reduced order model where simplifying assumptions on the underlying reactor physics ensure that the model is faster to solve

expedite development of a simulated neutronics facility. This, of course, will motivate both the multiphysics model and the choice of software used to construct the multiphysics model.

The multiphysics model will be rather crude in terms of fidelity and in terms of generously using simplifying assumptions to expedite development. However, the multiphysics model is developed such that, for researchers familiar with coupled multiphysics simulations, it should be easy to improve upon the existing model and apply a similar method to derive the data fitted surrogate model. Furthermore, the surrogate model used here is quite simple as well as we intend to develop a linear transfer function in this chapter to characterise neutronics behaviour. Specifically, we wish to study transients such as ULOHS.In such a scenario, the inlet temperature of the core may increase for protracted periods and the PSP does not trip. This leads to a decrease in reactor power. The increase in inlet temperature and decrease in reactor power would then produce changes in outlet temperature. We aim to quantify these changes by deriving an inlet temperature to outlet temperature transfer function at constant coolant flowrate.

To construct the multiphysics model, I favour Open Source options as the source code is readily available and modifiable. This makes it easier for researchers who don't have access to software with export control or a paywall to execute what is done in this work. Of course, proprietary software is also used provided it is not excessively expensive for students and academic researchers.

OpenMC as the Monte Carlo Code of Choice

Deterministic neutronics and thermal hydraulics solvers such as GeN-Foam require macroscopic flux averaged energy cross sections as inputs for each cell zone. This is then generated by a Monte Carlo Code. For this work, we can consider three out of the myriad of Monte Carlo codes which are used for Monte Carlo simulations. These are MCNP, Serpent and OpenMC.For this work, OpenMC was used because it is not as heavily export controlled as MCNP or Serpent even though it does not have the same delta tracking capacities. The license for MCNP and Serpent was only obtained two to three months after request, and by that time, I had began to use and learn OpenMC[12].To expedite research, OpenMC was used as the Monte Carlo code of choice.

GeN-Foam as Multiphysics Code of Choice

GeN-Foam vs Commercial Solvers

Comsol has been used to incorporate SP3 equations to model nuetronics for the FHR [X. Wang, 2018]. However, in this case, Comsol requires that one manually code in both equations for thermal hydraulics and the underlying Sp3 equations or diffusion equations for neutronics.

[12]This is anecdotal evidence based on my own experiences as an international student from Singapore applying for export controlled Monte Carlo code

Furthermore, commercial codes such as Comsol are proprietary and closed source, meaning that the source code of the software is not provided and cannot be edited. For a finer level of control, Free and Open Source (FOSS) software can be used as an alternative. For example, if the user wishes to speed up calculations using GPU computing, it is possible to do so with OpenFOAM [Krasnopolsky and Medvedev, 2016], but not with COMSOL as of 2023 [Comsol, 2023].Hence, if the user wanted to edit the source code of COMSOL to support GPU computing, it is not possible. But if the user wants to extend the functionality of OpenFOAM, it is very much a normal to delve into the source code and write a solver suited for one's use case for free. Furthermore, in terms of access, FOSS software is much more accessible since it is licensed under copyleft licenses such As GNU General Public License 3 (GNU GPL 3). This means that simulations can be done for relatively lower cost.

GeN-Foam uses OpenFOAM libraries and extends it for neutronics with a porous media model. It has built in libraries for Multi-Group diffusion [Fiorina, Kerkar, et al., 2016] and SP3 [Fiorina, Hursin, and Pautz, 2017]. It is also licensed under GNU GPL 3. Therefore, GeN-Foam seems to be a good choice among the FOSS software.

Caveats for Neutronics

The only caveats for GeN-Foam are that it is built for fast reactors. Most of these poisons have huge absorption cross sections when incident neutrons are mostly thermalised. For fast neutrons, the absorption cross sections are much smaller, and hence, these poisons become less significant in the fast spectrum reactor as compared to the thermal spectrum reactor. Therefore, the capability for simulating reactor poison buildup is absent. The capability for simulating decay heat is also similarly absent. The MGXS inputs also do not currently account for moderator temperature feedback by default.

When it comes to adjusting cross sections for temperature, GeN-Foam takes cross sections at two given temperatures and interpolates between them or extrapolates beyond them in a log-linear fashion. (see readNuclearData.H in GeN-Foam Source code). Interpolating interpolating temperature dependent MGXS in a log-linear fashion has been used before in FHR simulation [X. Wang, 2018] and seems to be a reasonable thing to do. However, we must ensure that cross sections for this interpolation are taken at appropriate temperature ranges.

For the reference temperatures, we consider that when transients occur, FHR temperatures can range from about 500 °C (773K) to 1000 °C (1273K). In normal operations, temperatures may range from 600 °C (873K) to 700 °C (973K). However, OpenMC's default library only contains cross sections at temperatures of 250.0K, 293.6K, 600.0K, 900.0K, 1200.0K and 2500.0K. To make things worse, OpenMC's default library does not contain thermal scattering data even at 900.0K. We could either use NJOY [Macfarlane et al., 2017] to generate thermal scattering data, or choose 600K and 1200K as the suitable interpolation points.

For this work, I found NJOY too tedious to learn and use correctly. Hence I have opted for to interpolate temperature dependent MGXS between 600K and 1200K.

Caveats for Thermal Hydraulics

In terms of thermal hydrualics, GeN-Foam, as of May 2023, lacks the capability for simulating a pure solid graphite reflector block because the porous media equations do not reduce to the conduction equation when porosity is equal to 0.

In fact, heat transfer for passive (non heat generating) structures between solid portions of each volume element is absent as the conduction term is missing. We can take a look at some excerpts of the source code and we will see that the conduction equation (pasEqn) only includes the time derivative term and the convection heat transfer term. For example:

"classes/thermalHydraulics/src/phaseModels/structureModels/structure.C":

```
if (Tpas_.writeOpt() == IOobject::AUTO_WRITE)
{
    //- Correct inert subStructure. What follows is the equivalent of doing
    //  the following:
    /*
        fvScalarMatrix pasEqn
        (
                fvm::ddt(alphaRhoCppas_, Tpas_)
            ==
                iApas_*HT
            -   fvm::Sp(iApas_*H, Tpas_)
        );
        pasEqn.solve();
    */
    //  Except, it is faster like this rather than to solve an equation
    //  over the entire mesh, as the passive subStructure might not exist
    //  everywhere

    scalar dt(mesh_.time().deltaT().value());
    const volScalarField& Tpas0(Tpas_.oldTime());
    forAll(cells_, i)
    {
        label celli(cells_[i]);
        const scalar& iA(iApas_[celli]);
        if (iA == 0) continue;  //- Avoid solving where the passive
                                //  structure does not exist
        scalar alphaRhoCpByDt(alphaRhoCppas_[celli]/dt);
        Tpas_[celli] =
            (
                iA*HT[celli]
            +   alphaRhoCpByDt*Tpas0[celli]
```

```
            )/
            (alphaRhoCpByDt + iA*H[celli]);
    }
    Tpas_.correctBoundaryConditions();
}
```

Here, we see in the comments the fvScalarMatrix pasEqn is being solved for the solid phase in the passive structure. "alpha" here represents the phase fraction, and in the case of a solid structure, the solid phase fraction. Users reading the source code may see "alpha" being used for liquid phase fraction as well, so it may be confusing.

"iApas_" represents the interfacial area of the passive structure per unit volume.

"fvm::ddt(alphaRhoCppas_, Tpas_)" represents $(1-\gamma)\rho c_p \frac{\partial T_{passive}}{\partial t}$. Where $1-\gamma$ is the solid phase fraction, ρ is solid phase density, c_p is solid phase heat capacity and $T_{passive}$ is the solid phase (passive structure) temperature.

```
iApas_*HT - fvm::Sp(iApas_*H, Tpas_)
```

Would represent $A_v h(T_{fluid} - T_{passive})$. Where A_v is interfacial area per unit volume, h is heat transfer coefficient, HT is simply hT_{fluid} and "fvm::Sp(iApas_*H, Tpas_)" represents $A_v h T_{passive}$.

This goes to show that modelling capabilities for heat conduction in the solid phase are quite limited. Therefore, in the GeN-Foam simulation, any graphite reflector structures do not add thermal inertia to the reactor, and instead are assumed to be at constant temperature.

The second issue, as explained earlier, is that the fluid thermal conductivity term is missing from the porous media equations. A reason why conduction may be neglected is because fluid advection is the dominant form of heat transport especially within the direction of flow. We discuss this matter after verifying that the conductivity term is missing in the source code of GeN-Foam. Let us look into the files "fluid.C" and "fluid.H", the thermal conductivity used in the energy equations under "EEqn_1p.H":

```
volScalarField alphaEff
(
    fluid_.thermo().alphaEff(fluid_.turbulence().alphat())()
);
```

This code excerpt shows that only the turbulent thermal diffusivity is used to set the fluid effective thermal diffusivity. One can verify that the α_t given in this equation comes from the turbulence model by looking where α_t is defined. For the porous media $k-\varepsilon$ model in GeN-Foam "porousKEpsilon.C", α_t is calculated in the "correctNut()" function which does:

```
void porousKEpsilon<BasicTurbulenceModel>::correctNut()
{
```

```
    this->nut_ = Cmu_*sqr(k_)/epsilon_;
    this->nut_.correctBoundaryConditions();

    //- Correct alphat before nut is stabilized
    this->Prt_ = dimensioned<scalar>::lookupOrDefault
    (
        "Prt",
        this->coeffDict(),
        1.0
    );
    this->alphat_ = this->rho_*this->nut_/this->Prt_;
    this->alphat_.correctBoundaryConditions();

    if (nutStabilization_)
    {
        this->nut_ +=
            pos(structure_)*FSPair_.fluidRef().magU()*DhStructPtr_()/
            laminarReStructPtr_();
        this->nut_.correctBoundaryConditions();
    }

    fv::options::New(this->mesh_).correct(this->nut_);

    // BasicTurbulenceModel::correctNut();  //- Eh, I don't want other things
                                            //  messing with the stabilized nut
}
```

This shows that α_t is correlated to μ_t, rather than being taken from the thermophysical properties of the fluid. However, it would seem that the subscale structures heat transfer terms correctly use the μ in their correlations. This would mean that the solid-fluid heat transfer equations in GeN-Foam are likely correct, but the fluid-fluid heat transfer equations are missing a conduction term due to a simplifying assumption that advection dominates conduction heat transfer. The dimensionless number to quantify the relative importance of advection to diffusion is known as the Peclèt number (Pe) [Huysmans and Dassargues, 2005]. One possible formulation is:

$$\mathrm{Pe}_{d_p} = \frac{d_p u_D}{\alpha} = \mathrm{Re}_{d_p}\,\mathrm{Pr} \tag{4.97}$$

A high Pe would mean that advection is much more important than conduction for heat transport. We can use typical Re and Pr found in FHRs to estimate Pe in the pebble bed. Typically in FHRs, one can expect Pr of about 16 and Re of 100 to 2250 [Dave, Sun, and L. Hu, 2020]. Therefore, a plausible lower bound for Pe is 1600, which can be found in

natural circulation. For forced circulation, Pe is even higher because of higher Re. Given that Pe is at least 1000 in either case for the pebble bed, we can justify that neglecting conduction heat transfer between adjacent fluid elements would not significantly impact heat transfer calculation. One can nondimensionalise the energy transport equation to see that the conduction term is scaled by $\frac{1}{\text{Pe}}$ [April Novak, 2020; Bejan, 2013].

For its intended purpose of modelling heat transport in porous media, GeN-Foam's porous media model works well as thermal conductivity is very small for flow in porous media as compared to dispersion from mixing and advection. GeN-Foam also allows the user to define a heat transfer coefficient via a Nusselt Number correlation between solid and fluid. These would include the Wakao Correlation. The heat transfer coefficient h is then calculated using fluid thermal conductivity and a system length scale to dimensionalise the Nusselt Number. Hence, the porous media thermal transport does perform its job well, but would have problems being adapted for use in modelling heat transfer regimes where conduction is dominant.

One example is when $\text{Re} \ll 1$. This happens perhaps when flow stops completely during salt freeze. In this case, conduction can no longer be neglected and the porous media equations as shown in GeN-Foam can no longer apply. For our case study of the reactor, we minimally have rates of circulation typical of natural convection. Hence, neglecting conduction along the direction of flow is a safe assumption. The other case is where radial conduction is considered. If we consider flow to be negligible in the radial direction, then radial conduction becomes important. This would result in higher peak temperatures in the core as high temperature fluid in the center of the core would not diffuse into the peripheral fluid regions. The end result is that we have a higher peak temperature in the core and the fluid. This could perhaps be a conservative assumption from the standpoint of safety. Hence, one need not worry too much about this issue, and any sensitivity study for this can be relegated to future work. Therefore, GeN-Foam should not severely impact heat transfer results unless $\text{Re} \ll 1$.

Nevertheless, conduction is modelled where it is most important: the boundary layers. Again, these are are accounted for via user defined empirical correlations describing subscale structure heat transfer between solid and fluid. So while conduction is neglected, GeN-Foam porous media equations may still serve as a good approximation for most transients. For example, GeN-Foam results are found to be in good agreement with TRACE for unprotected loss of flow (ULOF) and unprotected transient OverPower (UTOP) in the original GeN-Foam paper [Fiorina, I. Clifford, et al., 2015]. Furthermore, with a loss of conduction between volume elements in the fluid, the overall heat transfer rate from the core would be lower than if there was conduction. This provides us with a more conservative estimate of heat removal. Hence, one could argue that despite this simplification, GeN-Foam is still reasonably useful for transient analysis. Of course, it would be ideal to add the thermal diffusivity or non turbulent thermal conductivity term back into the energy equation in future work to account for radial conduction from the fluid into and from the graphite blocks.

These thermal hydraulics models can be modified by the user and the source code can be recompiled and used. This would of course take more time and testing to ensure that the

modified code works correctly. Due to time constraints, these are out of scope for this work and relegated to future work.

Conclusion for why I chose GeN-Foam

Despite some of the caveats in using GeN-Foam, the flexibility and accessibility it offers far outweighs the cost of having to use software with a paywall. I have opted to accept and use GeN-Foam with its existing caveats without modifying the source code for expediency in this work. This is because it would not severely impede the process of deriving a transfer function from the frequency response test of the simulated GeN-Foam reactor. We could always improve the fidelity of the multiphysics model in future work and use a similar process to derive a data driven surrogate model for controller design.

Geometry and Meshing Tools

It is important to consider geometry and meshing because the same geometry used for MGXS generation in OpenMC needs to be imported into mesh construction in GeN-Foam.

For geometry construction in OpenMC, there was a choice to use Computer Aided Design (CAD) software or Constructive Solid Geometry (CSG) for geometry construction. I used CSG because I was intending to generate a simple geometry for the reactor. CSG works quickly for such a situation. This geometry is easy to replicate in CAD software.

For GeN-Foam mesh construction, it is convenient to use FOSS tools with Graphical User Interfaces (GUI) for GeN-Foam and OpenFOAM.This is more convenient than OpenFOAM's command line tools such as blockMesh and snappyHexMesh for mesh construction. Therefore, FreeCAD [Riegel, Mayer, and Havre, 2016] was used for geometry construction Salome's TetGen platform was chosen for generating a tetrahedral mesh [Lee, Park, and S. W. Kim, 2014] from that geometry.

Conclusion for Software Choices and Multiphysics Models

I have chosen GeN-Foam and OpenMC for multiphysics simulation because they are released under FOSS licenses. This allows academic researchers who use them to have maximum flexibility and accessibility. OpenMC's CSG is used for geometry construction while FreeCAD and Salome Platform's TetGen are used to reconstruct this same geometry in GeN-Foam. With the software choices decided upon, we can explore some principles for reactor simulation and construction.

4.10 Design Principles for Arbitrary Reactor Construction

Overview

The reactor design principles are laid out in this section, this pertains mainly to the geometry and scaling for the reactor. As we have already discussed reactor physics to a fair degree, we will not repeat the discussion here.

We call the reactor here the arbitrary reactor. It is meant to be a scaled up version of CIET's Heater with some neutronics feedback added. As there are several mechanisms of neutronics feedback, development of a fully functional simulated neutronics feedback controller which can work for all transients may be challenging. To begin construction of such a controller, we want to break the problem down into segments and apply the rapid prototyping approach [Blandford et al., 2020] so that development can be expedited. Rapid prototyping usually involves iterative development, where we construct imperfect models to learn more about the design process so that the next model is better. What this means is that we would build a vastly simplified model and controller first, and then improve upon this model over several iterations until the desired outcome is achieved. This arbitrary reactor is merely the first of many reactor models that we can use to construct simulated neutronics feedback controllers. Hence, the scope of transient simulation is meant to be limited.

In the first iteration, we wish to obtain simulated fuel temperature feedback behaviour in unprotected loss of heat sink (ULOHS) transients. For simplicity, we will not change the coolant flowrate as these may introduce additional complications in system modelling and surrogate model development. In effect, we are modelling ULOHS without a PSP trip. A ULOHS transient without PSP trip serves as a good starting point due to the relative simplicity for modelling using transfer functions. Without PSP trips, the only change the reactor would likely experience is the inlet temperature. With a PSP trip, the reactor would be subject to a lower coolant flowrate and change in inlet temperature. The latter model requires a multiple input multiple output (MIMO) model while the former model only requires a single input single output (SISO) model.

As mentioned earlier, this ULOHS without PSP trip scenario is perhaps more dangerous for the primary loop compared to a PSP trip because in the former case, the reactor is rejecting heat at a higher rate into the metal structures. Therefore, it is a very important important transient to study. Thus, we have additional motivation to test the developed arbitrary reactor for this particular transient. We also want to develop the arbitrary reactor in such a way that we can scale it for use in CIET.These goals drive the design of the arbitrary reactor.

Of course, the ultimate goal is to demonstrate the utility of the digital twin of CIET (or at least just its heater for now) in expediting something as complicated as simulated neutronics feedback controller development. To do this, we will only be going through one such iteration.

Scaling to CIET

The arbitrary reactor must be designed such that it is scalable to CIET.CIET was originally meant to be a scaled down IET of the Pebble Bed Advanced High Temperature Reactor (PB-AHTR) [I. M. B. Johnson, 2022], one could use a customised scaling methodology to scale up CIET into a prototypical FHR design as was envisioned for the Mark II PB-FHR reactor [I. M. B. Johnson, 2022].

One of the limitations of CIET is that its heater, at its time of operation, is only 10 kWth. Hence, we must ensure that the arbitrary reactor's power output scales to this power output or less. This is because the reactor power output might spike to a level higher than its designed power output during unprotected transients. If we wish to enable ourselves to simulate such power spikes, then we cannot use the full heater power during steady state. For now, I will just assume the steady state power of CIET is 5 kWth.

We then need the arbitrary reactor to somehow scale down to this representative power level of CIET's heater. There are several ways this can be done. One of them is to use CIET's original scaling methodology [Zweibaum, J E Bickel, et al., 2015; Zweibaum, Guo, et al., 2016; Bardet and Per F Peterson, 2008]. This is the quickest and simplest way of scaling since a precedent has been established. However, Johnson [I. M. B. Johnson, 2022] has outlined a more comprehensive methodology for scaling which includes taking into account radiative heat transfer scaling distortions. For this dissertation, the simpler methodology is used for the first iteration. Future iterations of reactors based of different scaling methodology can be used in place of this first iteration of the arbitrary reactor and then used to construct reactivity feedback controllers for CIET.

To scale the power of the model (CIET) P_m to the power output of the prototypical reactor P_p, we can consider the following relation:

$$\frac{P_m}{P_p} = \frac{\rho_m c_{pm} \Delta T_m U_m L_m^2}{\rho_p c_{pp} \Delta T_p U_p L_p^2} \tag{4.98}$$

Where in equation 4.98, the subscript m represents the model (CIET) and the subscript p represents the prototypical FHR.c_p is constant pressure specific heat capacity, ΔT is characteristic temperature difference, U is fluid velocity, ρ is mass density and L is lengthscale. We can see that to determine power scaling of the arbitrary reactor to the CIET, we need to determine the length scales, time scales, and temperature scales as well.

Temperature Scaling

For CIET's original scaling methodology, characteristic temperatures of Dowtherm A are scaled to FLiBe such that Prandtl Numbers are matched. For Prandtl Number of 11.7 to 18.6 for FLiBe, this corresponds to a temperature of 600 °C to 704 °C[Zweibaum, J E Bickel, et al., 2015]. Readers should note that Therminol VP-1 and Dowtherm A are similar enough heat transfer fluids as both of them are eutectic mixtures of Diphenyl Oxide and Biphenyl. For the purposes of scaling at temperatures of operation, they are virtually identical [Ong, 2023].

Likewise for Dowtherm A, a Prandtl number of 12.8 to 16.9 corresponds to a temperature of 80 °C to 111 °C.

The temperature difference is chosen by matching the Grashof number between the prototypical reactor and CIET.The Grashof number based on a lengthscale L is [Bejan, 2013]:

$$\mathrm{Gr}_L = \frac{g\beta\Delta T L^3}{\nu^2} \tag{4.99}$$

In equation 4.99, ΔT is the characteristic temperature difference between the hot and cold regions, β is coefficient of thermal expansion, ν is kinematic viscosity and g is gravitational acceleration. For temperature difference, $\beta\Delta T$ in equation 4.99 is matched between both FLiBe and Dowtherm A [Bardet and Per F Peterson, 2008]. The ratio β is fixed when using Dowtherm A to simulate FLiBe, and the ΔT of the FHR is specified, leaving ΔT of the Dowtherm A to be determined. The remaining terms in the Grashof number are matched when matching Re.

The overall effect is that a 1K temperature difference in FLiBe represents a 0.3 K temperature difference in Dowtherm A Zweibaum, J E Bickel, et al., 2015.

Time Scaling

Time distortions were done to preserve the Strouhal number (Sr[13]) [Bardet and Per F Peterson, 2008]. The Strouhal number can be written as:

$$\mathrm{Sr} = \frac{fL}{U} \tag{4.100}$$

In equation 4.100, L represents a characteristic length, U represents characteristic fluid velocity and f represents a frequency [Ahlborn, Seto, and Noack, 2002]. The Strouhal number has more to do with vortex shedding [Ahlborn, Seto, and Noack, 2002] than heat transfer, but it was still used in CIET's scaling methodology [Bardet and Per F Peterson, 2008]. Hence, the input and output frequencies are also scaled according to this. 1 second of time in a FLiBe system represents 0.67s of time in a Dowtherm A system. Therefore, 1 Hz in a FLiBe system represents 1.5 Hz in a Dowtherm A system.

Length, Velocity and Flowrate Scaling

CIET was scaled to a length scale of about 0.45 times the height of a prototypical FHR [Bardet and Per F Peterson, 2008]. To obtain similitude, the Reynold's and Froude number of CIET and the prototypical FHR must match given this length scaling of approximately 0.45. Froude number is used to compare viscous forces to gravity forces [Bardet and Per F Peterson, 2008] and therefore scale the head gain and loss in the loop respectively. To achieve the same Froude and Reynold's number given a length scaling factor of 0.45, 1 m^3/s of FLiBe volumetric flowrate should correspond to 0.122 m^3/s of flowrate in Dowtherm A.

[13]Strouhal number is sometimes abbreviated as St [Ahlborn, Seto, and Noack, 2002], but I choose Sr to avoid confusion with the Stanton Number which is also commonly abbreviated as St[Bejan, 2013]

Power Scaling

Now that the temperature scales, length scales and time scales have been determined, power scaling can then be derived from temperature and volumetric flowrate scaling provided that the ratio of the volumetric heat capacities remain relatively constant at the relevant temperatures.

For CIET, Dowtherm temperatures range from 80 ° C to 110 ° C. With a mass flowrate of 0.18 kg/s, this corresponds to a power of 9.415 kW, which is almost the maximum power allowed by the CIET heater of 10 kW without upgrades[14] [Zweibaum, Guo, et al., 2016]. This corresponds to a FLiBe temperature difference of about 100 K, and a FLiBe flowrate of 2.88 kg/s , and a power of 696 kW. This shows a power scaling of about 1.33 % which is a reasonable estimate of the 1.6% power scaling described in Zweibaum's paper [Zweibaum, J E Bickel, et al., 2015; Zweibaum, Guo, et al., 2016] given round off errors and scaling differences in temperature.

Of course, considering that we want to scale to 5 kW in CIET's heater, we might have the reactor power for the arbitrary FHR to be around 300 kW. The FLiBe flowrate through the core would be closer to 1.44 kg/s.

Simplified Geometry

For this arbitrary FHR, the reactor geometry should be extremely simple to construct. Therefore, we will choose a simple cylindrical geometry to represent the reactor.

Lack of Control Elements

For such ULOHS transients, control rods are not important, at least for this first iteration. In such a transient, the primary system is not be able to remove heat from the system, and the system is unable to SCRAM in time to reduce heat output. As mentioned earlier, ULOHS transients mean that the PSP does not trip. This both simplifies the simulation and also allows us to explore the severe transient where the PSP does not trip. In this transient, the reactor should experience increased temperatures as heat removal from the primary loop through the CTAH or some other heat exchanger system is compromised. As a result, core temperatures rise, and this should shut down the reactor without intervention from control rods. The concern is whether this rise in temperature would result in fuel damage or structural damage, especially if the PSP does not trip. For such ULOHS transients, control rods are not needed.

For this work, we are only constructing a reactor for testing ULOHS effects (without PSP trip) in the primary loop. Additional capabilities are to be added in future iterations. Hence, we have not added control rods in this iteration of arbitrary reactor.

[14]It may be possible to upgrade CIET's heater to 20 kW, but this is out of scope of discussion for this work.

Double Heterogenity

We will want the reactor to behave as closely as possible to an FHR with pebble fuel filled with TRISO particles and cooled with molten FLiBe salt. Therefore, we want to make some effort to mimic double heterogenity effects and packing fractions in the pebble fuel so that overall reactivity of the pebble bed is preserved. Hence, a transform using principles of Ring RPT or RPT will be used to speed up Monte Carlo calculations. To further speed up construction and calculation, we will therefore use packing fractions similar to randomly packed pebble beds by arranging pebbles in a hexagonal lattice for OpenMC calculations. This packing fraction is around 0.605, which is reasonable for randomly packed pebble beds [Hao, Yang, and Y. Cheng, 2022; Abedi and Vosoughi, 2011].

Simplified MGXS Zones

In terms of MGXS generation, we build the reactor such it is split into four regions or cell zones:

- fuel cell
- reflector cell
- entrance FLiBe cell
- exit FLiBe cell

While it would be ideal to have the pebble bed split into many different cell zones to account for neutron importance, packing heterogenity and heterogenity in burnup, including such details would not contribute to the main aim of the test. Here, we merely wish to demonstrate that high fidelity models can be used to build surrogate models for use in simulated neutronics facilities. Ultimately, this is meant to demonstrate a use case for the Type I Digital Twin of CIET.Therefore, we only wish to have a single MGXS cell zone represent the entire pebble bed so as to expedite development. This will not mean that each zone shares the same homogeneous cross section, but rather that each zone shares the same cross section temperature dependence.

Simplified Energy and Angular Discretisation

For purposes of this dissertation, we want the multiphysics simulation to be able to run quickly without use of supercomputers. Furthermore, I did not want to introduce the extra complication of coupling two separate simulation programs with the use of a wrapper at this stage. Hence, we are not going to couple Monte Carlo code to thermal hydraulics directly. We will instead use the Monte Carlo Code OpenMC [Romano, Horelik, et al., 2015] to generate multigroup cross sections (MGXS) for each region in the reactor so that GeN-Foam

can use those cross sections for a two group diffusion simulation. This greatly simplifies energy and angular dependence of neutrons within the reactor.

For FHRs, we can use the Maxwell Boltzmann curve to decide what energy is an appropriate boundary for fast and thermal groups. Based on the gFHR flux spectrum [Kile et al., 2022] and previous TMSR-SF simulations [X. Wang, 2018], 3 or 4 eV would also be a decent choice to separate fast and thermal groups. 4 eV is of significance also because that is where the low energy resonance region for U-238 begins. This can be seen in TMSR-SF spectra [X. Wang, 2018].

Conclusion of Arbitrary Reactor (FHR) Principles of Construction

We have discussed a very simple approach to quickly constructing a multiphysics model of an Arbitrary FHR which I will just call the arbitrary reactor. The reader should note that this reactor is meant to be a representative model of higher fidelity reactor simulations which would account for energy and spatial dependence of flux and cross sections. Therefore, this arbitrary reactor is rather uses many simplifying assumptions and is rather crude in construction design as well as fidelity.

For now, we shall use a cylindrical arbitrary reactor with simplified cell zones and we will not include control rods for the time being. We also take measures such as RPT and heat transfer medium homogenisation to simplify the doubly heterogeneous TRISO pebble bed geometry. Furthermore, we also neglect phenomena such as reactor poison accumulation, burnup, structural expansion and decay heat so as to expedite model construction. Model fidelity can be improved upon in future iterations of the arbitrary FHR.

4.11 Transfer Function Construction and Scaling

Overview

For this work, we are using transfer functions as a data fit surrogate with which to construct simulated neutronics feedback controllers. The frequency response tests will involve using PRBS to perturb the arbitrary FHR to obtain a transfer function via the Bode Plots as described in the literature review.

We are interested in ensuring CIET's heater has essentially the same transfer function of inlet to outlet temperature as the scaled arbitrary FHR.In essence:

$$G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s) = G_{CIET\ heater\ inletT\ to\ outletT}(s) \tag{4.101}$$

The heater itself would have a heater power to outlet temperature transfer function at a set flowrate [De Wet and Per F Peterson, 2020]. To ensure the behavior of the heater reflects the scaled arbitrary FHR, a feedforward controller will need to be developed to control heater power based on heater inlet temperature.

$$G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s) = \\ G_{CIET\ heater\ heaterPower\ to\ outletT}(s) G_{CIET\ heater\ inletT\ to\ heaterPower}(s) \tag{4.102}$$

Where $G_{CIET\ heater\ heaterPower\ to\ outletT}(s)$ is the ideal transfer function of the feedforward controller.

Controller Design Principles

DeWet has provided some transfer functions for heater power to outlet temperature [De Wet and Per F Peterson, 2020]:

$$G(s) = e^{-4s} \frac{3.217 * 10^{-5} s^3 + 6.675 * 10^{-7} s^2 + 1.139 * 10^{-8} s + 2.423 * 10^{-11}}{s^5 + 0.2251 s^4 + 0.01688 s^3 + 0.0003548 s^2 + 3.057 * 10^{-6} s + 1.632 * 10^{-9}}$$

Should we find the scaled arbitrary FHR transfer function for inlet to outlet temperature $G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s)$, then we should be able to find the feedforward controller transfer function:

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s) = \frac{G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s)}{G_{CIET\ heater\ heaterPower\ to\ outletT}(s)} \tag{4.103}$$

It is noteable that we will be inverting a time delay of 4 seconds. This is because the heater power to outlet temperature transfer function in equation 3.65 contains a 4 second time delay that we wish to invert. Inversion can be approximated with use of a 2^{nd} order Padé approximation [Vajta et al., 2000]:

$$e^{-cs} \approx \frac{12 - 6cs + (cs)^2}{12 + 6cs + (cs)^2} \tag{4.104}$$

Vajita recommends a modified Padé approximation which avoids an output signal at $t = 0s$ and it can be written as [Vajta et al., 2000]:

$$e^{-cs} \approx \frac{6 - 2cs}{6 + 4cs + (cs)^2} \tag{4.105}$$

Or in the first order approximation [Vajta et al., 2000]:

$$e^{-cs} \approx \frac{1}{1 + cs} \tag{4.106}$$

The first approximation just approximates the time delay as a low pass filter, which makes it rather convenient.

The second thing we have to note is the numerator of the heater power to outlet temperature transfer function, which now becomes the denominator when inverted in the feedforward controller. We would ideally want to design a stable controller to emulate reactivity feedback, unless of course that feedback causes the system to be actually be unstable. In either case, we wish to find the stability of the feedforward heater controller with inverted transfer function. To determine stability, we need to find the roots of:

$$3.217 * 10^{-5}s^3 + 6.675 * 10^{-7}s^2 + 1.139 * 10^{-8}s + 2.423 * 10^{-11} = 0$$

Using MATLAB's root function, the roots are $-0.0092 + 0.0150i$, $-0.0092 - 0.0150i$, and -0.0024. The first two roots pertain to a sinusoidal function while the second root indicates $(s + 0.0024)$ is the other factor. These go to show that when the polynomial appears in the denominator, the poles are such that the transfer function is stable. If the roots contain unstable poles, a different procedure would have to be used. However, I will not cover the stabilisation procedure in this work.

Lastly, for transfer functions, the order of the denominator should be greater or equal to the numerator for it to be physically realisable and proper. We could add some simple low pass filters to make it physically realisable if need be. For controllers however, we note that the differential component of a Proportional Integral and Derivative (PID) controller is in itself improper, so the transfer function of the controller does not always have to be proper.

Arbitrary FHR Transfer Function Construction and Verification

We will use the frequency response derived Bode Plots to obtain a transfer function. We then verify those transfer functions with a simple step response test of both the arbitrary FHR and the derived transfer function.

To check if nonlinearities exist, we can alter the amplitudes of frequency response and step tests to check the limits of applicability for the resultant transfer function.

Arbitrary FHR Transfer Function Scaling

These transfer functions will need to be scaled for use in CIET so that they can be used. The simplest way to achieve this is to scale the time domain data directly before deriving a transfer function. Based on the time scales, we can scale time intervals Δt by 2/3 approximately, and based on the temperature scales, we can scale the using a linear relationship between Therminol VP-1 temperature and FLiBe temperature assuming that $600°C$ in FLiBe translates to 80 $°C$ in Therminol VP-1, $700°C$ in FLiBe translates to 110 $°C$ in Therminol VP-1. This results in:

$$T_{therminol\ VP-1} = mT_{FLiBe} + C$$

Where:

$$m = \frac{110 - 80}{700 - 600} = 0.3(\text{dimensionless})$$

$$c = 80 - 0.3 * 600 = -100°C$$

The final equation to translate all the time domain data is:

$$T_{therminol\ VP-1}(°C) = 0.3T_{FLiBe}(°C) - 100 \tag{4.107}$$

We can then repeat the transfer function construction process using the scaled test data.

4.12 Conclusion of Literature Review and Principles for Simulated Neutronics Facility Based on Data Based Surrogate Model

The construction of simulated neutronics facilities based on data fit surrogate models is a potential improvement over PRKE based simulated neutronics facilties. Nevertheless, the process of constructing a simulated neutronics facility in this manner proves to be a non trivial process. Therefore, it serves as a suitable case for which to demonstrate the utility of a digital twin for CIET in expediting new developments.

Here, we have discussed principles we can use to derive surrogate models for use in constructing a simulated neutronics controller. We have discussed the various types of surrogate modelling including hierarchical, data fit and projection based surrogate modelling. We also discuss the methods and principles for constructing the higher fidelity model. In this case, it is an FHR multiphysics model based on porous media simulated in GeN-Foam using multigroup diffusion model. This model becomes the basis from which we deriving the data for the data fitted surrogate model. Lastly, we describe how the information from the data fitted surrogate model can be scaled converted into information for controller design specifically for CIET.We also discussed the simplifications made in order to expeditiously create a first iteration of the mulitphysics arbitrary reactor simulation.

Now that we have sufficiently covered these general aspects of arbitrary FHR construction as well as scaling and conversion of its reactor feedback into a transfer function, we can now proceed to the specifics of the arbitrary FHR, its test results and transfer functions.

Chapter 5

Multiphysics Model and Surrogate Model Construction and Results

5.1 Introduction

In this chapter, we utilise the principles outlined in the preceding chapter to construct a scaled transfer function based on a high fidelity multiphysics model for simulated neutronics feedback in Integral Effects Test (IET) facilities such as the Compact Integral Effects Test (CIET).

Firstly, we discuss methods used to construct and simulate the arbitrary pebble bed fluoride salt cooled high temperature reactor (FHR). Secondly we systematically perturb the high fidelity model using periodic binary signals to obtain system frequency response in the form of a Bode plot. We then analyse the Bode plot using Matlab's tfest function to obtain a transfer function which is representative of data fit surrogate models one can use in future. Thirdly, we discuss methods used in constructing a scaled transfer function for use in CIET's heater, or more precisely, the digital twin of CIET's heater.

5.2 Reactor Construction Methods

Let us begin by constructing the arbitrary reactor. Figure 5.1 provides an overview of how we go from the conceptual design to OpenMC model and then lastly to the GeN-Foam model.

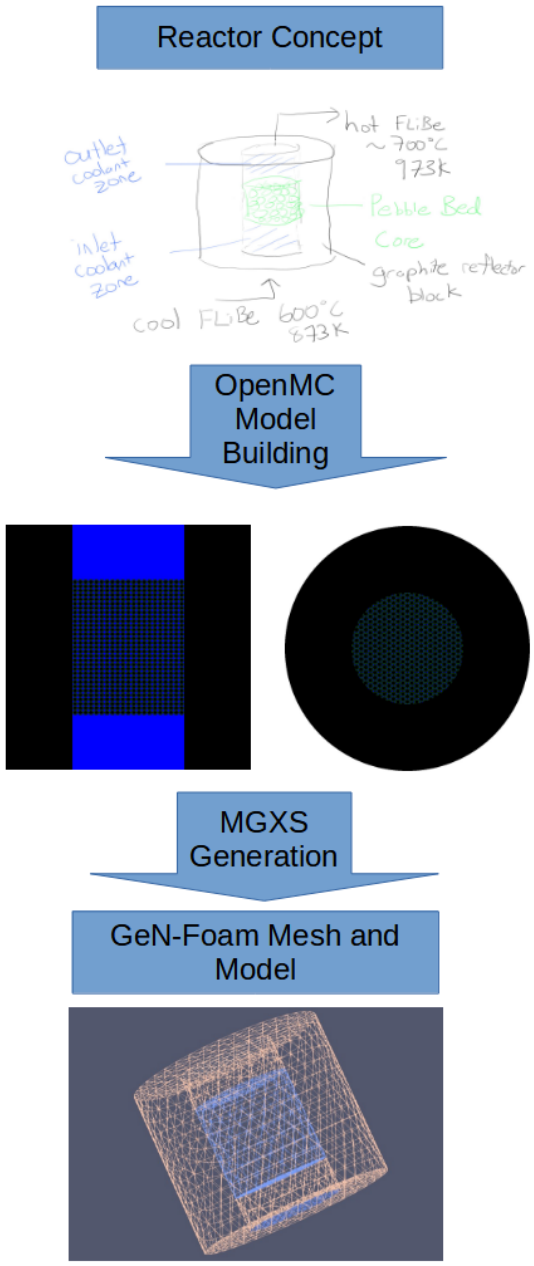


Figure 5.1: Overview of the Arbirary Reactor Design Process

Conceptual Design

As shown in Figure 5.1, we start with the concept. The arbitrary reactor or arbitrary FHR is designed using simple cylindrical geometry. Its pebbles are packed to a packing fraction similar to pebble beds of 0.61 far from the wall. Its coolant enters at $600°C$ and exits at $700°C$. The nominal power rating is 312.5 kW based on the constraints of CIET's heater. The pebbles themselves filled with TRISO.

However, simulating TRISO particles is computationally expensive to use directly in a full core. Hence, we use a modified Ring Reactivity Physical Transform (RPT) in order to speed up the simulation. Also, to speed up development of this first iteration, there are no control rods designed into the reactor. They can be added in future iterations. Furthermore, the random packing of the pebbles is simulated using a simple hexagonal lattice and the pebbles are clipped at the boundaries. This is, once more, to save time on development of a first iteration of the simulated neutronics controller. With these in mind, let us move on to performing Ring RPT of the TRISO fuel pebble.

Reactivity Equivalent Physical Transform of TRISO Pebble

Preliminaries

To start constructing the pebble bed core, we first consider the TRISO pebble. Normally, FHR pebbles are annular in design to allow for buoyancy [Cisneros et al., 2014]. In this work, the TRISO Fuel Pebble is of a non-annular design for simplicity. The reason is because I wanted to achieve criticality with a smaller core so that Monte Carlo simulation would be faster. To do so, I would then need to use fewer pebbles. The only way to do so is to fill each pebble with more TRISO particles.

TRISO Particle Properties in OpenMC

This TRISO pebble is 4cm in diameter and is separated into a region where a graphite matrix is filled with TRISO particles, and then a separate graphite shell on the outside. The TRISO particles themselves have a Uranium Oxycarbide (UCO) fuel kernel ($U_{0.333}C_{0.1667}O_{0.5}$). They are coated in shells of buffer carbon, Silicon Carbide (SiC) and Pyrolytic Carbon. Each shell in the TRISO particle has a fixed thickness shown in Table 5.1. We also show the outer radius of each layer in the TRISO particle in Table 5.1:

Table 5.1: TRISO Pebble Dimensions

Material Name	Outer Radius (cm)	Shell Thickness (cm)
Uranium Oxycarbide (fuel)	0.0215	-
buffer	0.0315	0.01
Pyrolytic Carbon (PyC1)	0.035	0.0035
Silicon Carbide (SiC)	0.0385	0.0035
Pyrolytic Carbon (PyC2)	0.0425	0.004

We should note that the UCO is the innermost sphere in the TRISO particle, and therefore, it does not have a shell thickness entry. We now move on to material composition.

FLiBe Material Inputs

For FLiBe, the Lithium-7 is enriched to about 99.995%. The inputs of for FLiBe material in OpenMC are summarised in table 5.2:

Table 5.2: FLiBe OpenMC Inputs

	FLiBe (600K)	FLiBe (1200K)
Density g/cm^3	2.1216	1.827
^{6}Li Fraction	0.000014286	0.000014286
^{7}Li Fraction	0.2857	0.2857
^{9}Be Fraction	0.14286	0.14286
^{19}F Fraction	0.57143	0.57143
Total Fraction	1	1

Two temperatures are provided in table 5.2 because we need to account for coolant temperature feedback. Thus, two different simulations of the pebble bed will be run at two bounding temperatures 600K and 1200 K. This is because graphite thermal scattering data was available at these two temperatures from the OpenMC nuclear data libraries but not at other temperatures. Unfortunately, 600K is below the freezing point of FLiBe, thus the density reflected in Table 5.2 is not a physically realistic density, but just an extrapolation

of the density correlation for FLiBe at this temperature. We simply ignore FLiBe freezing in this simulation and assume density varies linearly between 600K and 1200K. This should not be a problem for this high fidelity model because the FLiBe temperatures do not go below freezing point anyway.

Fuel Material Inputs

The uranium in UCO is enriched to 19.9 atom%. It is then homogenised with the carbon and SiC layers surrounding the UCO prior to Ring RPT form "Homogenised TRISO Fuel". The homogenisation was done to preserve total volume of the TRISO particles. A summary of material densities is shown in table 5.3:

Table 5.3: TRISO and Ring RPT Pebble Material Densities

Material Name	Density g/cm^3
Uranium Oxycarbide (fuel)	10.5
Buffer	1.0
Pyrolytic Carbon (PyC1)	1.9
Silicon Carbide (SiC)	3.2
Pyrolytic Carbon (PyC2)	1.87
Graphite Matrix and Shell	1.1995
Homogenised Triso Fuel	2.996095

The resultant atomic fraction of important nuclides and atoms within each material is summarised in table 5.4:

Table 5.4: TRISO and Ring RPT Pebble Material Atom Fractions

Material Name	^{235}U	^{238}U	^{16}O	Carbon (Natural Abundance)
Uranium Oxycarbide (fuel)	0.066333	0.267	0.5	0.16667
Buffer	0	0	0	1
Pyrolytic Carbon (PyC1)	0	0	0	1
Silicon Carbide (SiC)	0	0	0	0.5
Pyrolytic Carbon (PyC2)	0	0	0	1
Graphite Matrix and Shell	0	0	0	1
Homogenised Triso Fuel	0.0076737	0.030888	0.057843	0.79166

For SiC and the Homogenised TRISO Fuel in table 5.4, the remaining material is composed on Silicon with 92.2% ^{28}Si, 4.68% ^{29}Si, and 3.08% ^{30}Si.

The graphite matrix with TRISO will be filled with TRISO particles using packing fraction of 0.30. This means that, by volume, 30% of the graphite matrix region will be filled with TRISO particles. Figure 5.2 shows an OpenMC plot of such a pebble surrounded with FLiBe. Here, blue colour represents FLiBe. And for those who cannot see colour, the outermost ring is where the FLiBe is present.

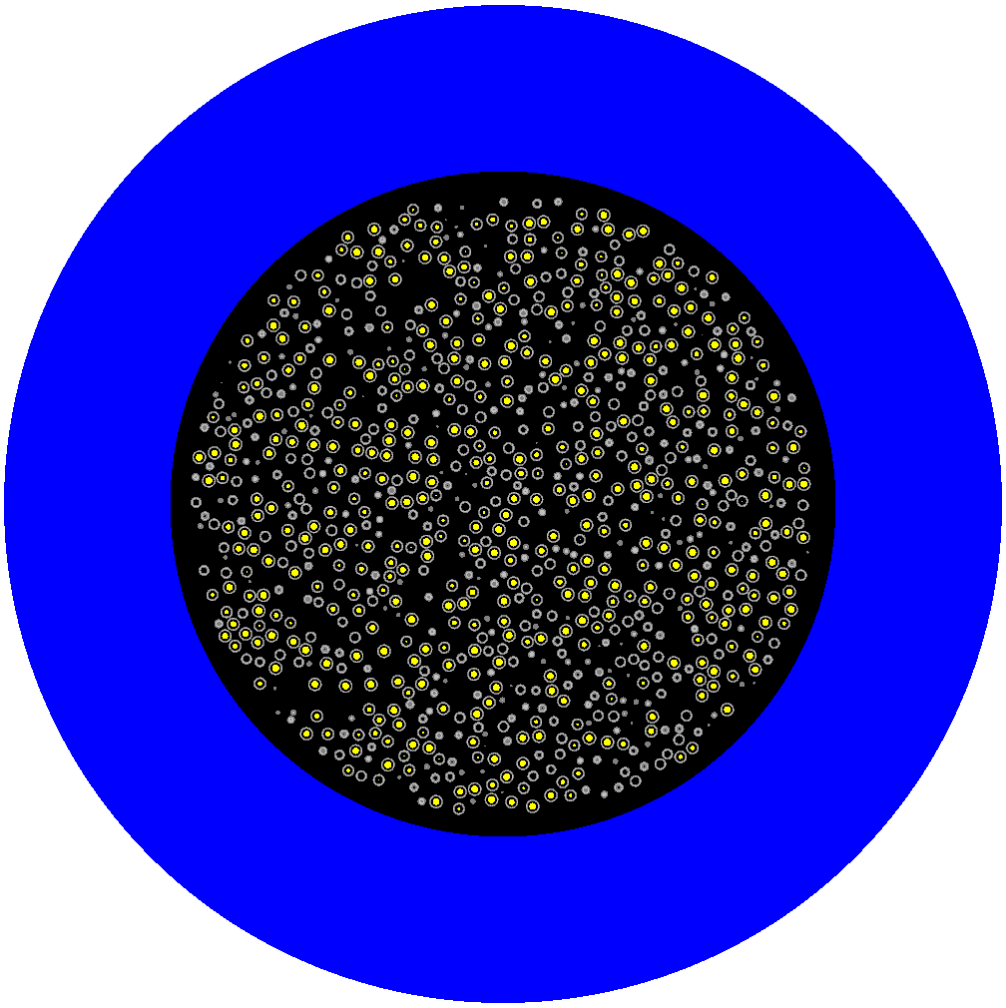


Figure 5.2: TRISO Pebble used for Ring RPT

Homogenised Ring RPT Temperature Considerations

The temperature at which the RPT was done was 600 K because that was a convenient temperature which had thermal scattering data in OpenMC's nuclear data libraries. Freezing phenomena and its associated density changes are ignored for this case because we do not intend to simulate FLiBe temperatures below freezing point anyway. Ideally, we would perform the RPT at least 873 K which is a typical coolant inlet temperature of the FHR.However, thermal scattering data in this temperature range was missing and needed to be generated by NJOY.This was too time consuming to do for this "dry run". Such things can be relegated to future work.

Homogenised Ring RPT compared to Traditional Ring RPT

To homogenise the pebble and simplify its geometry, a variant of the Ring RPT method [Lou, Yao, et al., 2020; Lou, Chai, et al., 2020] was used. In traditional Ring RPT, the TRISO fuel particles are combined into a ring. These fuel particles are not necessarily homogenised with the shell layers of the TRISO particle. Initial trials show that performing Ring RPT method without homogenisation of the TRISO particle with its shell layers failed for spherical fuel because the maximum radius of the Ring RPT method and the minimum radius of the Ring RPT method produce a k_∞ range that did not bound the k_∞ of the TRISO Pebble[1]. For traditional Ring RPT, one lowers the spatial self shielding effects by increasing the radius of the ring. We aim to increase the radius of the ring or spherical shell until the k_∞ matches that of the explicitly modelled TRISO pebble. However, the maximum radius of the ring is limited by the radius of the pebble itself. I found that even when maximising the radius of the ring, the k_∞ of the Ring RPT pebble exceeds that of the k_∞ of the TRISO pebble. This is likely because there is too much self shielding in the Ring RPT pebble as compared to the TRISO pebble. This agrees with results in literature which have also concluded that this same limitation for Ring RPT existed for spherical fuel [Lou, Chai, et al., 2020]. As I had already written code for the Ring RPT method, I wished to find a method which could reuse as much of that code as possible for convenience and expediency. I had also initially thought that I was not doing the Ring RPT method correctly and that perhaps the entire TRISO Particle with its shell needed to be homogenised. What I then decided to do is to homogenise the TRISO Fuel Kernel with its layers of Silicon Carbide, Pyrolytic Carbon and Graphite into a single Ring Layer or Shell Layer. Then I would perform Ring RPT from here. This approach turned out to be successful. I call this the "homogenised Ring RPT" approach. On hindsight, homogenised Ring RPT further lowers the spatial self shielding effects by partially dispersing the fuel with the carbon and silicon moderator materials, and then it allows the user to further tweak the self shielding experienced by the fuel by adjusting the radius. Further studies into this homogenisation method may be worthwhile, but are out of scope for this dissertation. Figure 5.3 compares these two approaches:

[1] In this trial though, I homogenised the shell layers as that would likely produce negligible change in k_∞ [Fratoni, 2008].

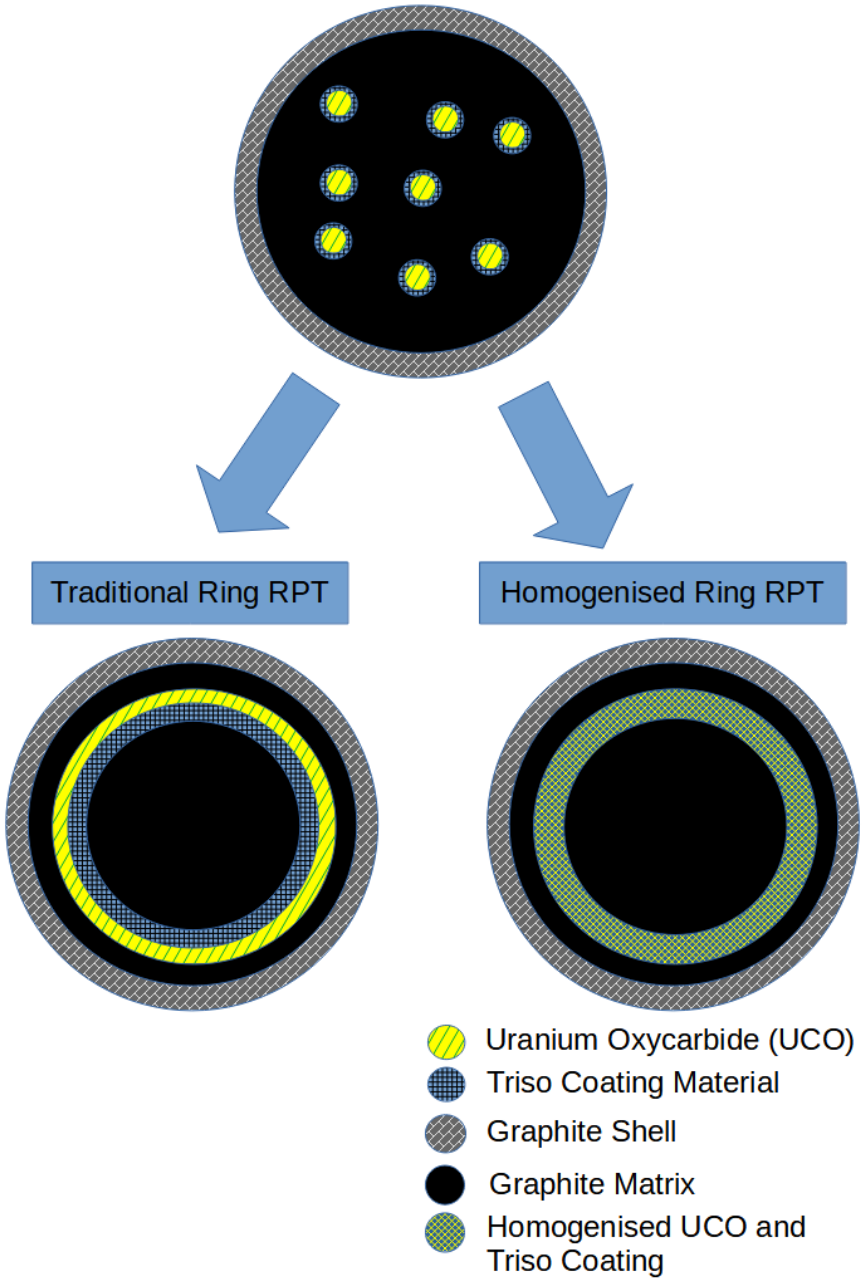


Figure 5.3: Homogenised Ring RPT vs Traditional Ring RPT

In comparison to traditional Ring RPT, homogenised Ring RPT partially homogenises the dispersed TRISO Fuel with graphite and SiC similar to traditional RPT [Lou, Yao, et al., 2020], but instead of concentrating material in the centre of the fuel pebble, it is smeared

out into a Ring or Shell Layer similar to Ring RPT [Lou, Chai, et al., 2020]. As mentioned before, the user then varies the radius of the ring or spherical shell to adjust the degree to which the fuel experiences spatial self shielding and therefore k_∞. This partially homogenised Ring RPT approach was able to reproduce k_∞ for pebble fuels and significantly sped up the simulation process.

A second ad hoc modification I made was to include an extra layer of FLiBe into the Ring RPT homogenisation process. This was initially done because FLiBe also moderates the neutron spectrum in addition to the carbon and silicon within the fuel. This process is best illustrated by Figure 5.4:

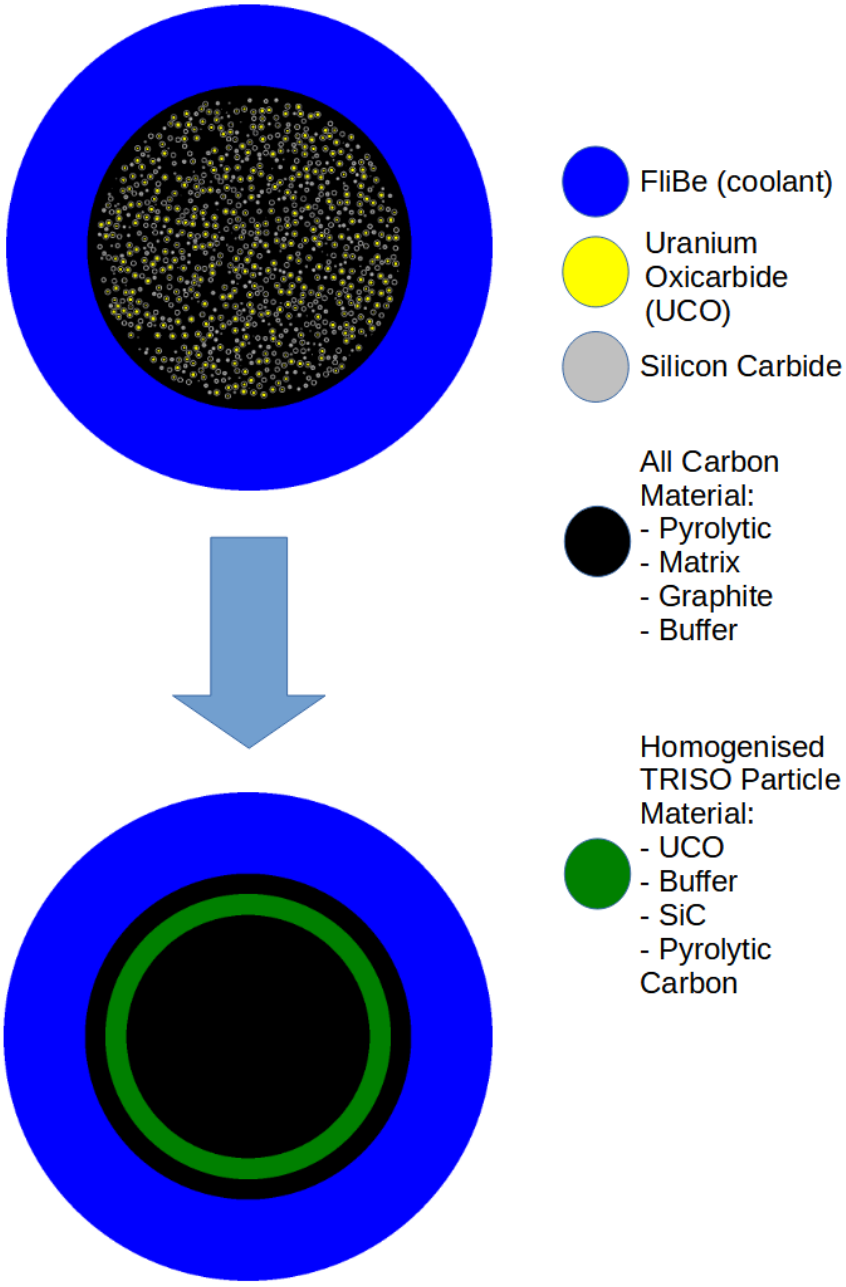


Figure 5.4: Process of Homogenised Ring RPT in OpenMC

Including FLiBe in the homogenised Ring RPT process was a mistake as Ring RPT in literature would just include homogenising the pebble such that k_∞ of the Ring RPT would match k_∞ of the TRISO Pebble [Lou, Chai, et al., 2020; Lou, Yao, et al., 2020]. This

did not take into consideration the nature or moderation effects of the coolant or reflector surrounding the pebble. However, in the interest of time, I decided that this mistake was not critical to correct for this first iteration, and just let it pass in the interest of increasing development speed so that I did not have to rewrite my python code again. With this in mind, I just decided to carry on performing Ring RPT using the current setup.

Eigenvalue Calculations

To perform my variation of Ring RPT, I first performed an eigenvalue calculation for the reference TRISO Pebble with FLiBe. This yielded the following results in OpenMC.

```
=======================>     TIMING STATISTICS     <=======================

Total time for initialization     = 2.0080e+01 seconds
Reading cross sections          = 6.9374e-01 seconds
Total time in simulation          = 2.0993e+04 seconds
Time in transport only          = 2.0990e+04 seconds
Time in inactive batches        = 1.0957e+03 seconds
Time in active batches          = 1.9897e+04 seconds
Time synchronizing fission bank = 2.5006e-01 seconds
Sampling source sites         = 2.1309e-01 seconds
SEND/RECV source sites        = 3.4434e-02 seconds
Time accumulating tallies       = 1.7900e+00 seconds
Time writing statepoints        = 8.0442e-01 seconds
Total time for finalization       = 4.7082e+00 seconds
Total time elapsed                = 2.1023e+04 seconds
Calculation Rate (inactive)       = 91.2685 particles/second
Calculation Rate (active)         = 95.4925 particles/second

============================>     RESULTS     <============================

k-effective (Collision)     = 1.36904 +/- 0.00087
k-effective (Track-length)  = 1.36826 +/- 0.00138
k-effective (Absorption)    = 1.36903 +/- 0.00067
Combined k-effective        = 1.36902 +/- 0.00066
Leakage Fraction            = 0.00000 +/- 0.00000
```

I used a search for k_{eff} method in OpenMC's application program interface (API). Some data for this is plotted in figure 5.5.

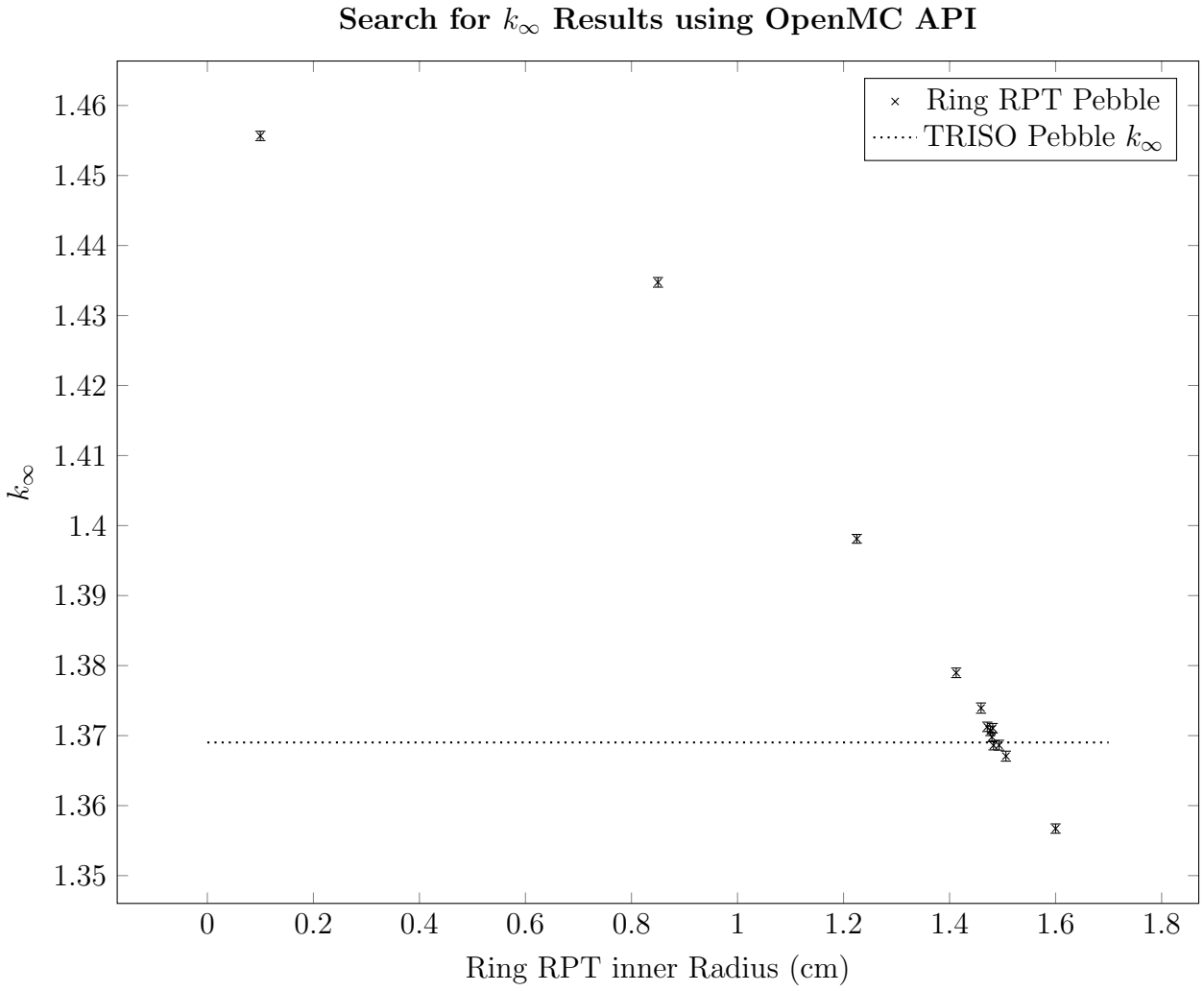


Figure 5.5: Search for k_{eff} data using OpenMC API for Ring RPT, error bars are 1σ

This was an eigenvalue calculation in OpenMC with 200 batches, 10 rejected batches, and 10,000 particles per batch[2]. The initial fission source was a distributed fission source that spreads source sites somewhat evenly among fissionable material within the pebble. This would ensure that the source sites of the Monte Carlo calculation would reach a more even distribution with fewer batches as compared to a point source in the beginning. Also, using such a source would help the Monte Carlo simulation converge for dispersed fuel such as TRISO fuel pebbles. Indeed, this initial fission source distribution among the fuel, the k_∞ of an OpenMC simulation for 10 and 50 batches were not statistically distinguishable. For 10 batches rejected in the TRISO pebble, 200 batches total $k_\infty = 1.36902 \pm 0.00066$

[2]For better source convergence, I should have rejected 50 batches. I did not redo most of these calculations including my neutron spectrum plots for expediency and convenience. For full core MGXS generation, however, I rejected 50 batches.

whereas for 50 batches rejected, 200 batches total, $k_\infty = 1.36828 \pm 0.00080$. For the TRISO pebble with reflective boundary conditions, the difference in k_∞ between using 50 inactive batches and 10 inactive batches was statistically insignificant. This is likely because we used a volumetrically dispersed source initially, and also that the reactor (or pebble) geometry was small compared to a full core which necessitated about 50 inactive batches. I used 10 discarded batches because they made the iterative process search for k_{eff} process faster.

I then iteratively obtained a homogenised Ring RPT pebble of specified inner radius at 1.4933593750000003 cm. This is the merely value given by the search for k_{eff} API and is not necessarily indicative of the level of precision needed to perform Ring RPT transforms.

This resulting RPT pebble was also subject to an eigenvalue calculation which yielded the following results, again with 10 batches rejected:

```
=======================>     TIMING STATISTICS     <=======================

Total time for initialization     = 7.8420e-01 seconds
Reading cross sections          = 7.4211e-01 seconds
Total time in simulation          = 1.1323e+02 seconds
Time in transport only          = 1.1292e+02 seconds
Time in inactive batches        = 4.4105e+00 seconds
Time in active batches          = 1.0881e+02 seconds
Time synchronizing fission bank = 1.9981e-01 seconds
Sampling source sites         = 1.6241e-01 seconds
SEND/RECV source sites        = 3.4997e-02 seconds
Time accumulating tallies       = 4.5674e-02 seconds
Time writing statepoints        = 9.2186e-03 seconds
Total time for finalization       = 1.7580e-03 seconds
Total time elapsed                = 1.1407e+02 seconds
Calculation Rate (inactive)       = 22673 particles/second
Calculation Rate (active)         = 17460.9 particles/second

============================>     RESULTS     <============================

k-effective (Collision)     = 1.36841 +/- 0.00117
k-effective (Track-length)  = 1.36763 +/- 0.00134
k-effective (Absorption)    = 1.36870 +/- 0.00070
Combined k-effective        = 1.36863 +/- 0.00070
Leakage Fraction            = 0.00000 +/- 0.00000
```

For 10 batches rejected in the homogenised Ring RPT pebble, 200 batches total $k_\infty = 1.36863 \pm 0.00070$ whereas for 50 batches rejected, 200 batches total, $k_\infty = 1.36880 \pm 0.00080$. Hence, even at 10 inactive batches, we may consider that the fission source has converged.

The residual of the Ring RPT k_∞ value and the TRISO Pebble k_∞ is 0.00039 or about 39 pcm. This is within 1σ or 1 standard deviation of either the k_∞ calculation for the TRISO fuel pebble. Thus, the k_∞ are not statistically different from one another. We can then conclude that Ring RPT was successfully performed.

Brief Investigation of Self Shielding Deviation between TRISO Pebble and its Homogenised Ring RPT Pebble Counterpart

While the Ring RPT was performed so k_∞ was matched, I found it important to check that the self shielding phenomena was at least partially accounted for using the Ring RPT method. To ensure that Ring RPT could somewhat reflect the self shielding phenomena in a TRISO Fuel Pebble, I decided to check if the Ring RPT fuel pebble could produce a similar degree of thermalisation as the TRISO Fuel Pebble. This is because the resonance escape probability p directly impacts the degree of thermalisation within a neutron spectrum. For a critical reactor, if there is a low resonance escape probability, then it is likely that a larger proportion of fission neutrons originate from fast fission. In other words, the fast fission factor ε has to be higher. In this case, we should observe a higher proportion of fast neutrons in the neutron spectrum as compared to thermal neutrons. Therefore, I wanted to inspect the ratio of fast neutrons to thermal neutrons, using the neutron spectrum. A second way to quickly check if self shielding was accounted for was to inspect how the terms in the four factor formula differ between the RPT pebble and TRISO pebble. We shall discuss both of these here.

The reader should note that this analysis is rather brief. This is because investigating the deviation in self shielding effects is not the main aim of this dissertation. Hence, any further detailed analysis on self shielding deviation due to the RPT transform is out of scope. This means that comparing quantities such as the sensitivity of p to temperature will not be done for the RPT pebble and TRISO pebble. Analysis of quantities like these can be relegated to future work.

Neutron Spectrum Let us begin by comparing the neutron spectrum of the TRISO fuel pebble and homogenised ring RPT pebble over the graphite region, FLiBe region and fuel regions. The graphite region plots are presented in Figure 5.6:

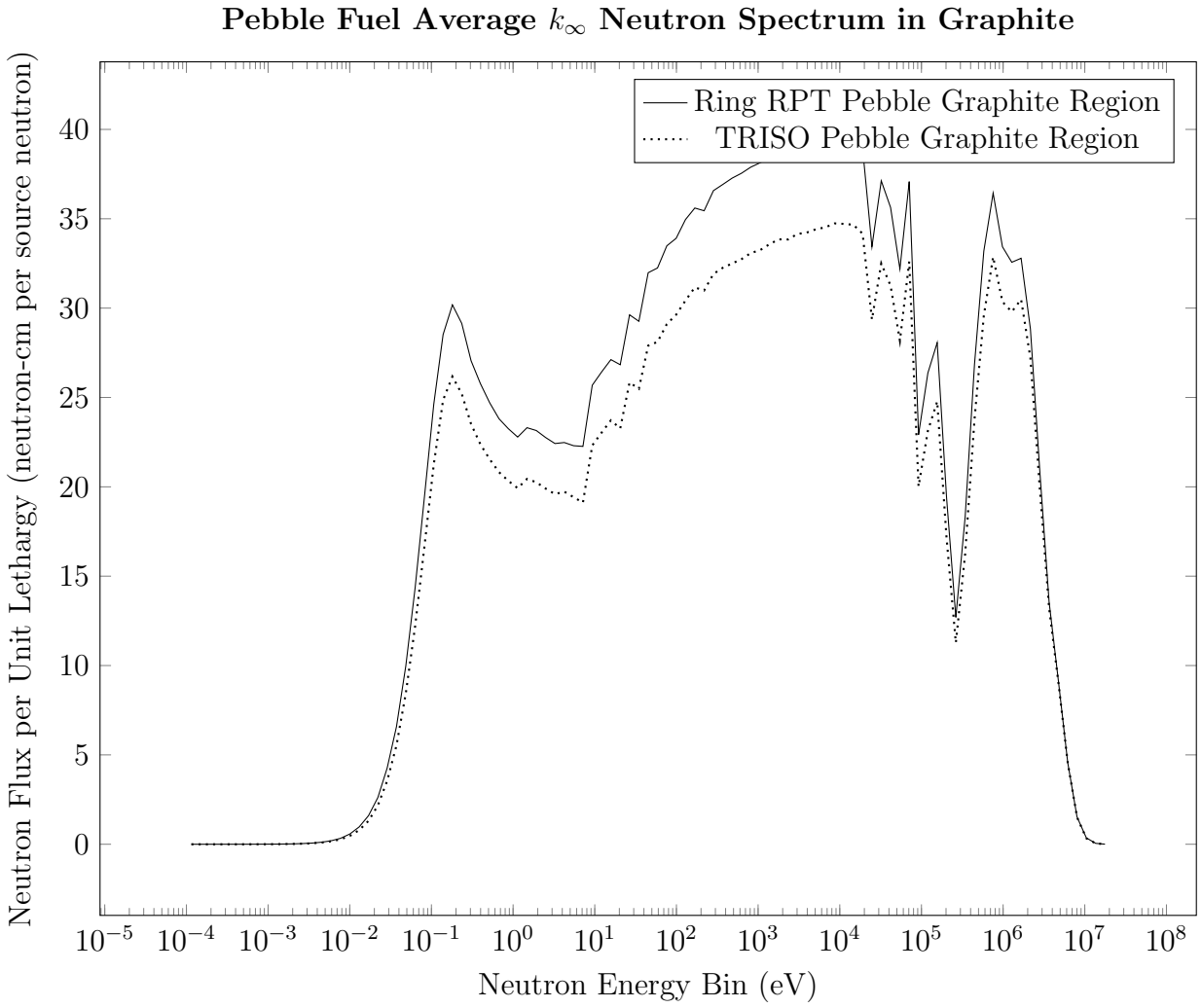


Figure 5.6: Neutron Spectrum for k_∞ for Pebble Fuel in Graphite

The neutron spectrum is plotted as Neutron Flux per Unit Lethargy in units neutron-cm per source neutron as is indicated in OpenMC documentation in the abscissa. The ordinate is the Neutron Energy in eV. Of course, when it comes to tallies in OpenMC, we call the energy range where we collect the "flux scores" as "energy bins" [Romano and Forget, 2013; Romano, Horelik, et al., 2015]. Therefore, the ordinate is called Neutron Energy Bin in eV. Do note that only 100 bins are used because we only want to visually inspect the qualitative shape of the graph and ensure that it is similar so that the degree of thermalisation of the neutron spectrum is similar.

We also see the same trend in the FLiBe region shown in Figure 5.7:

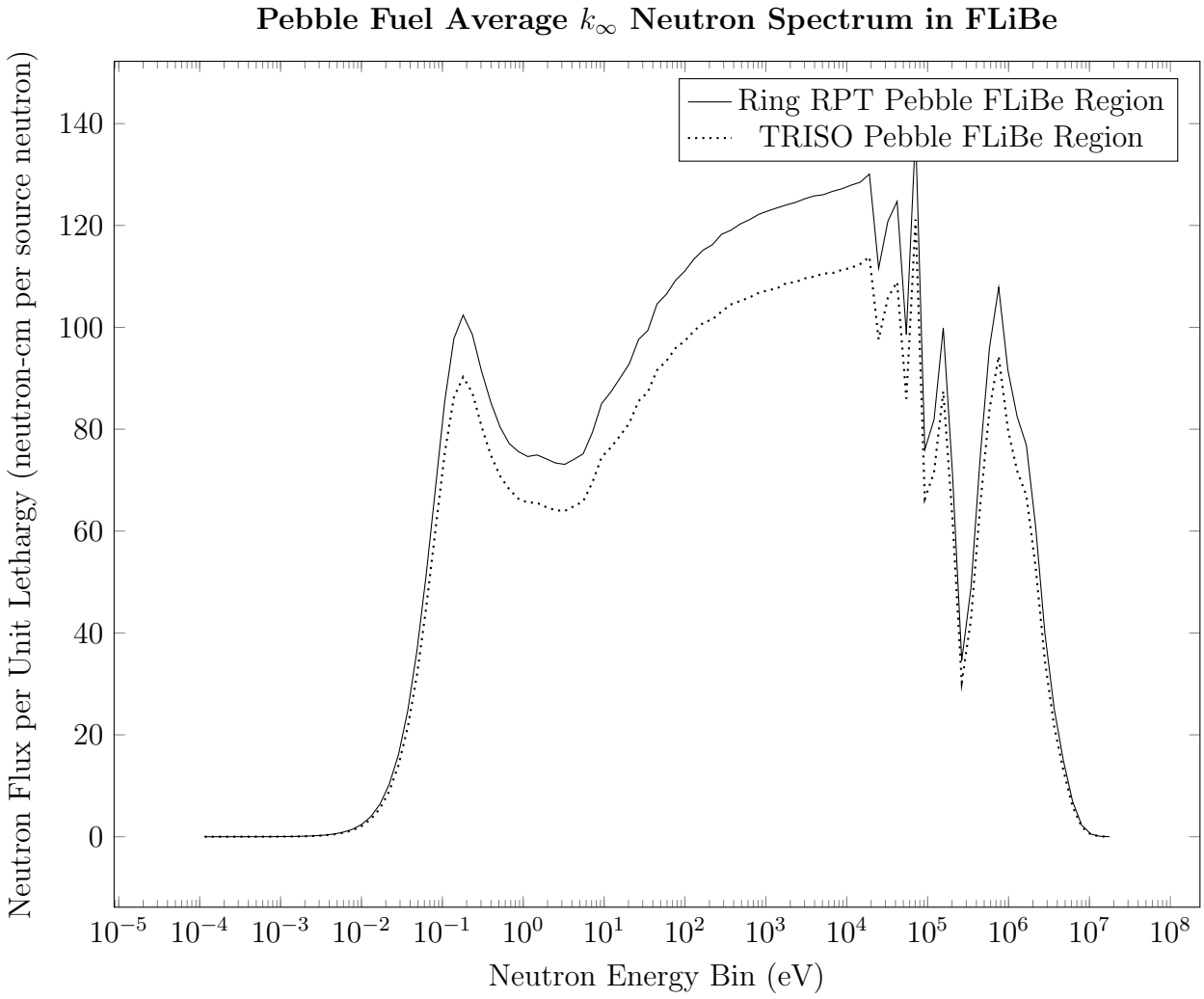


Figure 5.7: Neutron Spectrum for k_∞ for Pebble Fuel in FLiBe

In Figure 5.7, the neutron spectrum shapes for both ring RPT and TRISO pebble are quite the same shape. However, the total flux for the TRISO Pebble tends to be lower than the RPT version for all cases. We also see the same trends in the fuel region in Figure 5.8:

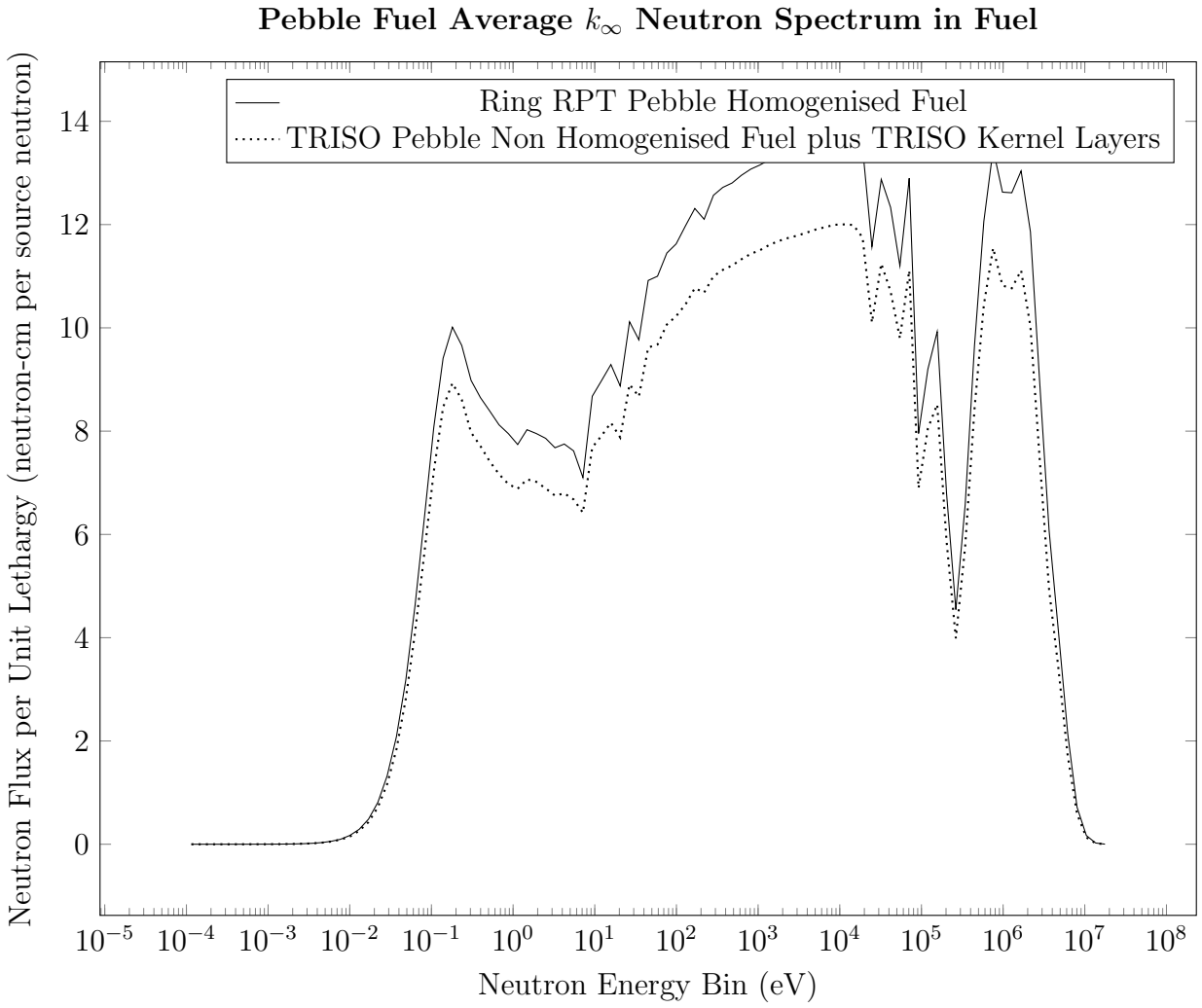


Figure 5.8: Neutron Spectrum for k_∞ for Pebble Fuel in TRISO Fuel and Homogenised TRISO Ring

The reader should note that for the fuel region plotted by Figure 5.8, I consider the TRISO Uranium Oxycarbide (UCO) kernel, SiC, buffer, and pyrolytic carbon layers all part of the "fuel region" since these layers are homogenised into the homogenised fuel region for the homogenised fuel layer for ring RPT fuel. This was done specifically for Figure 5.8 so that similar volumes and amounts of materials were used for tallying flux in both pebbles.

We can see that neutron spectrum in graphite regions, fuel regions and FLiBe regions bear a similar neutron spectrum shape qualitatively. Therefore, we qualitatively see the ratio of neutrons in the fast region, the resonance region and thermal region to be more or less equal. The characteristic Boltzmann Distribution curve in the thermal region can also be seen in both Ring RPT pebble and the reference TRISO Fuel pebble. The only difference is that the flux per source neutron is higher for the Ring RPT pebble than the reference fuel

pebble over the whole spectrum[3].

Four Factor Formula Differences Between TRISO Pebble and Ring RPT Pebble For now, we see that the neutron spectrum over between both TRISO pebble and Ring RPT pebble is more or less similar in terms of shape, except that the Ring RPT pebble has a higher flux value at all energies as compared to the original TRISO pebble. This could be due to discrepancies in resonance escape probability p for the Ring RPT pebble as compared to the TRISO pebble. To investigate whether this is the case, we can use OpenMC tallies to calculate terms in the six factor formula. For infinite medium, as we are calculating k_∞, the four factor formula would suffice. The four factor formula for infinite multiplication factor is [Duderstadt and Hamilton, 1976]:

$$k_\infty = \varepsilon p f \eta \tag{5.1}$$

Where ε is the fast fission factor, p is the resonance escape probability, f is the thermal utilisation factor and η is the fuel reproduction factor. We can estimate each term using tallies in OpenMC.When we multiply them together, we should get the infinite multiplication factor k_∞, which can be expressed in terms of tallies as:

$$k_\infty = \frac{\#\text{ neutrons produced}}{\#\text{ neutrons lost}} = \frac{\int_{reactor\ vol} dV \int_{All\ energies} dE \nu \Sigma_f \phi}{\int_{reactor\ vol} dV \int_{All\ energies} dE \Sigma_{absorption} \phi} \tag{5.2}$$

In the context of discussing k_∞, ν refers to number of neutrons produced on average per fission event, not kinematic viscosity. Σ_f refers to the macroscopic fission cross section, usually in units cm^{-1} for OpenMC.ϕ refers to neutron flux, and $\Sigma_{absorption}$ refers to macroscopic absorption cross section in the same units as Σ_f. dV is the volume element, and for this case, we perform volume integration over the whole pebble plus FLiBe layer. This becomes our "reactor". E is incident neutron energy.

ε can be expressed as the number of neutrons produced from all fissions, to the number of neutrons produced from thermal fissions [Duderstadt and Hamilton, 1976]. We can express this in terms of tallies:

$$\varepsilon = \frac{\int_{reactor\ vol} dV \int_{All\ energies} dE \nu \Sigma_f \phi}{\int_{reactor\ vol} dV \int_{thermal\ energies} dE \nu \Sigma_f \phi} \tag{5.3}$$

The fuel reproduction factor η can be calculated by comparing the number of neutrons produced during thermal fissions in the fuel, to the number of thermal absorptions in the fuel.

[3]For all neutron spectrum plots displayed in this section, about 100 energy bins were used for plot generation because I was not as interested in the resonance peaks. I was only interested to see the relative flux between fast and thermal and the general shape of the plot

$$\eta = \frac{\int_{fuel\ vol} dV \int_{thermal\ energies} dE \nu \Sigma_f \phi}{\int_{fuel\ vol} dV \int_{thermal\ energies} dE \nu \Sigma_f \phi} \tag{5.4}$$

The thermal utilisation factor f can be calculated by comparing the rates of thermal neutron absorption in the fuel to the rates of thermal neutron absorption in the whole reactor.

$$f = \frac{\int_{fuel\ vol} dV \int_{thermal\ energies} dE \Sigma_{absorption} \phi}{\int_{reactor\ vol} dV \int_{thermal\ energies} dE \Sigma_{absorption} \phi} \tag{5.5}$$

When instantiating these tallies however, it is important to specify what exactly the fuel is. This is because as the fuel itself is uranium oxycarbide (UCO), and during homogenised Ring RPT, the UCO kernels will be homogenised with the TRISO coatings to reduce spatial self shielding. We could also just define fuel as the uranium itself without the carbon and oxygen within the fuel. For this discussion in this dissertation, I will take the simple approach of considering that the fuel is the UCO material in the TRISO pebble, and the homogenised UCO material and TRISO layers in the homogenised Ring RPT pebble. I'm not going into detail here because I wanted to inspect the effect of the homogenisation process on p rather than η and f.

p is in turn the ratio of the number of neutrons reaching thermal energies without absorption in the resonance region to the number of neutrons in the fast region that are being slowed down. It can be estimated as:

$$p = \frac{\int_{fuel\ vol} dV \int_{thermal\ energies} dE \Sigma_{absorption} \phi}{\int_{reactor\ vol} dV \int_{All\ energies} dE \Sigma_{absorption} \phi} \tag{5.6}$$

We note that in this definition of p, we include neutron absorption in the resonance region resonance region and the fast region. In the fast region, it may seem as if resonances are absent if one looks at cross section library graphs, but it is good to remember that as the incident neutron energy increases, there will be more resonances which are spaced closely together. This is to the extent where the width of the resonances is larger than the differences in resonance energies for adjacent peaks. This can make resonances very difficult to distinguish from each other at high energies. At very high energies of several MeV $> 8MeV$, this effect manifests itself as giant resonances [Wong, 1998]. This means that at fast neutron energies, resonances exist also, but there is little practical use in distinguishing between the resonances. Hence, resonance peaks are averaged out in the fast region. So while resonance peaks look like they are absent in the fast neutron region, they are actually present. But it is difficult to distinguish one peak from another. Therefore, absorptions in the fast regions can also count as resonance absorptions. The most important difference, of course, between the resonances in the resonance region (with distinguishable peaks) and the fast region resonances is that Doppler broadening effects would more appreciably increase resonance capture in the resonance region with distinguishable resonance peaks.

With this in mind, we can multiply these terms together to form k_∞. Of course, to avoid confusion, we mention the fact that the neutron production rate in the fuel is equal to the neutron production rate in the whole reactor.

$$\int_{fuel\ vol} dV \int_{thermal\ energies} dE\nu\Sigma_f\phi = \int_{reactor\ vol} dV \int_{thermal\ energies} dE\nu\Sigma_f\phi \tag{5.7}$$

In this case, our "reactor" when calculating k_∞ for homogenised Ring RPT transforms is just the TRISO or Ring RPT fuel pebble with some FLiBe and a reflective boundary condition. A second consideration when instantiating these tallies is where the energy boundary should be between fast and thermal neutron energy groups. For this, when we consider that the Maxwell Boltzmann curve in the gFHR spectrum, 4.0 eV seems to be a suitable upper limit [Kile et al., 2022]. We may use that as the boundary between fast and thermal, though with an energy bound this high, it may be more accurate to designate it an "epithermal" group. This term is suitable for use since FHRs are known to operate with an epithermal neutron spectrum [X. Li et al., 2015].

Using these tallies, I plotted table 5.5:

Table 5.5: Comparison of Four Factor Formula Parameters using OpenMC Reaction Rate Tallies

Pebble Property	TRISO Pebble	Homgenised Ring RPT Pebble
k_∞	1.35009± 0.0016	1.35028± 0.0019
p	0.54495± 0.00078	0.54506± 0.00088
f	0.91144± 0.0016	0.91614± 0.0018
η	2.00765± 0.0035	1.99834± 0.0040
ε	1.35391± 0.0021	1.35316± 0.0024
ηf	1.82984± 0.0064	1.83076± 0.0062

Now, k_∞ here in table 5.5 differs significantly from $k_\infty = 1.36863 \pm 0.00070$ presented using the RPT Pebble transformation. One possible reason for this is that OpenMC uses collision, track length and absorption estimators [Romano, Horelik, et al., 2015] for k_{eff} calculations, but may use analog estimators for scoring reaction rates with energy filters. These different estimators are employed in a fashion similar to MCNP [Romano, Horelik, et al., 2015]. This is probably because estimators such as track length estimators do not require as much computational power (compared to the analog estimator method) to produce good k_{eff} statistics. However, using different estimators may then result in differing k_∞ calculations for the same geometry. Now, the track length estimator is okay for k_∞ calculations but

ill suited for reaction rate tallies requiring energy filters. Hence, analog estimators may be used for reaction rate tallies rather than track length estimators. For interested readers, a detailed discussion of estimators is found in "Estimation and interpretation of k_{eff} confidence intervals in MCNP" [Urbatsch et al., 1995], but such discussion is outside the scope of this dissertation.

With these in mind, we can now discuss the various terms in the four factor formula in table 5.5. k_∞ differs statistically insignificantly from TRISO or RPT pebble, the same goes for p and ε. This shows that at least for an infinite medium, the self shielding effects are reasonably well replicated in the homogenised Ring RPT method. Now, we consider the UCO kernel "fuel" in the TRISO pebble and homogenised fuel to be "fuel" in the homogenised Ring RPT pebble, hence, we expect f and η to differ. f would increase as the fuel volume increases in the homogenised Ring RPT pebble compared to the TRISO pebble. η should decrease because the homogenised Ring RPT fuel is more dilute in fissile material compared to pure UCO kernels in the TRISO pebble. We observe both these effects in table 5.5. We also observe that these effects cancel out as the deviation between ηf for both pebbles is statistically indistinguishable from the Monte Carlo uncertainty. Therefore, one can observe that self shielding effects are well captured at least from observing four factor formula terms in the infinite medium. We can conclude that those discrepancies in flux spectrum may be due to some other factor. For our intents and purposes, investigating the source of this discrepancy may not be so important since we have verified that the resonance escape probability is the same for TRISO Pebble and Ring RPT pebble.

There are a number of caveats of course in this four factor analysis. Firstly, we make no attempt to discuss temperature variations for these terms just yet though these are important for quantifying deviations in temperature dependent behaviour between homogenised Ring RPT pebbles and their TRISO pebble counterparts. This is because studying self shielding and ensuring maximum simulation accuracy does not contribute to the main goals of the dissertation. I am content for now to present the four factor formula comparison as is to justify that the homogenised Ring RPT pebble is a good enough transform for the TRISO pebble for the purposes of this dissertation. Secondly, we also make no attempt to compare leakage or scattering for both these pebbles. Again, this may be important for accurately determining scattering MGXS and mean free paths, but the accuracy of the multiphysics model is not the main goal of the dissertation, we just want to demonstrate that the data fit surrogate model can replicate behaviour of the multiphysics model. For this, I just wanted to perform the homogenised Ring RPT process so that I could set the arbitrary reactor simulation up for testing. To check if the homogenised Ring RPT was done properly, we had to briefly check whether k_∞ matched and as a sanity check, inspect the degree to which self shielding is replicated. Indeed, looking at the neutron spectrum and resonance escape probability, the self shielding behaviour is indeed replicated well at least for an infinite medium.

Conclusion for Homogenised Ring RPT Process

Since the k_∞ deviation of the RPT pebble and TRISO Fuel pebble of 39 pcm matches within error of 2σ of about 130 pcm. The neutron spectrum plots also show a similar degree of thermalisation within the graphite, FLiBe and fuel regions. We also analysed some terms in the four factor formula and showed that resonance escape probability was replicated well for the RPT pebble in an infinite medium. Hence, we can say that the RPT method was performed successfully.

In future work, where more rigour is desired, we could compare the neutron spectrum and k_∞ of both TRISO pebble and Ring RPT pebble at other temperatures of interest to see if the fuel and coolant temperature feedback is similar for both. This is not done here for the sake of brevity as we are only interested in performing a "dry run" iteration. We could in fact compare TRISO Fuel pebbles and Ring RPT pebbles in a small reactor setup to see if they behave similarly under temperature perturbation. One could do this in a real study if one had the computational resources and time to do so. However, it is not done here also because we are only doing a "dry run". Such work is relegated for future work where constructing a high fidelity reactor is more the focus of the project or paper.

Arbitrary Reactor Construction in OpenMC

Now that we have some assurance that Ring RPT was performed successfully, we can then construct the OpenMC model of our arbitrary reactor. The flux spectra of the Ring RPT Pebble Bed Reactor will be compared to flux spectra of Pebble Bed FHRs in literature such as the Thorium Molten Salt Reactor Solid Fuelled (TMSR-SF) simulations [X. Wang, 2018] and gFHR [Kile et al., 2022] so that we can have some confidence that the thermalisation process and self shielding effects of the Ring RPT Homogenised Reactor is similar to Pebble Bed FHRs found in literature.

The arbitrary reactor is meant to be a cylindrical reactor with entrance and exit Regions for FLiBe to flow in and out of it. It is then surrounded by a graphite reflector. For simplicity, no control rods are added at this point in time. This can perhaps be added in future iterations where transients involving control rods are to be simulated. For this simple demonstration case, an arbitrary reactor with simple geometry is constructed using the Ring RPT pebble. Furthermore, the pebbles are arranged in simple hexagonal lattices[4] because the packing fraction is about 0.605 [Abedi and Vosoughi, 2011] with pebble clipping permitted. This is similar to the HTR-PM's packing fraction of 61% [Hao, Yang, and Y. Cheng, 2022]. Therefore, simple hexagonal (SH) lattices were used in MCNP calculations for HTR-PMs [Abedi and Vosoughi, 2011]. We assume that the random packing in HTR-PMs and FHRs are quite the same, so we will just use SH lattices in this demonstration for

[4] Simple Hexagonal (SH) lattices are not the same as Hexagonal Closed Packed (HCP) lattices, HCP lattices have packing factor of 0.74 rather than 0.61 for SH lattices

simplicity[5]. The resulting OpenMC reactor geometry can be seen in Figure 5.9, Figure 5.10 and Figure 5.11:

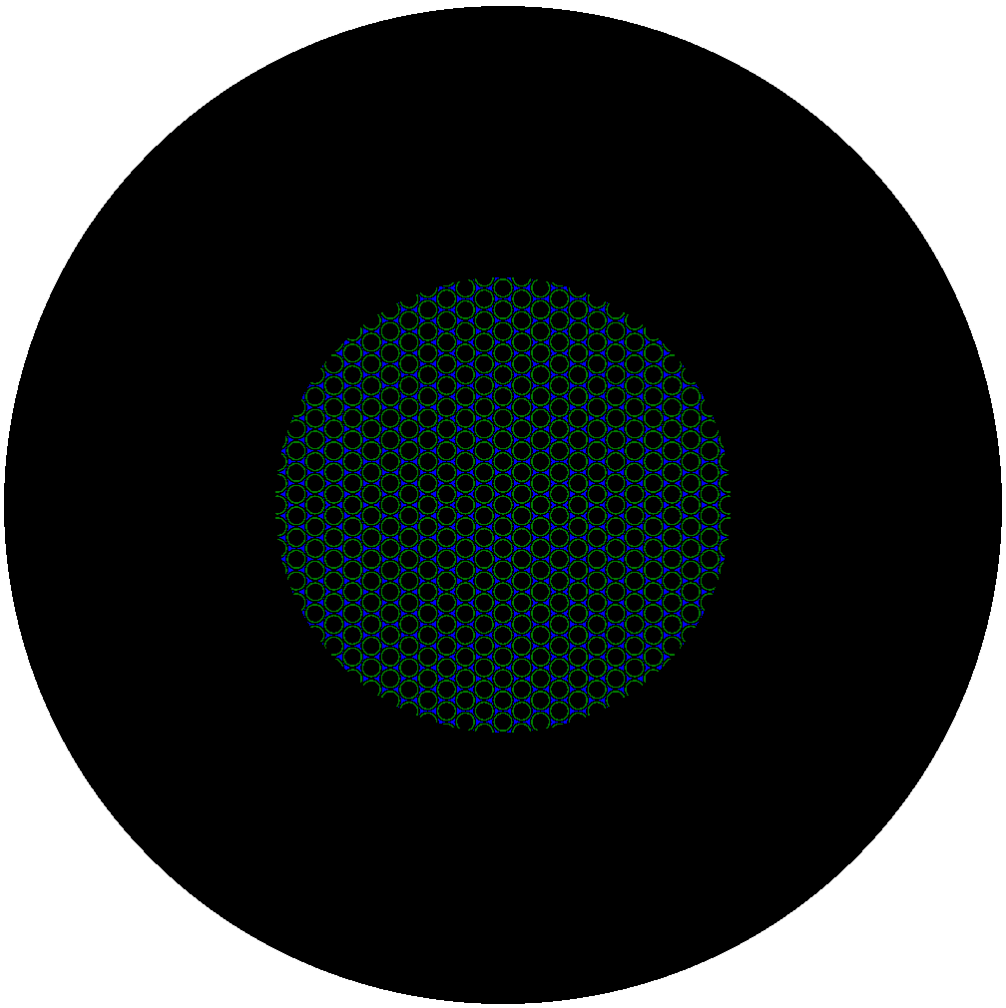


Figure 5.9: Arbitrary Reactor Cross Section on XY plane

[5]These SH lattices have also been used before to simulate PB-AHTR geometry [Fratoni, 2008]. Here, pebbles flow through graphite channels rather than a randomly packed core. While the geometries are fundamentally different, the packing fractions are remarkably similar.

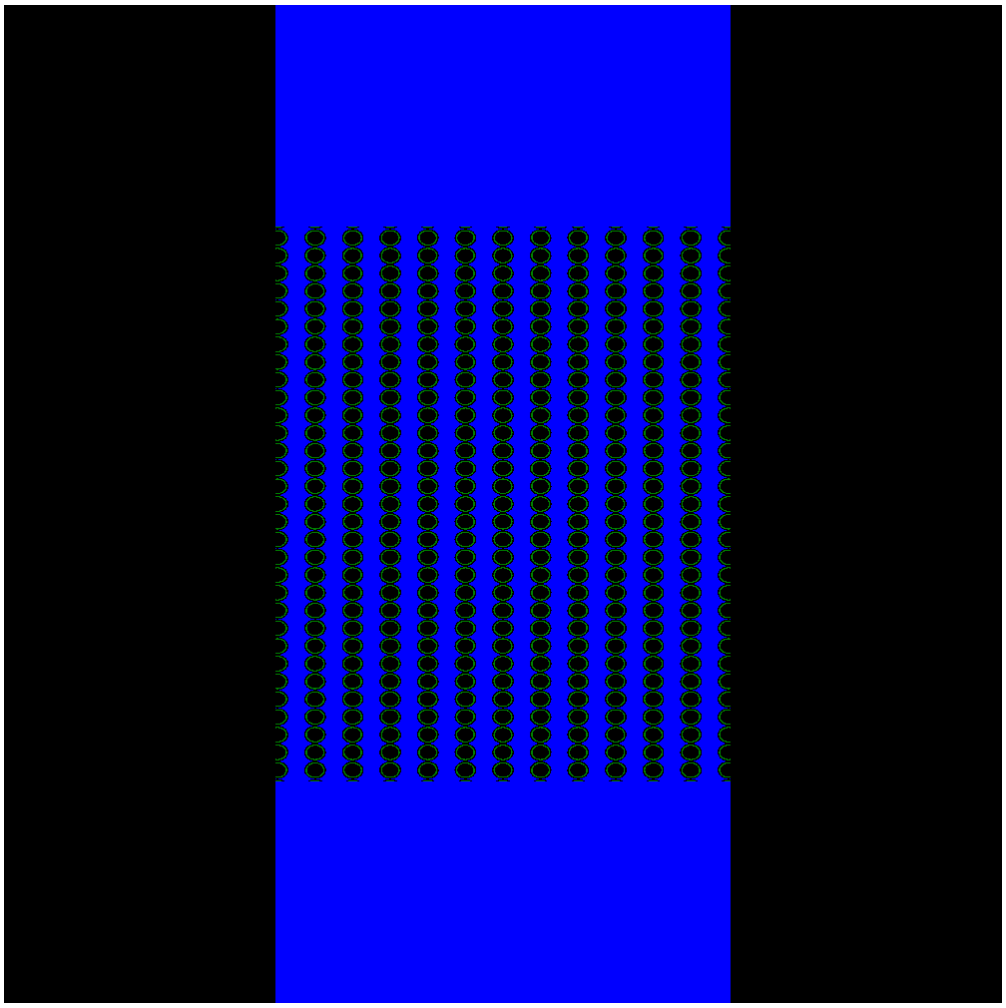


Figure 5.10: Arbitrary Reactor Cross Section on XZ plane

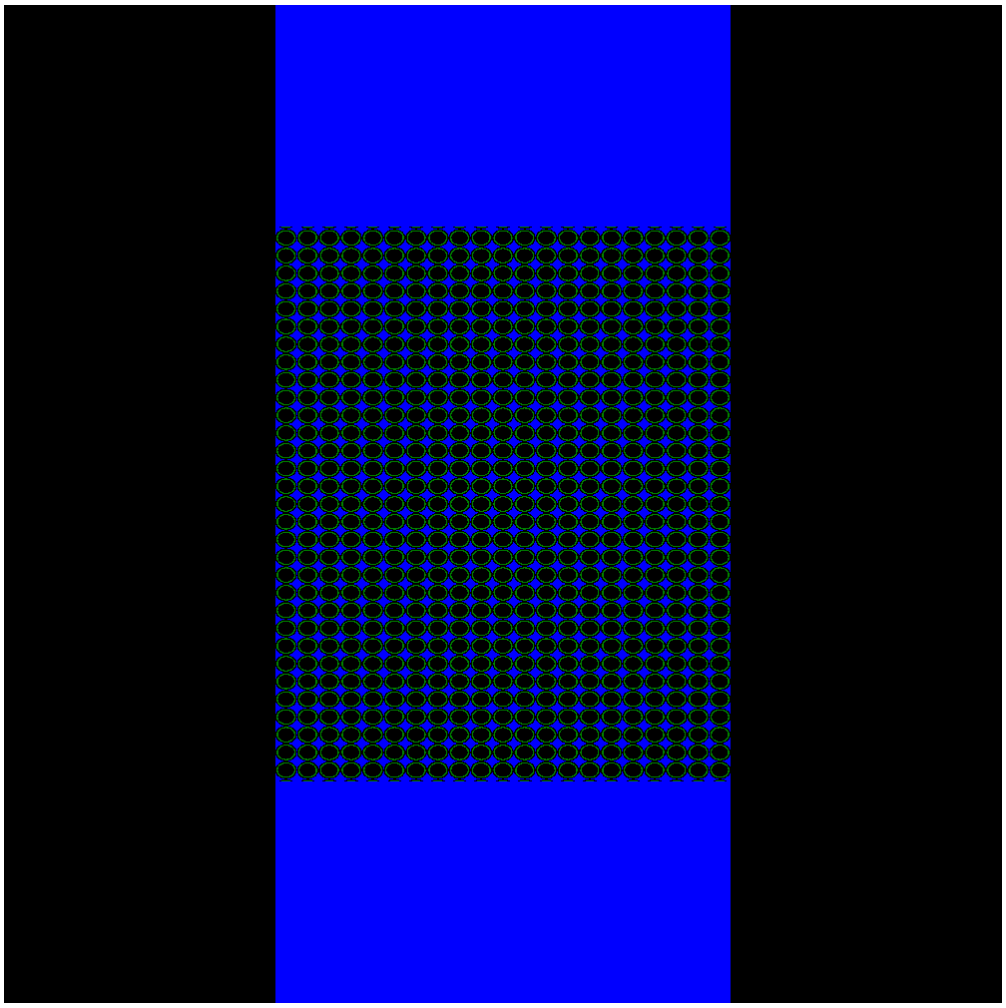


Figure 5.11: Arbitrary Reactor Cross Section on YZ plane

Now, homogenised Ring RPT was performed in the hope that self shielding would be at least partially accounted for in the transform process. This was verified to work for an infinite medium. To verify if the Ring RPT works for a finite medium, it would need to produce a neutron spectrum with a shape qualitatively similar to other similarly designed reactors such as the gFHR [Kile et al., 2022] or TMSR-SF [X. Wang, 2018]. To check if the Ring RPT method produces a reasonably correct neutron spectrum, we run eigenvalue simulations and produce neutron spectrum plots for the arbitrary reactor in Figure 5.12. For each arbitrary reactor simulation, I used 50 inactive batches, 150 active batches and 10,000 source particles each.

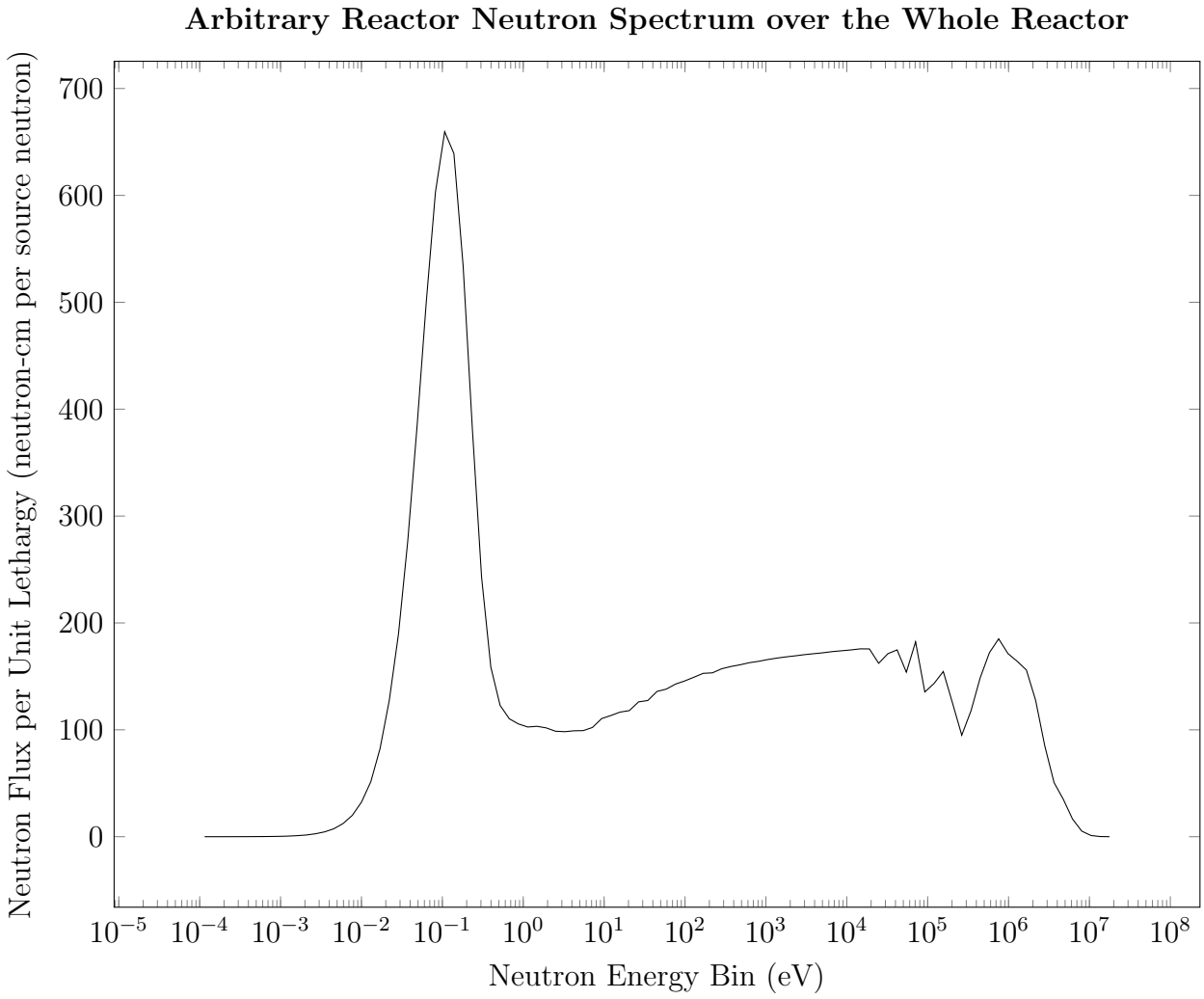


Figure 5.12: Neutron Spectrum for Arbitrary Reactor over Whole Reactor

Figure 5.12 was done using 100 energy bins, and was just meant to check the flux shape. Resonance peaks are not meant to be simulated in high resolution. Now, we see in Figure 5.12 that the flux shape is similar to that found in literature for the gFHR [Kile et al., 2022]. Here, we have a prominent Boltzmann Distribution characteristic of thermal neutrons. The end upper energy boundary of this Boltzmann Distribution is about 2.5 to 4.0 eV which is similar to that found in literature. This gives us some confidence that the Homogenised Ring RPT pebble bed would give a similar reactor feedback behaviour to a pebble bed with fully simulated TRISO particles.

We can also further check the shapes of the neutron spectrum plots are tallied based on cells in Figure 5.13 and compare it to spectra found in literature.

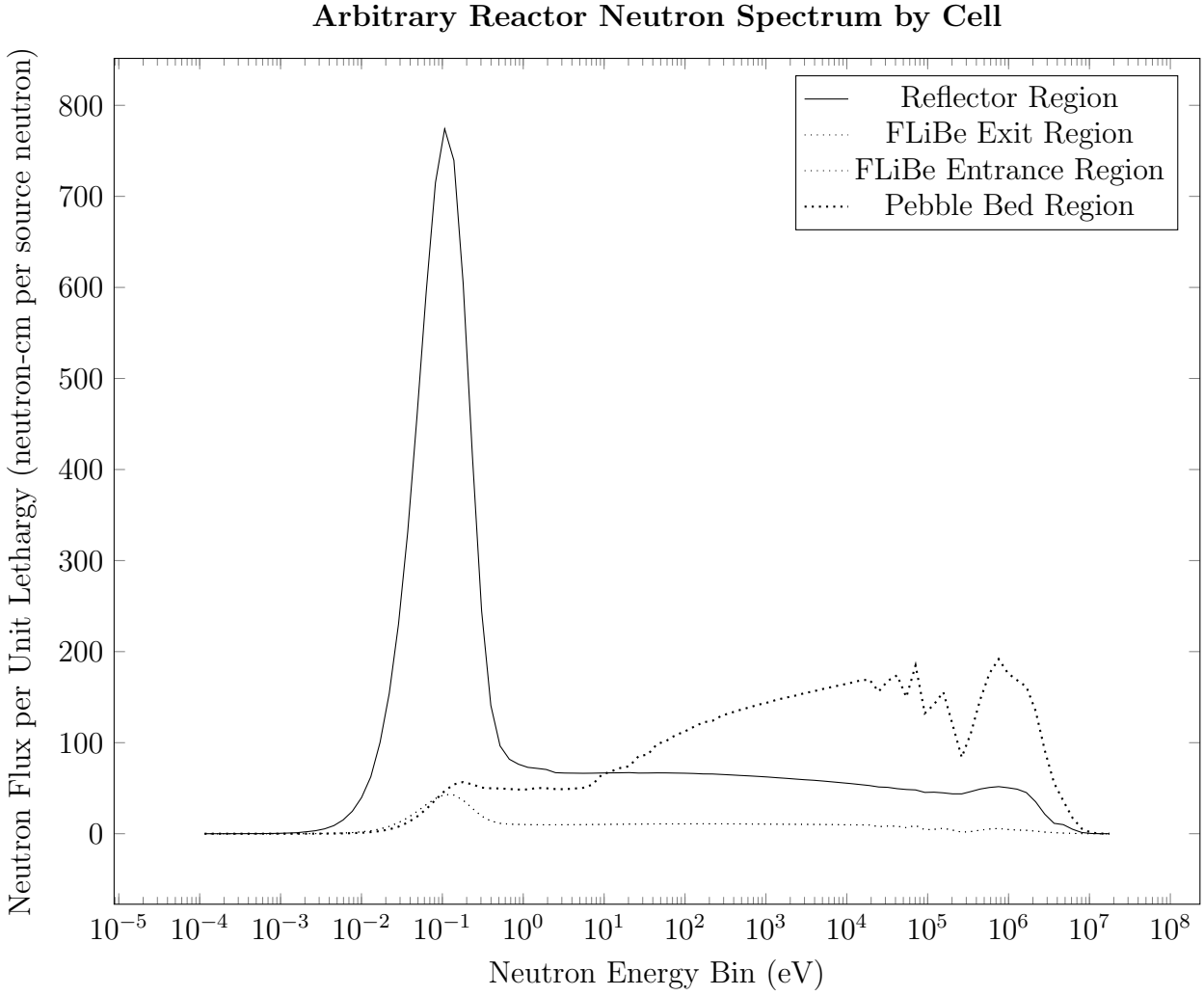


Figure 5.13: Neutron Spectrum for Arbitrary Reactor by Cell

We can see in figure 5.13, the neutron spectrum in the reflector region is predominantly thermal whereas the neutron spectrum in the fuel region is predominantly fast with a few resonance peaks. This flux shape for the reflector was in agreement with reflector neutron flux shapes in literature where the TMSR-SF was simulated [X. Wang, 2018].

Nevertheless, in the TMSR-SF simulations, there was a Maxwell Boltzmann peak present which was significantly higher than the flux in the resonance regions [X. Wang, 2018]. While a small Maxwell Boltzmann peak was present here, it was nowhere near as high as those found in literature. One explanation could be that the TMSR-SF had more graphite reflecting and moderating flux back in the entrance and exit regions of the reactor as compared to this arbitrary reactor model.

These plots show that the Ring RPT Pebble Bed Reactor can produce similar neutron spectra to those found in literature. Hence, we can have some degree of confidence in using

these reactors to generate MGXS.

Cross Section Generation for the Arbitrary Reactor

Preliminaries

In the generating MGXS we normally weight cross sections with scalar flux. Take the total cross section for example:

$$\Sigma_t = \frac{\int_V dV \int_{E_g}^{E_{g-1}} dE \phi(\vec{r}, E, t) \Sigma_t(\vec{r}, E, t)}{\int_V dV \int_{E_g}^{E_{g-1}} dE \phi(\vec{r}, E, t)} \tag{5.8}$$

This process is done in OpenMC to generate MGXS.For this work, we homogenise cross sections in a rather crude manner. Σ_t is averaged over the whole OpenMC cell or GeN-Foam cellZone. These are the fuel region, FLiBe inlet region, FLiBe outlet region and reflector region. If Σ_t or some other cross sections vary within the region, then this procedure is bound to have some approximation error. This error will not exist only if the cross sections were not a function of $\vec{r}$, we can indeed take the cross sections out of the volume integral.

$$\Sigma_t = \frac{\int_V dV \int_{E_g}^{E_{g-1}} dE \phi(\vec{r}, E, t) \Sigma_t(E, t)}{\int_V dV \int_{E_g}^{E_{g-1}} dE \phi(\vec{r}, E, t)} \tag{5.9}$$

If the cross sections do not vary with position, then we can effectively homogenise the entire core and obtain an exact expression. In reality, cross sections vary with position, as they do in a heterogeneous reactor. We then realise that homogenisation across the core will, in general, not produce accurate results [K. S. Smith, 1986]. This makes homogenisation across the core quite unsuitable for accounting for pebbles of different burnup or differing degrees of reactor poison accumulation, for example. Therefore, ideally speaking, different zones of volume integration are required for different materials. We do not concern ourselves with this for the time being as it does not fit the goals of this dissertation.

For now, we perform a two group MGXS generation as it is the simplest case of MGXS generation. This was done with 50 inactive batches, 150 active batches and 10,000 source neutrons per batch to achieve source convergence. We use the Maxwell Boltzmann distribution of the reactor to determine the boundaries for our thermal groups. Monte Carlo simulation shows 2.5 to 4.0 eV boundary for temperature of 600K. This may be a good boundary for thermal and fast spectrum neutrons. However, the neutron spectrum may harden as temperatures increase, and therefore the fast and thermal boundary may be quite temperature dependent. Therefore, for the boundary between fast and thermal neutrons, we may want to have it at a higher energy boundary. I took inspiration that in 8 or 9 group calculations for molten salt cooled pebble bed reactors [X. Wang, 2018], an energy group with higher bound of 4.0 eV was used to capture low lying resonances as well. This should give sufficient leeway for any sort of spectrum hardening. I also skipped the simulation of a neutron spectrum at 1200K to expedite the development and demonstration of this example.

In study where the emphasis is on accuracy, these graphs should be plotted and compared to check for the boundary between fast and thermal neutrons. For more than two groups, one should check the energy boundaries for the Boltzmann distribution as well as adjust the energy group structure to account for low lying resonances. Interested readers can visit Wang's dissertation [X. Wang, 2018] for an example of an 8 group model for the Thorium Molten Salt Reactor Solid Fuelled (TMSR-SF). I did not do any of the fine energy group modelling because this is only a demonstration case. Two group diffusion was used because it was relatively simple to setup, and yet it is the bare minimum energy discretisation for neutrons that can account for the effect of neutron cross section based on neutron energy. Given all the above factors, I used 4 eV as my boundary between fast and thermal (or epithermal) neutrons.

GeN-Foam MGXS Input Requirements

We are generating MGXS for GeN-Foam, hence we want to start by understanding the input format of GeN-Foam MGXS files, otherwise known as the "nuclearData" input files. These files contain the cross section information required for the deterministic simulation. Moreover, GeN-Foam uses these files to interpolate cross sections which may change due to fuel temperature feedback and coolant void feedback. These files are "nuclearDataFuelTemp" for fuel temperature feedback, "nuclearDataTCool" for coolant temperature (without expansion) and "nuclearDataRhoCool" for coolant void or density feedback. Again, the reader should note that expansion feedback is neglected in this work. Quantification and simulation of structural expansion feedback mechanisms are relegated to future studies.

In each of the nuclearData input files, the user is expected to specify the solid fuel fraction (fuelFraction) inverse of velocity $\frac{1}{v}$ (IV), Diffusion Coefficient (D), and several other properties for each group. An example is shown here for the fuel_cell region:

```
fuel_cell {
fuelFraction 0.61 ;
IV nonuniform List<scalar> 2 (
3.304179060252947e-06 0.00013921702249789254 );
D nonuniform List<scalar> 2 (
0.012083545941701444 0.010801380798703051 );
nuSigmaEff nonuniform List<scalar> 2 (
0.2295056339627519 3.2189494876053133 );
sigmaPow nonuniform List<scalar> 2 (
2.9101924511187916e-12 4.093464684696107e-11 );
scatteringMatrixP0 2 2(
(29.197094830484378 0.12178685124792211 )
(0.004719753596159622 30.310442485169414 )
);
sigmaDisapp nonuniform List<scalar> 2 (
```

```
0.3970189096562604 1.657938605609266 );
chiPrompt nonuniform List<scalar> 2 (0.9999999999999949 0.0 );
chiDelayed nonuniform List<scalar> 2 (0.9999999999999949 0.0 );
Beta nonuniform List<scalar> 6 (
0.00022786363922605484 0.0011808230019307714
0.0011299792880896187 0.0025439311569539894
0.0010539386895804413 0.00044111556283448633 );
lambda nonuniform List<scalar> 6 (
0.012466675909351533 0.02829172165550797 0.04252436690551811
0.2924671647932258 1.6347810862262862 3.5546009259484372 );
discFactor nonuniform List<scalar> 2 (1 1 );
integralFlux nonuniform List<scalar> 2 (1.0 1.0 );
}
```

$\nu\Sigma_f$ (nuSigmaEff) is the average number of neutrons produced per fission ν multiplied by macroscopic fission cross section Σ_f. Σ_{pow} (sigmaPow) is the Σ_f multiplied by energy released per fission. The zeroth moment scattering matrix (scatteringMatrixP0) contains coefficients for scattering cross section Σ_s within the energy groups and between energy groups. The removal cross sections are represented by sigmaDisapp. Thereafter, the fission spectrum for delayed neutrons (chiDelayed) and prompt neutrons (chiPrompt) is put in. The delayed fractions (Beta) and their corresponding decay constants (lambda) is also given. The discontinuity factor (discFactor) is meant to correct inaccuracies in the diffusion approximation during spatial homogenisation [Fiorina, Kerkar, et al., 2016; K. S. Smith, 1986]. Consideration of such factors is neglected for this work, so that it is just set to 1. The integralFlux entry, upon examining source code, is used in Aitken Acceleration. Aitken Acceleration is used in GeN-Foam to speed up solution for transient solvers. This option is disabled for this work (it is disabled by default) and the integralFlux value was left at 1.0 after copying over this entry from a tutorial case.

For the rest of the dimensioned entries, the user must note that the length scales used in OpenMC are in centimeters, while the length scales used in GeN-Foam are in meters. While OpenMC does have MGXS generating libraries, these are in differing units from GeN-Foam. Additionally, different syntax is given for OpenMC's python API and GeN-Foam as GeN-Foam's input file syntax behaves more like C rather than python. These make it quite cumbersome to translate the MGXS data from OpenMC format to GeN-Foam format manually. To automate the process, a python script was written to generate these GeN-Foam nuclearData input files automatically.

OpenMC to GeN-Foam MGXS Input Write Python Script

We shall now describe a python script which converts OpenMC MGXS data into GeN-Foam input files. In such a script, the first thing to do would be to use the OpenMC MGXS libraries to generate appropriate tally inputs for the simulation prior to running OpenMC.After the

simulation is run, OpenMC would generate hierarchical data format 5 (hdf5) files which store information about the tallies. The native OpenMC MGXS libraries would information from the simulation and generate the corresponding MGXS.Thereafter, I would perform some unit conversion to convert the OpenMC units to GeN-Foam units. For example, we convert cm to m for length scales and electron-volts to joules for energy.

An example of the python function to return sigmaPow is given:

```
def printSigmaPow(self,cellIndex,printData=False):

    # this prints the GeNfoam input for Sigma f multiplied by
    # the power per fission coefficient in J/m

    description='sigmaPow nonuniform List<scalar>'
    dimensions=' '+str(self.nEnergyGroups())+' '
    dataString='('
    ev2j = 1.602176487e-19
    cm2m = 0.01

    for i in range(self.nEnergyGroups()):
        # load sigmaPow coefficient in eV cm^-1 units
        sigmaPowUnitsOpenMC=self.load_sigmaPow_value(i+1,cellIndex)
        # now convert this to SI units
        # conversion factor is *1.6e-19 eV/J *100 cm/m
        sigmaPow=sigmaPowUnitsOpenMC*ev2j/cm2m
        dataString+=str(sigmaPow)+' '

    dataString+=');\n'

    fullString=description+dimensions+dataString
    print(fullString)

    if printData == True:
        print('sigmaPow is printed for the following cell: ')
        print(self.getCell(cellIndex))
        print('in units of J m^-1')

    return fullString
```

In the code excerpt, sigmaPow is loaded from OpenMC and converted to GeN-Foam units for a given cell. We then concatenate the strings together for all MGXS over an OpenMC cell, which could be our reflector region for example.This process is repeated for all cells we wish to obtain MGXS data for. For this work, it would be the entrance and exit FLiBe

regions, and the fuel region as a whole. The resulting string is then written to a file with the output name "nuclearData". The "nuclearData" file is written for MGXS at 600K due to lack of thermal scattering data for carbon at other temperatures.

We then repeat the process for generating "nuclearDataFuelTemp" by increasing the fuel temperature (the whole pebble) to 1200K and applying this script. Now this, of course is rather crude, since the fuel temperature specifically refers to UCO within the fuel rather than the whole pebble. Technically speaking, the carbon matrix of the pebble acts more as a moderator rather than the fuel itself. However, GeN-Foam did not contain files for varying MGXS based on changing moderator temperature. Since the fuel pebble would be more or less similar in temperature to the UCO, a crude approximation would be to simply assume that they are the same temperature. This was motivated out of expediency rather than accuracy.

For "nuclearDataTCool", it was found that altering the coolant temperature without changing its density did not change the two group MGXS much. Hence, "nuclearDataTCool" was left blank. By default, GeN-Foam interprets blank perturbed nuclearDataTCool files to mean that MGXS does not change with coolant temperature and constant density.

For "nuclearDataRhoCool" the FLiBe density was altered to approximately 1.827 g/cm^3 and the MGXS generation process was repeated. The FLiBe temperature remained at 600K.

GeN-Foam also contains input files such as "nuclearDataCladExp", "nuclearDataAxialExp" and "nuclearDataRadialExp" to account for cladding expansion, as well as reactor expansion in axial and radial directions. These files are left blank to indicate to GeN-Foam that MGXS does not change with these perturbations.

To test if the coefficients were somewhat correctly translated from OpenMC to GeN-Foam, a simple test case of a bare critical sphere was simulated in OpenMC and then translated to GeN-Foam for a one group case. An eigenvalue calculation was run on the Godiva sphere test case with extracted MGXS.The k_{eff} for this was 0.998372, which is quite reasonable. During debugging of the code, k_{eff} would sometimes go as high as 2.2 due to conversion errors of MGXS.This gives us a ballpark figure of how much error to expect if the python MGXS code translates MGXS wrongly between OpenMC and GeN-Foam. As an additional test, the arbitrary reactor geometry, generated using a very coarse mesh, yielded a k_{eff} of 1.066352. This is not ideal at about 6000 pcm difference from OpenMC's k_{eff}, but compared a 100,000 pcm error that occurred during erroneous translation of MGXS, this is small. So it is likely that the errors would have arisen from elsewhere rather than MGXS translation errors.

GeN-Foam Geometry and Mesh Construction

For model building for GeN-Foam, we wish to use Computer Aided Design (CAD) to replicate the arbitrary reactor geometry constructed using constructive solid geometry (CSG) in OpenMC.While OpenMC has the ability to use CAD geometries, it was more expeditious to use OpenMC's CSG for the arbitrary reactor. This arbitrary reactor geometry is relatively simple since it only consists of enclosed cylinders, and therefore is easy to construct.

For the arbitrary reactor, we have a central fuel (pebble bed) region and cylinder. This has a diameter of (D) 140 cm and height (H) of 135 cm. The reflector is 70 cm thick, making for an outer diameter (OD) of 280 cm. This is best seen in Figure 5.14:

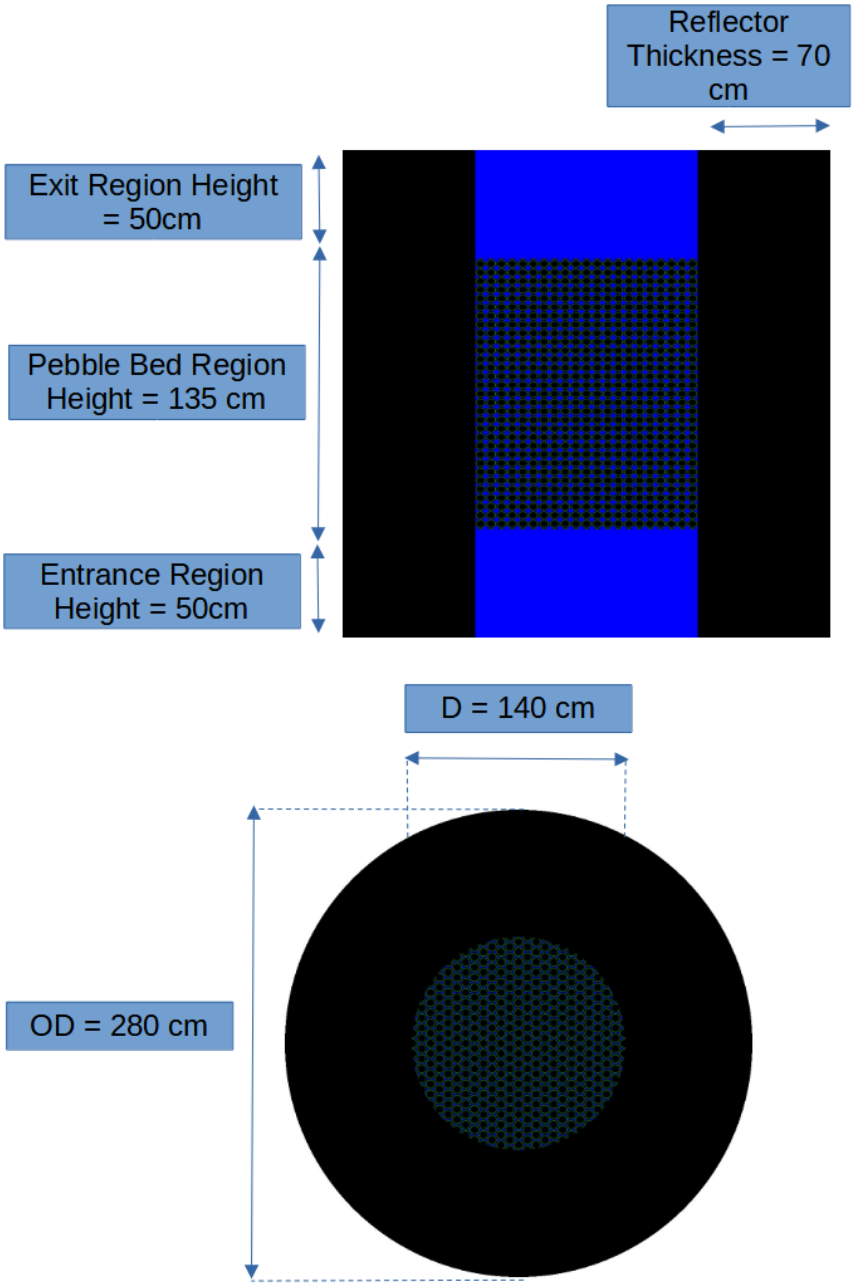


Figure 5.14: Arbitrary Reactor Schematics

FLiBe flows into the reactor through an entrance region with the same diameter of 140 cm, but an entrance length of 50 cm. After it exits the core, FLiBe then exits through an exit region with the same 140 cm diameter and exit length of 50 cm. The FLiBe entrance, fuel

region and FLiBe exit regions are surrounded by a cylindrical graphite reflector also 70 cm thick in the radial direction and 50 cm thick in the axial direction as shown in Figure 5.14. There are no control rods in this setup for simplicity's sake. Lengthscales for the reflectors and entrance/exit regions are mostly such that they are on the same order of magnitude as the lengthscale of the core.

We first constructed the mesh by using FreeCAD to construct the geometry, and then importing into SALOME platform to generate a crude tetrahedral mesh. This process is summarised in Figure 5.15:

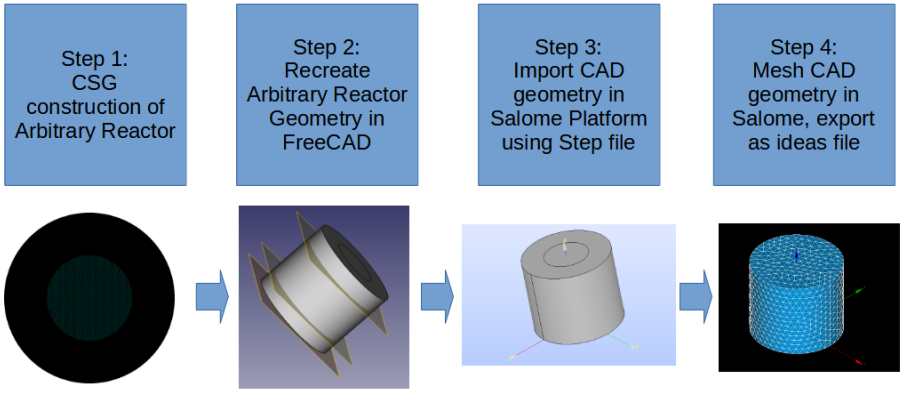


Figure 5.15: Meshing Process using FreeCAD and Salome

Salome can export the mesh via an ideas File which can be converted to OpenFoam's mesh files in the polymesh folder using ideasUnvToFoam. No mesh refinement study was done in this case, and the mesh is rather coarse for expedited testing. Any mesh refinement study is relegated to future work.

GeN-Foam Inputs other than MGXS

Besides the mesh geometry and the MGXS inputs, GeN-Foam still requires other inputs in order to simulate the arbitrary reactor.

Solid Phase Thermal Hydraulics Properties

The first of these inputs is the thermal hydraulics properties we wish to discuss are that of the solid phase. These inputs will be stored in a file called "fluidRegion". The reader should note that the "fluidRegion" contains information for both solid and fluid thermal hydraulics calculations within porous media. It is perhaps better to consider it a "porousFluidRegion" so that it is less confusing.

For the heat generating solid media of the core, we adapt a 1D generating media heat conduction model:

$$\rho c_p \frac{\partial T}{\partial t} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(kr^2\frac{\partial T}{\partial r}\right) + q''' \tag{5.10}$$

Thermal Conductance and Conductivity Here, q''' represents the heat generation term, r is the radius, T is temperature, ρ is density, k is thermal conductivity and c_p is specific heat capacity. Equation 5.10 of course assumes homogenised media. However, the TRISO pebble is quite heterogeneous. Nevertheless, in the interest of expediency, a homogenised model was assumed for the thermal hydraulics for this iteration. We therefore use a volume weighted thermal conductivity [M. Liu et al., 2019].

$$k_{Thermal\ effective} = \frac{V_{TRISO}k_{TRISO} + V_m k_m}{V_m + V_{TRISO}} \tag{5.11}$$

Equation 5.11 uses the total volume of the particles V_{TRISO}, total volume of the graphite matrix V_m, thermal conductivity of the TRISO particles k_{TRISO} and matrix k_m in order to obtain a volume weighted average thermal conductivity $k_{Thermal\ effective}$. We can express equation 5.11 in terms of packing fraction f:

$$k_{Thermal\ effective} = fk_{TRISO} + (1-f)k_m \tag{5.12}$$

Where for equation 5.11, f = $\frac{V_{TRISO}}{V_{TRISO}+V_m}$. These thermal conductivities, along with other properties such as the density ρ and heat capacity c_p will be used to calculate the thermal resistance in a nodalised thermal conductance (or indirectly, thermal resistance) model by GeN-Foam [Robert et al., 2023]. A two node thermal conductance or resistance model is shown in Figure 5.16:

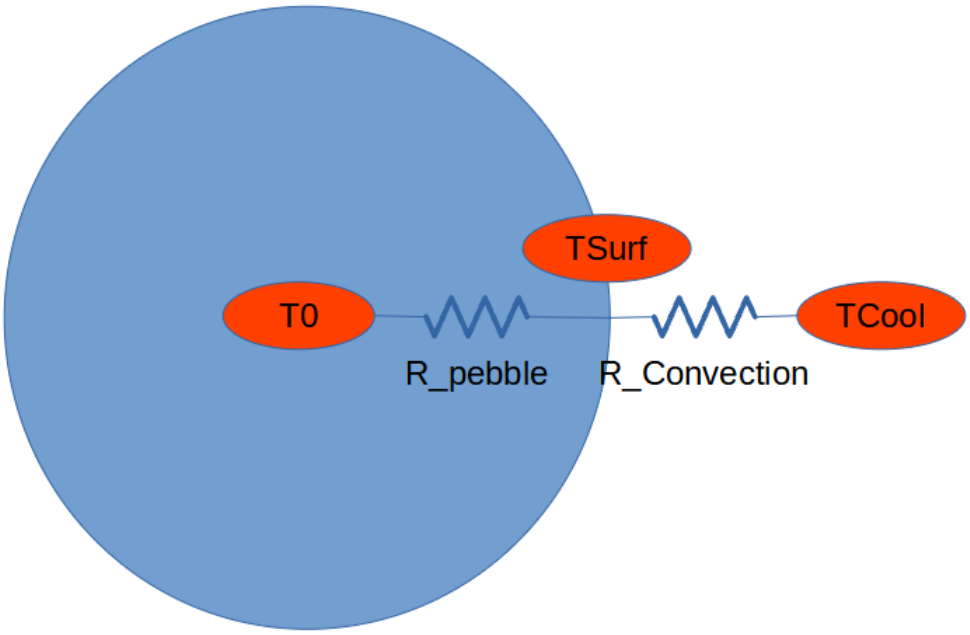


Figure 5.16: Two Node Thermal Conductance (Resistance) Model

A two node model is used because this is the minimum number of nodes required that does not assume lumped capacitance of the pebble. Now, in GeN-Foam, the surface temperature is not considered a "node". In GeN-Foam, each node must have a characteristic temperature and thermal inertia (volumetric heat capacity). The surface does not have volumetric heat capacity and is therefore not considered a "node". Hence, this two node model is considered a one node model in GeN-Foam. For GeN-Foam in particular, the requested input for thermal conductance is the inverse of thermal resistance per unit volume of the pebble (V_{pebble}). For a spherical shell, the 1D thermal conductance per unit volume (Hs) can be expressed as:

$$\text{Hs} = \frac{k4\pi r_1 r_2}{r_1 - r_2} \frac{1}{V_{pebble}} \tag{5.13}$$

Where conductance is measured between the radius r_1 and r_2. Also, $r_1 > r_2$. In equation 5.13, k represents thermal conductivity of the layer. As of the time of writing this dissertation, GeN-Foam lacks the ability to adjust thermal conductance based on temperature. This can of course, be coded in manually. However, in the interest of expediency, this was considered out of scope for this present work. We can improve upon model fidelity and accuracy in future iterations. For now, some averaged value of thermal conductance will be used.

To calculate thermal conductance, we also require two reasonable bounds for r_1 and r_2. r_1 was chosen as 2 cm because the pebble is 4 cm in diameter, and the outermost node needs to be the surface temperature. For a two node model, we need to assume the inner node with temperature T0 as shown in Figure 5.16, to behave as a lumped capacitance

body. That is to say r_2 should be decided such that the inner sphere has a spatially uniform temperature at all times. We could do some scaling analysis and derive some dimensionless number analogous to the Biot Number. However, I chose a guess of r_2 =0.2 cm to expedite development of this present reactor.

For calculation of the thermal conductances, plenty of libterty was taken besides what was already mentioned. We calculate a rough guess of the effective thermal conductivity $k_{Thermal\ effective}$ assuming the TRISO particles have a similar thermal conductivity to UO_2 of 3.86 $W/(mK)$ [M. Liu et al., 2019], and also that the graphite matrix thermal conductivity is 25 $W/(mK)$. With a packing fraction of 0.3, we get a $k_{Thermal\ effective} = 18.658\ W/(mK)$. Using the $r_1 = 2$cm and $r_2 = 0.2$ cm, the thermal conductance H was estimated at 0.521029 W/K. With a pebble volume based on $r_1 = 2$cm, Hs = 15548 $W/(m^3 \cdot K)$.

In future reactor iterations, we might redo this analysis by considering correlations for thermal conductivity of UCO, SiC, pyrolytic carbon and buffer layers from literature [W. Jiang et al., 2021]. Furthermore, we may consider not homogenising the entire medium and accounting for thermal conductivity variation with temperature. For now, however, we shall not consider these as it can be quite time consuming and not critical to the goals of this dissertation.

Specific and Volumetric Heat Capacity Next, we shall discuss c_p. We wish to be more meticulous for c_p and ρc_p because we want to match the thermal inertia of the core. For c_p, correlations for various stoichiometric ratios of U, C and O were available for UCO [W. Jiang et al., 2021]. Substituting our stoichiometric ratio of $U_{0.3333}C_{0.1667}O_{0.5}$, we obtain the following correlation for c_p using reduced temperature in kelvin T_K:

$$c_{p,UCO}\left(\frac{J}{kgK}\right) = \frac{52.1743 + 87.951T_K - 84.211T_K^2 + 31.542T_K^3 - 2.633T_K^4 - \frac{0.71391}{T_K^2}}{0.2675} \tag{5.14}$$

Where $T_K = \frac{T(K)}{1000}$.

Seeing how GeN-Foam does not allow for temperature variation, we have to average it using:

$$c_{p,ave} = \frac{\int_{T_1}^{T_2} c_p dT}{T_2 - T_1} \tag{5.15}$$

A simple way for interpolation is to use Simpson's $\frac{1}{3}$ rule for the integral [Uddin, Moheuddin, and Kowsher, 2019]. The resultant expression for $c_{p,ave}$ is:

$$c_{p,ave} \approx \frac{1}{6}\left(c_p(T_1) + c_p(T_2) + 4c_p(\frac{T_1 + T_2}{2})\right) \tag{5.16}$$

For UCO, I would select the temperature bound to be $700°C$ to $1000°C$ as these are reasonable guesses of temperatures which the fuel experiences during normal operation. A similar procedure was done for SiC with a temperature correlation of [W. Jiang et al., 2021]:

$$c_{p,SiC}\left(\frac{J}{kgK}\right) = 925.65 + 0.3772T_K - 7.9259 * 10^{-5}T_K^2 - \frac{3.1946 * 10^7}{T_K^2} \tag{5.17}$$

The averaged values of c_p for UCO and SiC are 316 $J/(kg\ K)$ and 1223 $J/(kg\ K)$ respectively. It is noteworthy that the c_p of UCO in this temperature range is similar to the nonstoichiometric UO_2 c_p value of 310 $\pm$ 10 $J/(kg\ K)$ [Kavazauri et al., 2016]. The c_p value of UCO also changes little in this temperature range, quite comparable to measurement error of 10 $J/(kg\ K)$. For SiC, we can see from table 5.6 that the values vary quite linearly with temperature in this range. Hence, using Simpson's $\frac{1}{3}$ rule should not result in gross inaccuracies. The c_p dependence on temperature for UCO and SiC are shown in Table 5.6:

Table 5.6: c_p for UCO and SiC $\frac{J}{kg\ K}$

Temperature (K)	c_p UCO	c_p SiC
873	311.3	1153
973	313.8	1184
1073	315.6	1211
1173	317.1	1236
1273	319	1258
1373	321.6	1277
1473	325.3	1295

For the carbon layers such as the buffer and pyrolytic carbon, a typical average value for the heat capacity is 720 $J/(kg\ K)$ [W. Jiang et al., 2021]. The volumetric heat capacity for a TRISO particle then then be a volume weighted average of the volumetric heat capacity of each layer denoted i:

$$\rho_{TRISO}c_{p,ave,TRISO} = \sum_i^N \frac{V_i}{V_{TRISO}}\rho_i c_{p,ave,i} \tag{5.18}$$

The volume fractions, densities and average heat capacities for each layer is shown in table 5.7:

Table 5.7: Volume Fraction, Densities and c_p used to calculate $\rho c_p \ \frac{J}{m^3 \ K}$

Layer within TRISO Particle	Outer Radius (cm)	Volume Fraction	$\rho \ (kg/m^3)$	$c_{p,ave} \ \frac{J}{kg \ K}$
UCO Fuel Kernel	2.15E-02	0.129	10500	316
Buffer	3.15E-02	0.278	1000	720
PyC1	3.50E-02	0.151	1900	720
SiC	3.85E-02	0.185	3200	1223
PyC2	4.25E-02	0.257	1870	720

The resultant average volumetric heat capacity we use is $1.905 * 10^6 \ \frac{J}{m^3 \ K}$. This can be used to calculate the average volumetric heat capacity of the pebble ($\rho c_{p,ave,pebble}$). Each pebble is modelled with a TRISO matrix and a pure graphite outer shell. The TRISO matrix layer has a radius of 1.9 cm and is covered by the graphite shell 0.1 cm thick. Overall, the pebble has a radius of 2 cm or diameter of 4cm. The formula used to calculate the average $\rho c_{p,ave,pebble}$ is:

$$\rho_{pebble} c_{p,ave,pebble} = \sum_{i}^{N} \frac{V_i}{V_{pebble}} \rho_i c_{p,ave,i} \tag{5.19}$$

The ρc_p for graphite matrix and the graphite outer shell is calculated using $\rho_{graphite} = 1199.5 \ \ kg/(m^3)$, and $c_p \ \ = 720 \frac{J}{kg \ K}$. The TRISO volume fraction is modelled to be 0.3 of the volume within the TRISO matrix.

Table 5.8: Volume Fraction, Densities and c_p used to calculate $\rho c_p \ \frac{J}{m^3 \ K}$

Material	Volume Fraction	$\rho c_{p,ave} \ \frac{J}{m^3 \ K}$
TRISO Particles	0.2572	1904790.8
Graphite within TRISO matrix	0.6002	863640
Graphite Shell	0.1426	863640

The resulting volumetric heat capacity of the pebble as a whole is $1.131 * 10^6 \frac{J}{m^3 \ k}$. This can be substituted into the conduction equation as a GeN-Foam input.

Parameters for Solid-Fluid Interface

In GeN-Foam, correlations for heat transfer and pressure drop across the pebble bed are used to calculate the subscale structure interaction terms for the momentum and energy equation in porous media. These correlations are in turn based on the Reynold's Number (Re) and Prandtl Number (Pr).

We can see from "FSPair.C" that Re is defined as:

$$\mathrm{Re} = \frac{\varepsilon u_{interstitial} d_h}{\nu} \tag{5.20}$$

Hence, for all correlations in GeN-Foam, we will need to define correlations in terms of Re as shown in equation 5.20. Here, d_h is hydraulic diameter, $u_{interstitial}$ is the interstitial velocity, ε is the porosity of the medium and ν is kinematic viscosity of the fluid. This is important to note because correlations such as the Wakao correlation and Ergun Correlation are based on particle diameter rather than hydraulic diameter [Wakao, Kaguei, and Funazkri, 1979; Ergun and Orning, 1949]. Hence, we will need to perform some conversion to obtain these equations in terms of d_h.

To sort this issue out, we can use the Carman-Kozeny theory to correlate hydraulic diameter to pebble diameter [Pilehvar et al., 2013]:

$$d_h = \frac{4V_{void}}{A_{pebble}} \tag{5.21}$$

Where V_{void} is the volume of the voids or fluids within the pebble bed and A_{pebble} is the total surface area of the pebbles within the pebble bed. It is sometimes easier to write equation 5.21 in terms of V_{pebble}, the total volume of the pebbles:

$$d_h = \frac{4V_{pebble}}{A_{pebble}}\varepsilon \tag{5.22}$$

The surface to volume ratio for spherical pebbles is simply $\frac{3}{r}$ where r is the radius of the pebble. We use a solid packing fraction of 0.61 to get a porosity of 0.39. Hence, we arrive at d_h in terms of pebble diameter d_p:

$$d_h = \frac{2d_p}{3} * 0.39 \tag{5.23}$$

We can obtain Ergun's equation for spheres in terms of d_h:

$$-\frac{\Delta P}{L} = \frac{66.6667 q\mu}{d_h^2}\frac{(1-\varepsilon)^2}{\varepsilon} + \frac{1.1667\rho q^2}{d_h}\frac{1-\varepsilon}{\varepsilon^2} \tag{5.24}$$

ρ is fluid density, and q is Darcy velocity or superficial velocity:

$$q = \varepsilon u_{interstitial} \tag{5.25}$$

We can write Re in terms of q:

$$\mathrm{Re}_{q,d_h} = \frac{q d_h}{\nu} \tag{5.26}$$

GeN-Foam requires input to be in the form of the drag coefficient f_D:

$$f_D = \frac{\Delta P}{\frac{1}{2}\rho u^2_{interstitial}} \frac{d_h}{L} \tag{5.27}$$

Substitution of these equations together, and using ε of 0.39, we can show that:

$$f_D = \frac{19.3492}{Re_{q,d_h}} + 1.4234 \tag{5.28}$$

For clarity, I denote Re as Re_{q,d_h} so that we know the Reynold's number is based on Darcy velocity q and hydraulic diameter d_h. Similarly, the Wakao correlation has to be transformed in a similar manner:

$$\mathrm{Nu}_{d_h} = 0.52 + 0.64177\mathrm{Pr}^{1/3}\mathrm{Re}^{0.6}_{q,d_h} \tag{5.29}$$

Of course, Nu_{d_h} only gives us the heat transfer coefficient h to calculate a heat flux between pebble surface and fluid. A surface area input is still needed for heat transfer calculation. GeN-Foam calls this input the "volumetricArea", which is surface area per unit volume of the reactor (including coolant and pebbles).

$$\mathrm{volumetricArea} = \frac{4\pi r^2}{\frac{4}{3}\pi r^3}(1-\varepsilon) = \frac{3}{r}(1-\varepsilon) \tag{5.30}$$

For pebble packing fraction of 0.61 and pebble radius r of 0.02 m, the volumetric area is 91.5 m^{-1}.

GeN-Foam also requires a "powerDensity" input for its input file. Normally, this input is neglected when neutronics solvers are switched on. While we may be tempted to leave this entry at some random value, the powerDensity term is important in stabilising the GeN-Foam solver. GeN-Foam tends to be quite unstable during case startup where neutronics and thermal hydraulics solvers are switched on. The procedure to work around this is to switch on these solvers sequentially. That is, we switch on thermal hydraulics solvers first and assume some constant power is supplied to the reactor. This term is the powerDensity. This is the total pebble power per unit volume of the whole pebble bed (pebble and fluid). It is calculated using the reactor's target operating power of 312.5 kWth. The final input of powerDensity is 246513 W/m^3.

Fluid Thermal Hydraulics Inputs

Now, for the fluid, we deal with single phase FLiBe where radiation heat transfer is ignored for simplicity. We have also seen that this is probably a safe assumption within the pebble bed [I. M. B. Johnson, 2022].

FLiBe Thermophysical Properties FLiBe thermophysical properties for GeN-Foam used are [Sohal et al., 2010]:

$$\rho(kg/m^3) = 2415.6 - 0.49072(T(K)) \; ; \; T(K) = [732.2, 4498.8] \tag{5.31}$$

$$\mu(Pa \cdot s) = 0.000116 \exp\left(\frac{3755}{T(K)}\right) \; ; \; T(K) = [873, 1073] \tag{5.32}$$

$$c_p(J/(kg \cdot K)) = 2415.8 \; ; \; T(K) = 973 \; K \tag{5.33}$$

$$k(W/(m \cdot K)) = 0.629697 + 0.0005 \cdot (T(K)) \; ; \; T(K) = [500, 650] \tag{5.34}$$

On Thermal Conductivity As mentioned in the preceeding chapters, it seems that the thermal conductivity correlations [Sohal et al., 2010] were not sufficient as they do not cover the temperature range of operation. Nevertheless, a measured value for FLiBe thermal conductivity at 1 $W/(m \cdot K)$ was given for 873K [Sohal et al., 2010]. For 973K, a thermal conductivity of 1.1 $W/(m \cdot K) \pm 10\%$ [Romatoski and L.-W. Hu, 2017]. Given a 10% error bar, the correlations provided [Sohal et al., 2010] do are indeed applicable in the operating temperature of the FHR.Figure 5.17 plots some existing FLiBe thermal conductivity data [Romatoski and L.-W. Hu, 2017] against given correlations [Sohal et al., 2010] with 10% error bars:

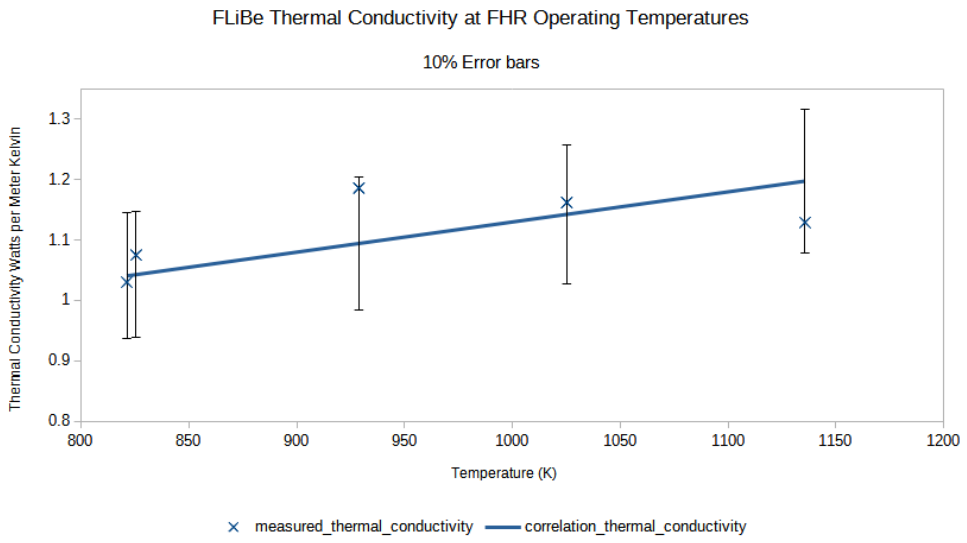


Figure 5.17: FLiBe Thermal Conductivity Correlation and Measured Data

This shows that the provided correlations are indeed within experimental error up to approximately 1130 K. The data for plotting Figure 5.17 is in Table 5.9:

Table 5.9: Experimental Thermal Conductivity compared with Correlation

Temperature (K)	Experimental Thermal Conductivity $(W/(m \cdot k))$ [Romatoski and L.-W. Hu, 2017]	Correlation $(W/(m \cdot k))$ [Sohal et al., 2010]
821	1.03	1.04
826	1.075	1.042
929	1.186	1.094
1025	1.162	1.142
1136	1.129	1.198

On Dynamic Viscosity FLiBe's dynamic viscosity is described by Cantor's Correlation [Sohal et al., 2010; Romatoski and L.-W. Hu, 2017] with error bars of about ±20% [Romatoski and L.-W. Hu, 2017]. Unfortunately, its range of validity is from 873K to 1073K. A similar equation, by Gierszewski [Romatoski and L.-W. Hu, 2017] has a wider temperature range and is only slightly different:

$$\mu(Pa \cdot s) = 0.000116 \exp\left(\frac{3760}{T(K)}\right) \; ; \; T(K) = [600, 1200] \tag{5.35}$$

Gierszewski's correlation would sufficiently cover the range of temperatures during reactor operation for FLiBe. However, another technical issue arises due to GeN-Foam's input format. Here, if we wish to use polynomial correlations to describe thermal conductivity for FLiBe, we must also use polynomials to describe all other thermophysical properties. Due to this limitation in input, we had to fit a polynomial to Gierszewski's correlation. This was done in the range for $[700K, 1200K]$ where transients of interest are expected to occur. The polynomial is written as:

$$\begin{aligned} \mu(Pa \cdot s) = 1.0651 - 3.9767 * 10^{-3} T(K) + 5.6614 * 10^{-6} T(K)^2 \\ -3.6142 * 10^{-9} T(K)^3 + 8.6943 * 10^{-13} T(K)^4 \; ; \; T(K) = [600, 1200] \end{aligned} \tag{5.36}$$

To verify that the polynomial works for our desired temperature range, we plot Cantor's Correlation, Gierszewski's correlation and this polynomial in the range $[700K, 1270K]$:

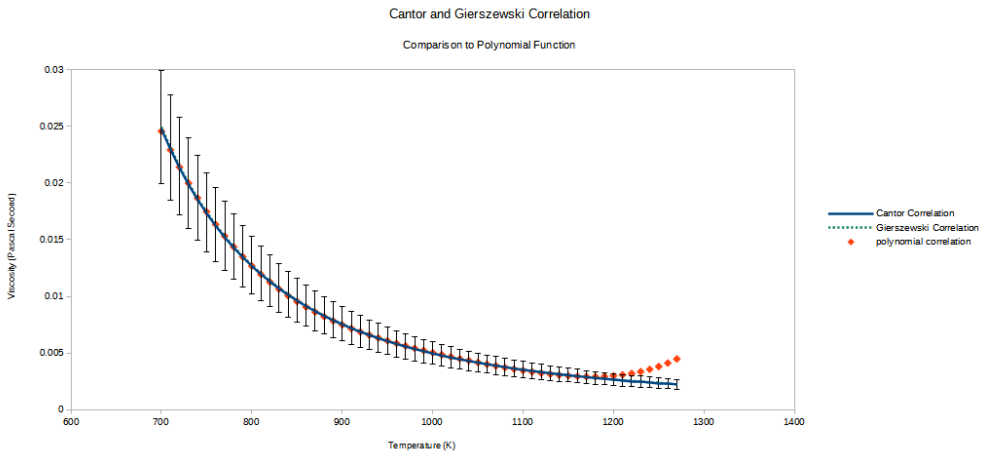


Figure 5.18: FLiBe Viscosity Correlation Comparisons

Figure 5.18 shows that the polynomial satisfies both Cantor and Gierszewski's correlation within 20% error bar for the temperature range $[700K, 1200K]$.

Molar Weight, a Potentially Redundant Input for Single Phase Pure Mixture of FLiBe For thermophysical property input syntax, we need to specify the model "icoPolynomial" in the thermophysical property library to indicate that polynomials based on temperature are used as inputs for thermophysical properties of an incompressible fluid. Unfortunately, the "icoPolynomial" input syntax also requires a "molWeight" entry which refers to molecular weight. 99.0377003 $g/gmol$ was used as the input molecular weight for FLiBe. This molecular weight was not done rigourously because the readme portion in IcoPolynomial.H found in OpenFOAM's source code seems to indicate that the code uses a molar basis internally for its calculation. Hence, molar weight is used to convert a mass basis inputs to a molar basis, and then the outputs are likely returned in mass basis form using the same "molWeight". Hence, our choice of molar weights may not matter too much.

Boundary Conditions and Initial Conditions We also set initial and boundary conditions for Temperature (T) fields, velocity (U) fields, pressure (p) fields and pressure less the hydrostatic component (p_rgh) fields.

The temperature fields are set to zeroGradient at all boundaries except for the entrance, this is essentially an adiabatic condition for simplicity and expedited initial model development. The zeroGradient boundary condition is a Neumann boundary condition where the gradient or derivative of the field is set to zero. T is set to a specific uniform inlet value (uniformFixedValue) based on a custom csv file so that either a custom uniform temperature or a PRBS signal can be given as a forcing function. The initial conditions are 873K uniformly through the mesh.

U is set such that the mass flowrate of the inlet is 1.2934 kg/s. This would ensure that at 312.5 kWTh, the temperature change is about 100K. The outlet boundary condition

is inletOutlet, essentially a modified zeroGradient boundary condition meant for improved stability of the numerical solver.

The p boundary conditions depend on p_rgh. Dirichlet (fixedValue) boundary conditions are set for the exit of approximately 1 atmosphere, thus indicating that the reactor should operate at atmospheric pressure if hydrostatic pressure is subtracted.

Simplifications for Fluid Models For this work, turbulence is ignored because the flow was slow enough. Furthermore, in the pebble bed, it is safe to assume that interaction with the subscale structure would dominate turbulence effects [Fiorina, I. Clifford, et al., 2015]. Outside the pebble bed, flow was slow enough that Re did not exceed the transition regime for turbulence.

Simplifications for Reflector Structure Models For this work, thermal inertia of the reflector structure is ignored as the porous media models within GeN-Foam are not yet well suited for a solid reflector block with conduction heat transfer. Hence, for this iteration, the reflector is essentially simulated at a constant temperature. The source code can be rewritten in future to include the reflector structure for conjugate heat transfer plus moderator temperature feedback in future iterations.

Fuel Region Input Example

Based on these inputs, one can program in properties for each region of the reactor. Only the fuel region inputs are supplied here as an example as it is most complex:

```
"fuel_cell"
{

    volumeFraction      0.61;
    Dh                  0.0104;

    powerModel
    {
        type                lumpedNuclearStructure;
        // total pebble surface area per unit
        // volume of reactor (includes coolant and pebbles)
        // (m^2/m^3) units
        volumetricArea      92;
        // Power density smeared over the entire
        // structure (W/m^3)
        // this is total pebble power divide
        // by total pebble volume
        powerDensity        2.46513e5;
```

```
// Number of nodes
nodesNumber         1;
// Define which node temperature is used
// to paramtrize XS according to
// nuclearDataFuelTemp
nodeFuel            0;
// Define which node temperature is used
// to paramtrize XS according to
// nuclearDataCladExp. Of course, it does
// not have to represent a cladding. It
// could be used to paramtrize over the
// graphite temperature in PBRs
// I'm not calculating expansion, so just leave
// it as zero
nodeClad            0;
// Heat conductances from
// average to max,
// and average to surface.
// Note that heat conductances (W/K)
// must be divided by the volume
// of the entire structure.
// This is thermal conductance divide by
// volume of ONE pebble
//
// There must be one more conductance
// than the number of nodes
// (see explanation in class)
heatConductances    (15548000 15548);
// volumetric heat capacity of
// each node. This is J/(m^3 K)
// for each node
rhoCp               (1131437);
// Frction of the volume of the structure
// occupied by each node
volumeFractions     (1);
// Fraction of total power in the strucure
// that goes to each node
powerFractions      (1);
// Initial temperature, if no temps found in
// the time folder
// kelvin unit
T0                  1173;
```

```
    }
}
```

Heat Transfer Parameters Explained Here, an initial temperature entry for the fuel (T0) is required and put at 1173 K. Since this is a one node structure, only one volumetric heat capacity (rhoCp) is supplied. Two heat conductances are supplied, a heat conductance to some inner maximum temperature $T_{\max}$ and a heat conductance to the pebble surface. This $T_{\max}$ is connected to $T[0]$ via a thermal conductance. It can be used to represent the maximum temperature of the fuel.

In lumpedNuclearStructure.H in GeN-Foam, we see a diagram in the readme for a model with n nodes:

```
Tmax           T[0]           T[1]          T[n-1]         Tsurface
| --- H[0] --- | --- H[1] --- | --- H[2] --- |  --- H[n] --- |
```

Here, n represents the number of heat conductances the user must supply. Of course, in this case, n = 3. $T_{\max}$ is used by GeN-Foam to calculate the maximum fuel temperature in that cell in case that becomes important. For TRISO fuel however, we are not interested in this because the threshold for fuel failure is almost never reached during reactor transients of interest. In fact, for the purposes of reactor fuel temperature feedback, the user can only select the representative fuel temperature from the various nodes inside the model from $T[0]$, $T[1]$, all the way to $T[n-1]$. $T_{\max}$ and $T_{surface}$ are not even selectable. Hence, $T_{\max}$ is quite irrelevant except for printing out the maximum fuel temperature. This can be verified by looking in GeN-Foam's source code in lumpedNuclearStructure.C.

For this reason, I gave $H[0]$ an arbitrary value which was 1000 times the conductance value between $T[0]$ and $T_{surface}$. This is because I wanted $T_{\max}$ to be the same value as $T[0]$ as far as possible. Therefore, I gave it an arbitrarily large conductance value. Again, the subscale fuel model is a grossly simplified version of a potentially doubly heterogeneous heat transfer model. Fidelity improvements can be made in future work, but not in this iteration as fidelity in terms of the heterogeneous heat transfer media is not the goal in this dissertation.

Temperature used for Fuel Temperature Feedback We use one node for this fuel because it is the most simple model available. The nodeFuel "0" entry here indicates that I use $T[0]$ as the relevant temperature for fuel temperature feedback. The nodeClad entry is functionally similar except that it pertains to cladding expansion. Since cladding expansion is irrelevant in this scenario, the default value of "0" is also given.

Subscale Heat Sink and Momentum Sink Now, the Wakao Correlation and Ergun Correlation are also put in here:

```
"fuel_cell:reflector_cell"
{
    type        byRegime;
    regimeMap   "lamTurb";

    //- List of subdicts specifying a heatTransferModel for each regime
    //  in the lamTurb regimeMap
    "laminar"
    {
        // Nu = const + coeff * ReˆexpRe * PrˆexpPr
        type    NusseltReynoldsPrandtlPower;
        const   0.52;
        coeff   0.64177;
        expRe   0.6;
        expPr   0.333333333333;
    }
    "turbulent"
    {
        type    NusseltReynoldsPrandtlPower;
        const   0.52;
        coeff   0.64177;
        expRe   0.6;
        expPr   0.333333333333;
    }

"fuel_cell"
{
    // this is based on Ergun Correlation
    // for workings and derivation, please see readme
    // f_d = 19.3492 / Re + 1.4234
    // Again, we are not really interested in
    // drag for now, but it is good to see
    // how the terms are derived through the readme.
    type    ReynoldsPower;
    coeff   19.3492;
    exp     -1.00;
    const      1.4234;
}
```

On the Reflector and FLiBe Regions

For regions other than the fuel region, we know physically that porous media does not apply. The reflector region is meant to be 100% solid graphite and the FLiBe regions were meant to be 100% fluid.

For FLiBe, it was easy to set porosity to 100% so that the equations (almost) resembled pure Navier-Stokes with heat transfer. We mentioned before that radial conduction was neglected, and this could cause some inaccuracies. However, since advection was still modelled, and this was still somewhat okay as we could model at least one form of heat transport.

For the graphite reflector region, the porosity cannot be set to zero so that the porous media does not contain fluid. These porous media equations are designed to have fluid, so there was little choice in what could be done with GeN-Foam in its present (June 2023 and prior) state. The graphite reflector was then modelled to have a 1% porosity, which was close-ish to 99%. This was done chiefly because the porous media model cannot accept a 0% porosity input in its present state. Of course, in a real FHR, the graphite reflector is not usually a monolithic block, but instead is made of smaller constituent blocks. There would be gaps between these blocks which would be filled with FLiBe, and this would perhaps be a physical means of justifying the 1% porosity. However, the porous media model in GeN-Foam fails to account for thermal conduction between the solid phase in adjacent control volumes. A redesign of source code is still necessary to model conjugate heat transfer accurately. For now, we do our best to approximate a solid medium, and this means setting the momentum sink terms to very high values within the graphite block. This would ensure that the coolant mostly flows through the FLiBe entrance and exit regions as well as the core and not through the graphite block. In terms of heat transfer, the reflector structure cannot participate in it because heat does not flow between adjacent solid regions of the cells by default.

Therefore, the graphite reflector effectively becomes a reflector structure perfectly thermally insulated from the reactor. This inaccuracy was tolerated as I would otherwise have to re-write, compile and test the GeN-Foam source code. This was out of scope for this work. A second consideration I thought of, but didn't work, would be to make the graphite region a fluid, but replace it with a "fluid" of extremely high viscosity so much so that it acts like a solid similar to glass or tar. The medium would then conduct heat away from the reactor (or to it) and act as a body with thermal inertia. Unfortunately, we found that the GeN-Foam equations do not normally include fluid conduction for its porous media. Hence, this approach is also currently unsuitable, but can be explored in future work.

Neutronics Boundary Conditions

Vacuum Boundary Condition Syntax in GeN-Foam For neutronics, vacuum boundary conditions were placed at all boundaries for simplicity. We adapt a boundary condition from SP3 equations [Fiorina, Hursin, and Pautz, 2017] known as the "albedoSP3" boundary conditions put under a "defaultFlux" field. The source code can be found in "albedoSP3FvPatchField.C" and "albedoSP3FvPatchField.H".

The equations used for this boundary condition are as follows [Fiorina, Hursin, and Pautz, 2017]:

$$D_i \nabla \hat{\phi}_{0,i} = -\frac{1}{2}\left[\frac{1-\alpha_i}{1+\alpha_i}\right]\left(\hat{\phi}_{0,i} - \frac{3}{4}\phi_{2,i}\right) \tag{5.37}$$

$$\frac{3}{7}\frac{1}{\Sigma_{t,i}}\nabla \phi_{2,i} = \frac{1}{2}\left[\frac{1-\alpha_i}{1+\alpha_i}\right]\left(\frac{3}{20}\hat{\phi}_{0,i} - \frac{21}{20}\phi_{2,i}\right) \tag{5.38}$$

$\phi_{2,i}$ is a second moment flux for energy group i and it pertains to the SP3 equations. For diffusion neutronics, $\phi_{2,i}$ can be essentially set to zero to reduce the SP3 boundary conditions to a diffusion neutronics boundary condition. This would simplify the term $\hat{\phi}_{0,i} = 2\phi_{2,i} + \phi_{0,i}$ to $\hat{\phi}_{0,i} = \phi_{0,i}$. The resultant albedoSP3 boundary condition equation becomes:

$$D_i \nabla \phi_{0,i} = -\frac{1}{2}\left[\frac{1-\alpha_i}{1+\alpha_i}\right]\phi_{0,i} \tag{5.39}$$

Where α_i is the ratio between incoming and outgoing neutron partial currents for energy group i [Fiorina, Hursin, and Pautz, 2017]:

$$\alpha_i = \frac{J_i^-}{J_i^+} \tag{5.40}$$

Verification of Adapting SP3 Boundary Condition for Diffusion Vacuum Boundary Condition We can verify that the albedoSP3 boundary conditions can indeed be adapted for diffusion by setting $\hat{\phi}_{0,i} = \phi_{0,i}$ and $\phi_{2,i} = 0$ by deriving the flux boundary conditions using partial currents. This limited derivation is for a 1D slab geometry. We can start by expressing angular flux assuming it is linearly anisotropic [Duderstadt and Hamilton, 1976]:

$$\psi(x,\mu) = \frac{1}{2}[\phi_0(x) + 3\mu J(x)] \tag{5.41}$$

From diffusion theory, we further approximate neutron current as follows:

$$J(x) = -D\frac{\partial \phi_0(x)}{\partial x} \tag{5.42}$$

From diffusion neutronics, we can express angular flux in terms of the zeroth moment flux, scattering angle μ, Diffusion Coefficient D and flux gradient:

$$\psi(x,\mu) = \frac{1}{2}\left[\phi(x) - 3\mu D\frac{\partial \phi(x)}{\partial x}\right] \tag{5.43}$$

Positive and negative partial currents can be found by integration:

$$J^+(x) = \int_0^1 d\mu\mu\psi(x,\mu) = \int_0^1 d\mu\mu \left(\frac{1}{2}\phi(x) - 3\mu D\frac{\partial\phi(x)}{\partial x}\right) = \frac{\phi(x)}{4} - \frac{D}{2}\frac{d\phi(x)}{dx} \tag{5.44}$$

$$J^-(x) = \int_0^1 d\mu\mu\psi(x,\mu) = \frac{\phi(x)}{4} + \frac{D}{2}\frac{d\phi(x)}{dx} \tag{5.45}$$

We can then substitute these expressions of partial currents into the definition of α_i:

$$\alpha_i = \frac{\frac{\phi(x)}{4} + \frac{D}{2}\frac{d\phi(x)}{dx}}{\frac{\phi(x)}{4} - \frac{D}{2}\frac{d\phi(x)}{dx}} \tag{5.46}$$

Using an ad-hoc extension of the 1D flux derivative to 3D, and by making $D_i\nabla\phi_{0,i}$ the subject of Equation 5.46, we obtain Equation 5.47:

$$D_i\nabla\phi_{0,i} = -\frac{1}{2}\left[\frac{1-\alpha_i}{1+\alpha_i}\right](\phi_{0,i}) \tag{5.47}$$

Equation 5.47 is indeed the SP3 boundary conditions adapted for diffusion by setting $\hat{\phi}_{0,i} = \phi_{0,i}$ and $\phi_{2,i} = 0$ in equation 5.39.

albedoSP3 γ input for Vacuum Boundary Condition Now that we have verified this, we need to specify parameters for the albedoSP3 boundary condition. The most important parameter for this boundary condition is γ where:

$$\gamma = \frac{1}{2}\frac{1-\alpha}{1+\alpha} \tag{5.48}$$

For a vacuum boundary condition, incoming flux J^- is zero, so α_i is zero for all energy groups. Hence $\gamma = 0.5$.

Post Processing

Use of the Boussinesq Approximation The outlet temperature was measured using OpenFOAM's postProcessing. This is put into GeN-Foam's "controlDict" input file. We can obtain a mass flow weighted average of the temperature measured at the outlet. We are given options to perform a weighted area average of the temperature field by the fluid flux, also known as "phi" in the OpenFOAM solvers. Unfortunately, OpenFOAM may use "phi" to represent mass flux or volume flux depending on the solver.

In GeN-Foam's context, phi does look like it represents volume flux rather than mass flux. However, it may not matter greatly as FLiBe is a liquid. Liquids do not change much in density so their changes in density can be neglected except in calculating buoyancy forces. I opted to use this assumption (Boussinesq approximation) and used the volume flowrate weighted temperature to approximate the mass flowrate weighted temperature. This was

purely out of convenience because accessing mass flux weighted temperature in the post processing tools was more difficult in terms of syntax. I chose volume averaged flux because it saved some time.

To show that mass flux averaged temperature and volume flux averaged temperature is more or less equal, we can begin with some equations for mass flowrate averaged temperature:

$$T_{bulk} = \frac{\iint dA\ u\rho c_p T}{\iint dA\ u\rho c_p} \tag{5.49}$$

For FLiBe, heat capacity is essentially constant based on the molten salt databases [Sohal et al., 2010]. Hence, it can be divided out:

$$T_{bulk} = \frac{\iint dA\ u\rho T}{\iint dA\ u\rho} \tag{5.50}$$

Now of course, we can assume fluid density doesn't change much, this is essentially the Boussinesq approximation. Now, we are left with:

$$T_{bulk} = \frac{\iint dA\ uT}{\iint dA\ u} \tag{5.51}$$

Given this assumption, we can infer that volumetric flow weighted average temperature is equal to mass flow weighted average temperature. Of course, we can estimate the degree to which density affects the discrepancy between volumetric flow rate averaged temperature and mass flowrate averaged temperature.

$$T_{bulk\ mass\ average} - T_{bulk\ volume\ average} = \frac{\iint dA\ u\rho c_p T}{\iint dA\ u\rho c_p} - \frac{\iint dA\ uT}{\iint dA\ u} \tag{5.52}$$

Let ρ be represented as a sum between a reference density and a density deviation:

$$\rho(T) = \rho_{ref} + \Delta\rho(T) \tag{5.53}$$

If the density deviation is small enough, we should be able to safely say that volume weighted temperature averages are same as mass weighted temperature averages. One way to quantify if a density change is small enough is to compare it to experimental measurement uncertainty and data variability. For FLiBe, an estimate of this variability is about 2% to a maximum of 11% [Vidrio et al., 2022]. We can use equation 5.54 to estimate $\Delta\rho(T)$:

$$\rho(kg/m^3) = 2415.6 - 0.49072(T(K)) \tag{5.54}$$

To estimate an upper bound for temperature variation for outlet fluid temperatures assuming it is somewhat well mixed, we can use the temperature difference between inlet and outlet temperature of the Mark I FHR, which is about $100°C$:

$$\Delta\rho(T) = -0.49072 * 100 \approx 49.1 kg/m^3 \tag{5.55}$$

Now, the variation in experimental data for FLiBe density being around 2.2% to 11% translates to about 42 kg/m^3 to 213 kg/m^3. This is at a reference ρ of 1938kg/m^3 at 973 K. This temperature variation is within this variability and therefore the temperature discrepancy between using volume weighted and mass weighted temperature averages is quite indistinguishable from variations between mass weighted temperature averages due to variations in FLiBe density measurements.

Hence, for this case, whether ϕ represents volumetric flux or mass flux, one need not worry too much about the discrepancy in temperature measurement. A second issue with ambiguous naming convention phi is that is called "flux" in the documentation. Volume flux can sometimes be in units of m^3/s [Ma et al., 2023] or be considered volumetric flowrate per unit area m/s. This naming convention can be a source of confusion for users of OpenFOAM and GeN-Foam. Therefore, it is important to explore the source code to ascertain what "flux" actually means in each context. In the case of GeN-Foam, it is likely to be volumetric flowrate through each face in the mesh. This is shown from some source code exploration in the Appendix on page 457. With these concerns out of the way, we can find the average temperature using volume flux (phi) as a weight field.

Simplification: Natural Convection Switched off For low flowrates, however, natural convection induced local flow tends to produce flows with local velocities comparable to the superficial velocity of fluid through the reactor. This tends to cause the ϕ averaged temperature generated by the post processor to become inaccurate and unphysical. For example, we get negative temperatures in kelvin due to backflow or some other flow patterns. I could of course debug the post processing tool to ensure that it produces the right averaged temperature values. However, I decided to switch off the natural convection phenomena in this case as I was not interested in studying mixed convection. This may reduce heat transfer coefficient as we are having aiding mixed convection, but it will not impact the main goals of the work. This goal is to demonstrate the methodology of "training" a surrogate model to replicate transient behaviour of the multiphysics reactor.

Input functionObject Example The following syntax was used to obtain volumetric flowrate averaged temperature. These inputs are placed in "controlDict" under a "functions" section:

```
outletTempAvg_byFlux
{
    type            surfaceFieldValue;
    libs            ( fieldFunctionObjects );
    writeControl    adjustableRunTime;
    log             true;
    writeFields     false;
    regionType      patch;
    name            outlet;
```

```
    operation       weightedAverage;
    weightField     phi;
    region          fluidRegion;
    fields          ( T );
    executeControl  adjustableRunTime;
    executeInterval 0.5;
    writeInterval   0.5;
}
```

Stabilisation Procedures

Getting a coupled neutronics and thermal hydraulics multiphysics simulation presents some practical challenges in terms of solver stability and ensuring that $k_{eff} = 1$. If one were to simply start an eigenvalue simulation or transient simulation with all equations solved simultaneously, solver instability becomes almost certain. In this section, we discuss some important procedures used to ensure that the GeN-Foam solver can run in a stable manner.

Understanding the GeN-Foam Solver

GeN-Foam uses the multigroup diffusion equation with delayed neutrons accounted for [Fiorina, Kerkar, et al., 2016] to solve its diffusion equation:

$$\begin{aligned}\frac{1}{v_g}\frac{\partial}{\partial t}\phi_g(\vec{r},t) = \frac{\chi_g(1-\beta_{total})\sum_{g'=1}^{G}\nu_{g'}\Sigma_{f,g'}\phi_{g'}(\vec{r},t)}{k_{eff}} + \sum_{k=1}^{6}\lambda_k C_k \\ + \sum_{g'=1}^{G}\phi_{g'}(\vec{r},t)\Sigma_{s,g'\to g} + Q_{ex,g}(\vec{r},t) \\ +\nabla \bullet D_g \nabla\phi_g(\vec{r},t) - \Sigma_{t,g}\phi_g(\vec{r},t)\end{aligned} \tag{5.56}$$

Where β_{total} is the total delayed fraction. λ_k represents decay constant for precursor group k and C_k represents concentration of delayed precursors for group k in m^{-3} [Fiorina, Kerkar, et al., 2016]. For solid fuel, the delayed precursors are governed by balance equations for each group:

$$\frac{\partial C_k}{\partial t} = \frac{\beta_k \sum_{g'=1}^{G}\nu_{g'}\Sigma_{f,g'}\phi_{g'}(\vec{r},t)}{k_{eff}} - \lambda_k C_k \tag{5.57}$$

The diffusion solver can be used for eigenvalue calculations, and here, all time dependent variables in equation 5.57 and equation 5.56 are set to zero, and the solver iteratively finds k_{eff}. In transient (non-eigenvalue) mode, k_{eff} is specified by the user manually, while the time dependent terms are non zero [Fiorina, Kerkar, et al., 2016]. For a critical reactor $k_{eff} = 1$. However, even if the Monte Carlo simulation has a k_{eff} of 1 and we replicate the

geometry and cross sections in GeN-Foam, k_{eff} may not be exactly 1 but a value close to 1 possibly because of approximation errors within GeN-Foam.

Hence, to get the reactor ready for transient simulation, we must ensure the reactor is 1 even if the eigenvalue calculations do not yield $k_{eff} = 1$. To do so, we first run an eigenvalue simulation at the steady state conditions of the reactor so as to obtain a k_{eff}, and then substitute this k_{eff} into the transient simulator to compensate for whatever errors or discrepancies may arise from the multiphysics diffusion and Monte Carlo simulation.

To do so, we ideally want to run GeN-Foam in eigenvalue mode so that the k_{eff} can be obtained by the diffusion solver at steady state. I found that when I tried solving the thermal hydraulics and neutronics equations simultaneously, the solver ran into instabilities.

Solver Multiphysics Stabilisation

To solve the solver instabilities during eigenvalue calculations, I ensured that neutronics was not solved within the simulation first. GeN-Foam has entries within its "controlDict" input to allow the user to switch neutronics, energy, fluids and thermal mechanics solvers on and off. As mentioned before, thermal mechanics solvers are switched off always. However, I switched on the energy and fluid mechanics solvers first and switched neutronics solvers off. The reactor would then output a fixed power output of 312.5 kW. This would allow us to have a steady state temperature field to work with.

The second step I took was to switch on the neutronics solver with eigenvalue mode on. However, I switched the energy solver off. Hence, only the neutronics and fluid solvers were working. These neutronics and fluid mechanics were not as interdependent as heat transfer and neutronics, and therefore, the simulation was quite stable. Turning off the energy solver meant also that the temperature distribution in the core was preserved. This may not be representative of a real core since power distribution is uniform. In a real reactor, neutron flux and therefore power generation near the center of the core is higher. However, a uniform power distribution also allows for maximum temperature to be at the center of the core. In terms of temperature profile, it is a close enough approximation to a realistic power distribution. This approximation is quite important for solver stability because the solver might calculate large changes in the system heat transfer or neutronics. These large changes simulated tend to destabilise the solver.

At the end of these two steps, we would have had a k_{eff} estimation, a starting temperature distribution, as well as a neutron flux distribution. With these, we could switch neutronics, energy and fluid mechanics solver on, and toggle off the eigenvalue mode for GeN-Foam. Thereafter, the GeN-Foam solver could be run in transient (non-eigenvalue) mode. Before the model was subject to further perturbation, GeN-Foam was allowed to run the model at steady state for an extended time with all the physics turned on.

After this period of stabilisation, we can say that the reactor has somehow reached steady state where we have a steady temperature distribution and a steady neutron flux distribution. At this state, multiple copies of the reactor simulation were made so that the

computer could run frequency response tests and step response tests starting from the exact same initial steady state condition.

I used a bash script to automate this process because it was quite tedious to remember what steps needed to be taken. An excerpt of this script is presented in the Appendix on page 449 for the reader's reference.

Mesh Stabilisation

Another stability issue I encountered was due partly to the tetrahedral meshes and finite volume discretisation schemes. In OpenFOAM and GeN-Foam, partial differential equations (PDEs) are converted into matrix equations by discretisation. For example, we discretise the thermal hydraulics equations by splitting the entire volume into discrete finite volumes. We make approximations of the divergence, gradient and laplacian terms within each equation so that we get a system of matrix equations.

For GeN-Foam and OpenFOAM, these have to be manually specified by the user. For this arbitrary reactor, a tetrahedral mesh using Salome's NetGEN 1D-2D-3D algorithm was used. Nevertheless, some discretisation defaults used in previous simulations were not suitable as they were meant for more structured meshes. For tetrahedral meshes, Jasak recommends a least squares gradient scheme, upwinding for convection (divergence) schemes, and non-orthogonality limiters for diffusion (laplacian) schemes [Jasak, 2015]. An example of fvSchemes for the thermal hydraulics solver is provided for the reader's reference in the Appendix on page 450.

5.3 Obtaining Transfer Function

Preliminaries

Now that we have a GeN-Foam simulation set up, we need to obtain data for a data driven surrogate model. For surrogate modelling, we wish to represent reactor transient behaviour with the use of transfer functions. Ideally we would have a multiple input and multiple output (MIMO) system of equations. However, for demonstration purposes, a single input single output (SISO) system would suffice.

For a practical frequency response test in an experimental setup, the amplitude should achieve [De Wet and Per F Peterson, 2020]:

- 1. the desired signal to noise ratio
- 2. be low enough not to interfere with designed purpose of the facilities
- 3. be low enough to avoid perturbing nonlinearities
- 4. correspond to intervals that can be perturbed by chosen method

- 5. be within operating and safety limits

Some of these pertain only to an experimental setup, where frequency response was meant to be run during plant operations. The perturbations were not meant to disrupt the operations and safety of the plant.

We generally wish to achieve high signal to noise ratio without perturbing the nonlinearities for frequency response. This would, of course, mean that a simple transfer function is not able to replicate nonlinearities within the simulated reactor. However, nonlinearities can be explored in future work, and for a first iteration of a simulated neutronics feedback controller a transfer function model would suffice.

Even with this simplification, simulating frequency response tests to probe GeN-Foam models for a suitable transfer function takes about 1 to 2 weeks to complete on a regular PC.Hence, using single frequency forcing functions would be quite impractical. To reduce computation time, broadband signals are used instead to perturb the system. A transfer function was used to fit the frequency response data, and this transfer function was validated using a time domain step response test.

Pseudorandom Binary Sequence (PRBS)

The PRBS sequences were used because they could save time by perturbing multiple frequencies per test. I used MATLAB's SerDes package which contains a PRBS generating function. This function as used to generate a 128 bit PRBS signal. I then converted this signal to a inlet temperature PRBS signal of 30s per bit. This was sufficient to capture the transient characteristics for the reactor simulation. With a high frequency PRBS signal, I could not get enough signal energy to excite the lower frequency perturbations. Hence, I wanted to increase the number of seconds per bit. CIET was tested using PRBS of 10s per bit [De Wet and Per F Peterson, 2020], but I found that 30s per bit seemed to work for generating desirable input signal to noise ratios at lower frequencies. The PRBS sequence can be shown as follows in table 5.10:

Table 5.10: PRBS 128 Bit (16 Byte) Sequence

Byte Number	PRBS Sequence
1	00000100
2	00011000
3	01010001
4	11100100
5	01011001
6	11010100
7	11111010
8	00011100
9	01001001
10	10110101
11	10111101
12	10001101
13	00101110
14	11100110
15	01010101
16	11111100

The PRBS sequence is presented as 16 bytes, where 1 byte is 8 bits. Given that there are 30 seconds per bit, each PRBS sequence takes about 3810 seconds to complete. The full PRBS sequence with timestamps is shown in the Appendix on page 452.

Bode Plots and Transfer Functions

Using these PRBS signals, I then performed frequency response tests using the inlet temperature as a forcing function to probe how the reactor might respond to increased coolant temperature. The amplitudes chosen were 10K, 30K and 100K respectively.

After testing and post processing, the bulk temperature time domain data at the inlet and outlet were both subject to fast fourier transforms using a Matlab script. The resulting peaks were filtered using Matlab's peakFinder function from the Signal Processing Toolbox to select the most prominent signal energies. This was done so as to select about 50 to 100 data points for the Bode Plots. Thereafter, Matlab's bodeplot function from the System Identification Toolbox was used to plot the Bode Plot data. Lastly, Matlab's in house tfest

function in the system identification toolbox was used to estimate a transfer function using the frequency response data.

Matlab's tfest requires the user to input the number of poles and zeroes as its input. For this I simply tried fitting lower order models first to the frequency response data and then I verified whether it fit using the step response test. That means I start with n poles and (n-1) zeroes and increased n until there was the transfer function was verified using step response data. To verify that the transfer function is derived properly from the frequency response data, it is subject to step changes and compared to data derived from the same test performed in GeN-Foam. In this case, a 30K step increase in inlet temperature was performed on the transfer function. This 30K step increase was done because it simulates an unprotected loss of heat sink (ULOHS) transient. We assume no changes in fluid mass flowrate for simplicity. Physically, this would mean that the primary salt pump (PSP) is still running. In a similar manner, the same 30K step increase was performed for GeN-Foam and the average outlet temperature was measured. We quantify whether a fit is "good enough" by comparing the deviation to typical uncertainties in K type thermocouples which might be used to measure temperatures in an FHR.These uncertainties are around ±2.2K [Kollie et al., 1975]. Therefore, 2.2 K error bars are added to show the are added to show if differences are well within thermocouple measurement uncertainty.

Transfer Function and its Verification

The result from this process is a second order transfer function. The inlet temperature to outlet temperature transfer function computed in Matlab using 30K perturbation amplitude is:

$$G(s) = \frac{7.455 * 10^{-5}s - 7.726 * 10^{-8}}{s^2 + 0.0005918s + 2.268 * 10^{-7}} \tag{5.58}$$

Deriving Transfer Function using "tfest" from Matlab on Frequency Response Data The transfer function was obtained using Matlab's "tfest" on the frequency response data of the arbitrary reactor. Here, we present the Bode plot of the frequency response test for a 30K perturbation amplitude in Figure 5.19:

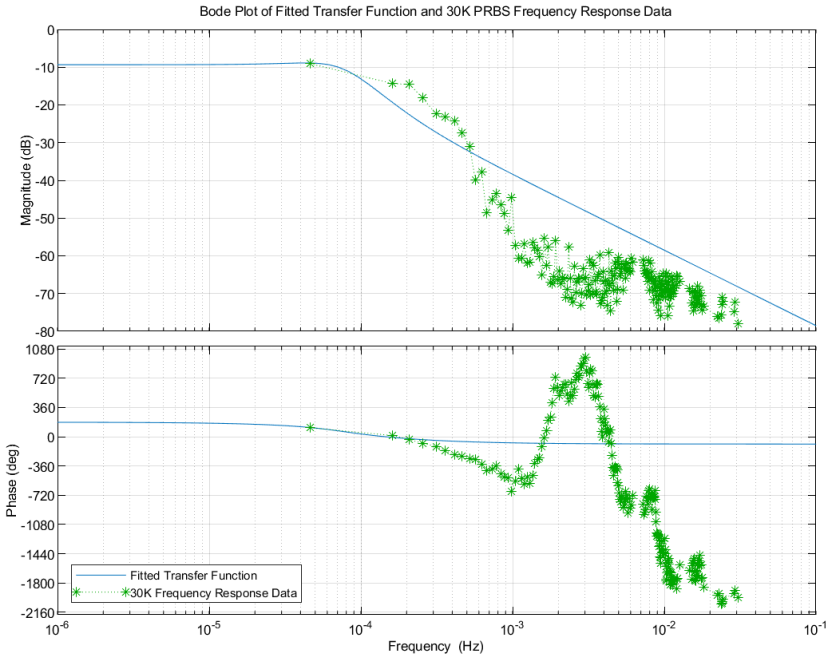


Figure 5.19: Bode Plot of Inlet Temperature to Outlet Temperature Transfer function using 30K Perturbation Amplitude, done using Matlab

For the gain bode plots, we use equation 5.59:

$$\text{Gain (dB)} = 20 \log_{10} \frac{\delta O}{\delta I} \tag{5.59}$$

Figure 5.19 plots the fitted transfer function alongside the frequency response data of the arbitrary reactor in GeN-Foam. To estimate the transfer function, Matlab's "tfest" requires that a suitable number of poles and zeroes are given as an input. Ideally, we would want to fit every single point in the plot and give place it in the transfer function. We may not have to worry as much about overfitting data (compared to fitting experimental data) in this case because the results are simulated and we do not have measurement uncertainties due to thermocouples or other measurement apparatus. However, adding more poles and zeroes to fit the transfer function proved troublesome. Sometimes, the tfest algorithm would produce a worse fit at when more poles and zeroes were given. Due to this issue, I decided that using a simpler data-fit model which fit most of the important data points would suffice. This importance can be quantified by the gain. Hence, I opted to ignore points below a certain gain. This threshold was -30 dB. Usually, a -30dB gain means that the output is about 3% the magnitude of the input. For a 30K temperature input, this is about 0.9K of output. We would then question whether we need to worry about 0.9K of temperature deviation. However, as mentioned earlier, this deviation is rather insignificant especially if we compare

it to ± 2.2K of thermocouple measurement uncertainty. From this perspective, those points should be safe to ignore. The other concern is system instability. This is because if there are unstable poles at any frequency, high or low, of any gain, then we cannot simply ignore them. Thankfully, the system and the transfer function are relatively stable. We shall see that this is the case when we subject the transfer function and arbitrary reactor in GeN-Foam to a step response test.

Another figure of merit to indicate whether the transfer function is a good fit is the steady state gain (gain at 0 Hz). However, we don't plot steady state gain on the Bode Plot. A proxy for this is that we place more importance on gains at lower frequencies. Practically, this means that the data points, especially that of the lowest frequency should be well fitted to using the transfer function. We see that this is the case for Figure 5.19. Using these figures of merit, we can justify that the fitted transfer function is a decently good fit to the frequency response data.

Validation using Step Response Tests While the Bode plots show that the fitted transfer function can replicate the frequency response data of the arbitrary reactor, the surest way we can convince ourselves that the fitted transfer function is indeed a good fit is using a step response test.

Therefore, the arbitrary reactor and fitted transfer function were both subject to the same step change. This step change is a 30K step increase in inlet temperature while the flowrate is kept constant. Such a step increase would mimic a ULOHS transient and is therefore interesting to study for FHR safety. The resulting step response plots in Figure 5.20:

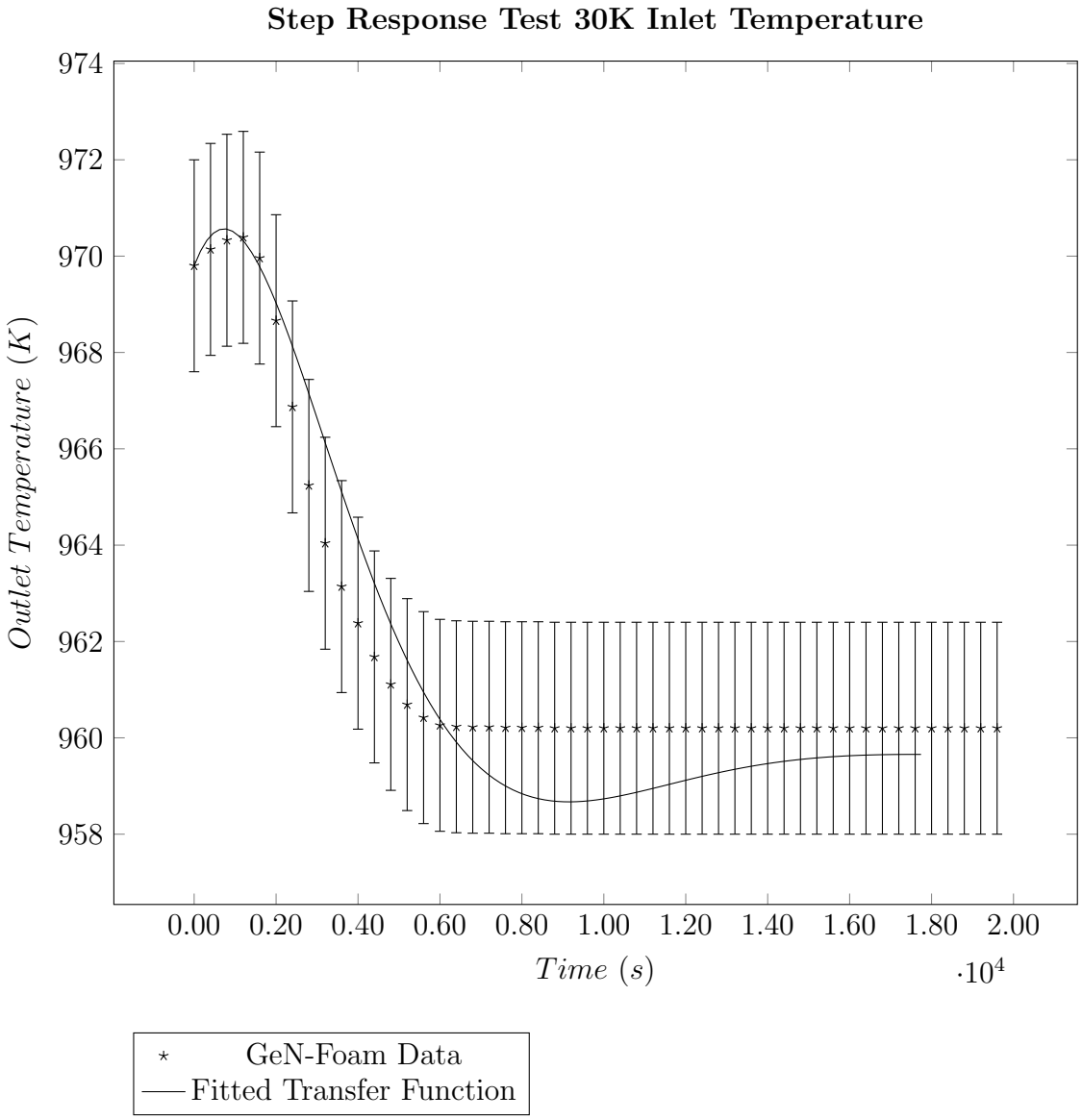


Figure 5.20: Step Input of 30K applied to Inlet Temperature for GeN-Foam compared to Derived Transfer Function

Here, we worry whether ignoring certain frequencies would result the transfer function being a bad fit for the arbitrary reactor behaviour. Indeed, we observe in Figure 5.20 that there is some deviation between the outlet temperature of the transfer function and arbitrary reactor. To quantify if this deviation is important, we can once again compare it to the thermocouple measurement error of $\pm$ 2.2 K. Thankfully, results in Figure 5.20 show that when both GeN-Foam and the transfer function are subject to the same 30K step increase in inlet temperature, the system responses match to within thermocouple measurement error.

Test for Nonlinearities with 10K Frequency Response Data Now, a transfer function model is a linear time invariant (LTI) model whereas a multiphysics reactor simulation would have non-linearities. An issue with using transfer functions is that transfer functions cannot capture the nonlinear behaviour of the system. To quantify non-linearities within the system, we can subject the multiphysics simulation to PRBS signals of different amplitudes. We show the frequency response data at for a 10K ampltiude PRBS signal is overlaid with with data from Figure 5.19 in Figure 5.21:

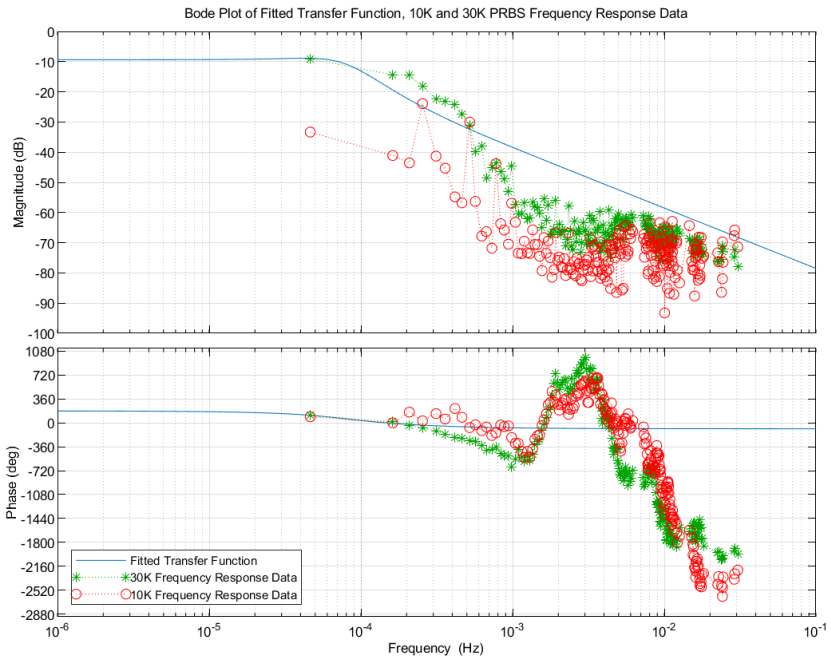


Figure 5.21: Bode Plot of Inlet Temperature to Outlet Temperature Transfer function overlaid with 30K amplitude frequency response data and 10K amplitude frequency response data

Figure 5.21 shows that some data points from the 10K frequency response dataset match that of the 30K PRBS dataset. However, there is significant deviation especially at low frequencies. As mentioned previously, one important figure of merit for how well a transfer function fits the frequency response data is the ability of the transfer function to replicate steady state and low frequency behaviour. This may be due to system non-linearities. In fact, at low frequencies, the gain is less than -30 dB. This means that the gain of the system at low level frequencies is on a comparable order of magnitude as noise. For example, at $5.2 * 10^{-5}$ Hz, the predicted gain from the transfer function is -10dB, but the gain for the 10K PRBS frequency response test is -30dB. Again, a -30dB gain means that the output is about 3% the magnitude of the input. For a 10K temperature input, this is about 0.3K of

output. Given a typical thermocouple measurement error of 2.2 K, it would be difficult to distinguish this output signal from noise in a real experiment. Hence, data points like these may not be good for fitting transfer functions.

These results seem to indicate that the transfer functions are probably not a good idea for fitting reactor feedback in general as the multiphysics simulation in GeN-Foam is showing some nonlinear behaviour. However, these non-linearities are sometimes small enough to ignore. Figure 5.22 shows that despite the non-linearities that show between 30K and 10K PRBS datasets, the non-linearities may be small enough to neglect:

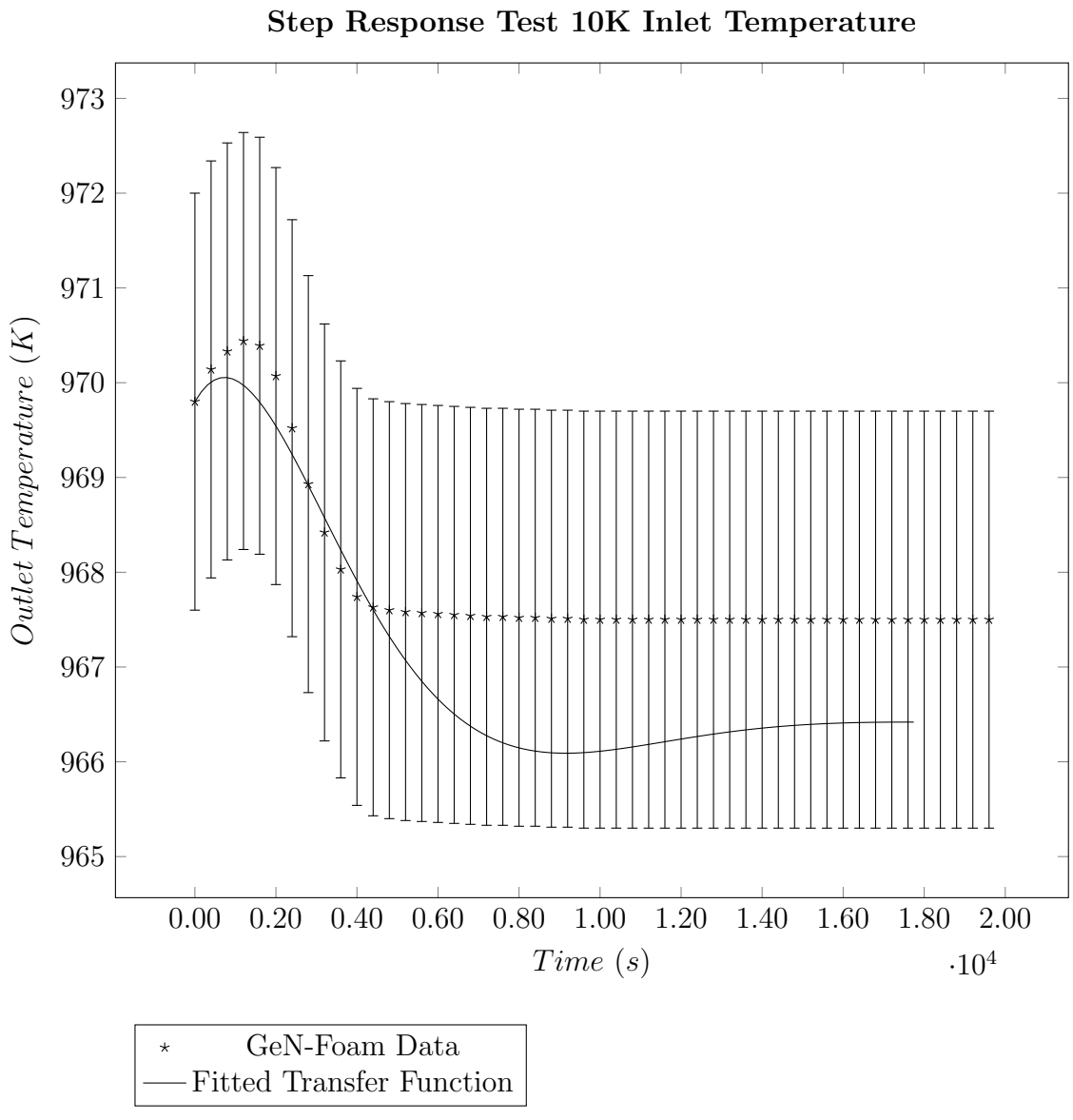


Figure 5.22: Step Input of 10K applied to Inlet Temperature for GeN-Foam compared to Derived Transfer Function

Figure 5.22 shows deviation in system response between the transfer function and the multiphysics reactor model in GeN-Foam. While there are deviations, they are smaller than a typical thermocouple uncertainty of $\pm$ 2.2K. Thus, they are statistically indistinguishable from measurement error if the reactor transient response data was obtained experimentally. This measurement error is shown by the error bars in Figure 5.22. Thus, despite the existence of non-linearities, the non-linearities themselves may be small enough to ignore, the transfer function would still be applicable at this temperature range.

Limitations of using Transfer Functions to model FHR Transient Behaviour

Step Response at 100K Step Input While deriving transfer functions from frequency response testing has worked well in nuclear reactor stability analysis, we also want to show that it has limits. We got away with non-linearities occurring between 30K and 10K PRBS frequency response tests because these were small. However, for the GeN-Foam simluated arbitrary FHR, non-linearities do become significant given a large enough step increase in inlet temperature. Thus, transfer functions meant to model linear time invariant (LTI) systems are not able to capture these nonlinear effects.

To prove this point, the arbitrary reactor and the fitted transfer function were both subject to a step increase of 100 K in inlet temperature. The resulting system responses show in Figure 5.23:

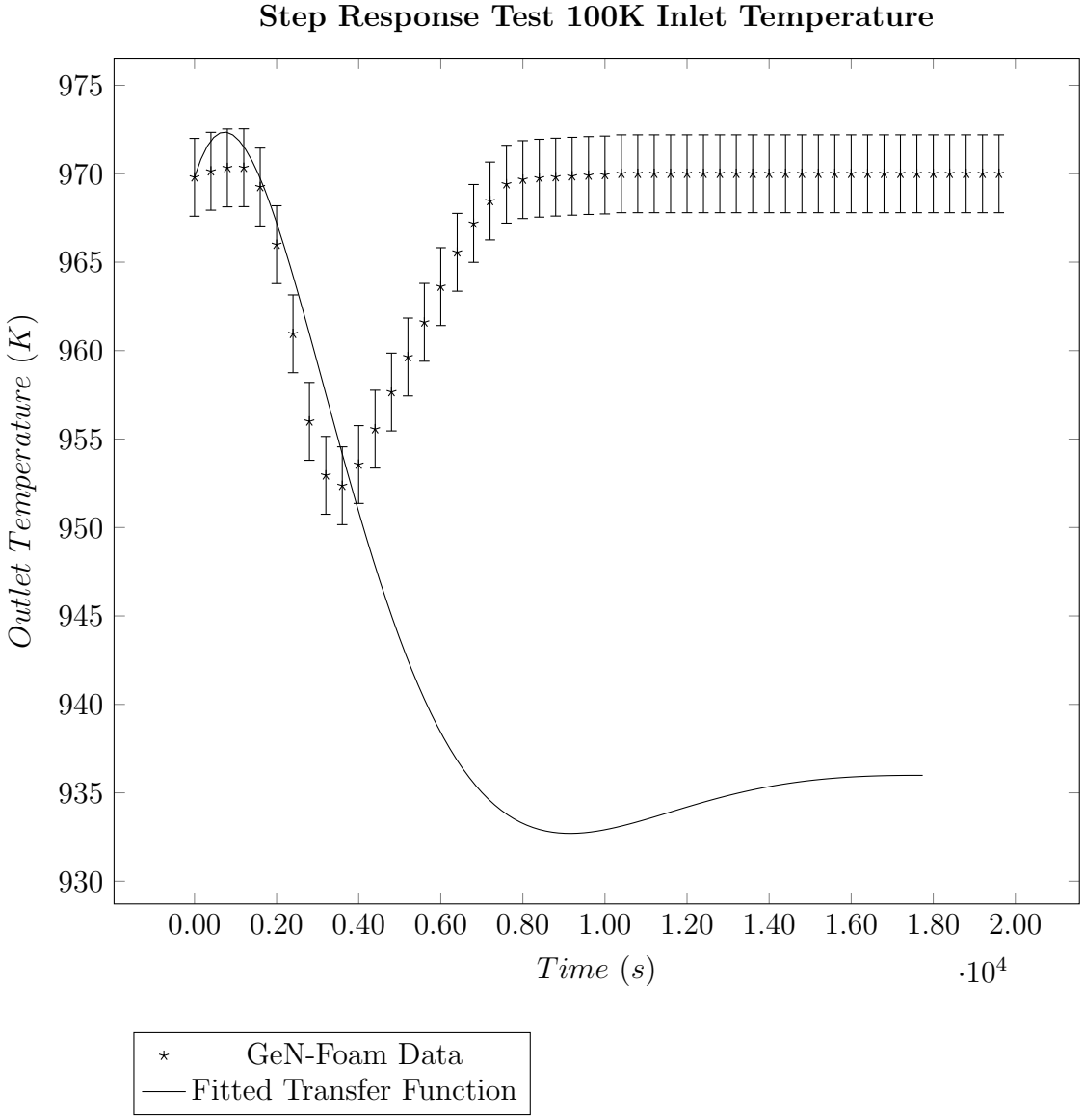


Figure 5.23: Step Input of 100K applied to Inlet Temperature for GeN-Foam compared to Derived Transfer Function

With a step input of $100K$ for the GeN-Foam simulated reactor, the inlet temperature increases from $873K$ to $973K$. The transfer function model predicts that the steady state outlet temperature would fall below $973K$. This would be an unphysical result considering the energy balance. The GeN-Foam model shows that the reactor outlet temperature dips initially from $970K$ to about $952K$, but rises back up to about $970K$ at 10,000s. I found it too much trouble for too little gain to wait for the outlet temperature to rise to $973K$, and hence, I just extrapolated the data up to 20,000s at $970K$. In reality, we should expect the outlet temperature to rise to $973K$ given the energy balance of the multiphysics reactor.

Either way, this seems to indicate that the pebbles in the core are absorbing heat from the FLiBe rather than supplying heat to it as the outlet temperature reaches some steady state. Of course, in a real reactor, this is an unphysical response because there would be decay heat, but we shall ignore that for now. The main point is that significant non-linearities exist where transfer functions fall short.

Frequency Response using 100K PRBS The bode plots also show that the frequency response, especially in the higher frequency regions for 30K and 100K PRBS tests do not match well especially when it comes to the Phase Angle plots in Figure 5.24:

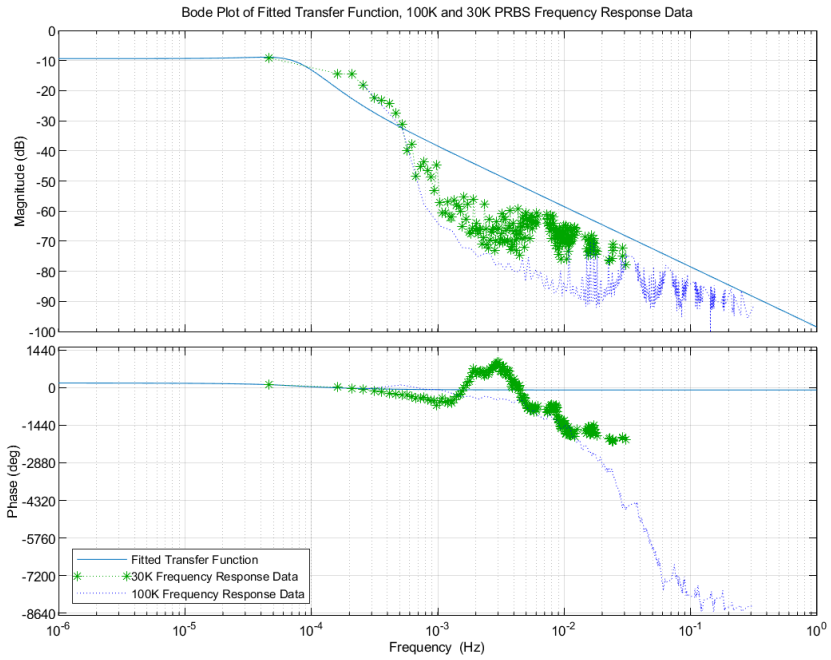


Figure 5.24: Bode Plot of Inlet Temperature to Outlet Temperature Transfer function overlaid with 30K amplitude frequency response data and 100K amplitude frequency response data

Data at lower frequencies was not captured in Figure 5.24. This is because I used a filtering algorithm to remove data points with gains that were too low so as to prevent my dataset from having too many points. This means that the steady state gain was essentially zero for the multiphysics reactor model.

Discussion of Non-Linearities Both step response and frequency response plots show that non-linearities emerge for coupled reactor multiphysics tests. This is because the multiphysics reactor simulation is a non-linear system. Firstly, cross sections vary with the

logarithm of temperature rather than temperature [X. Wang, 2018]. Therefore, given a constant flux, reactor power might vary linearly with the logarithm of inlet fluid temperature rather than the fluid temperature itself. For small enough perturbations, the log linear dependence may be small enough to ignore for this case, but this would not hold for larger perturbations.

Secondly, we also consider that the reactor cannot physically cool the fluid. When inlet coolant temperature increases, the fuel temperature also increases. This should lower k_{eff} and cause the reactor to lose power until another steady state is reached. However, we cannot expect the reactor to keep losing power till the power output becomes negative (or below decay heat value). This is a second source non-linearities besides the log temperature dependence. To visualise this, we can consider Figure 5.25:

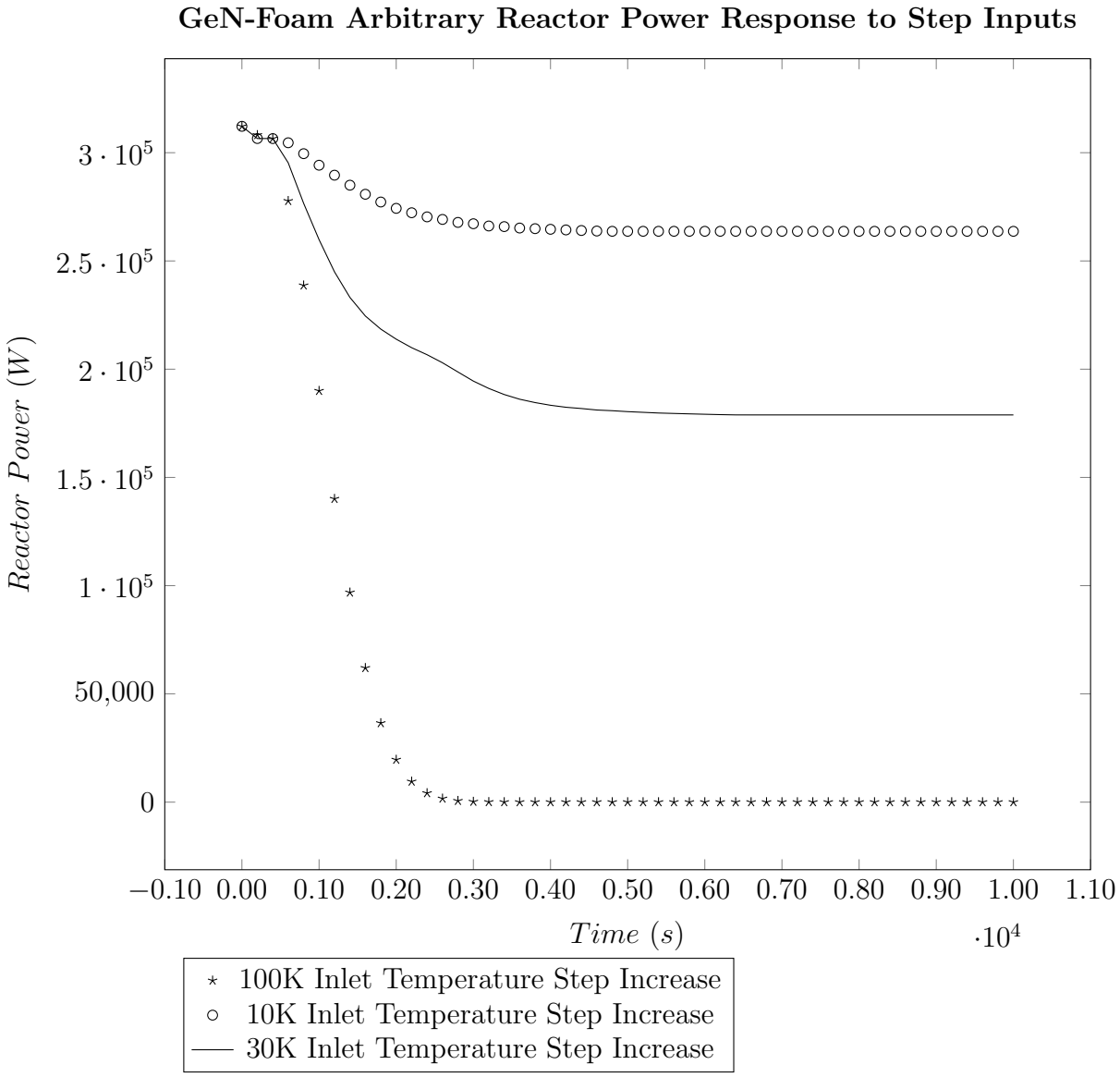


Figure 5.25: Reactor Power Response to Step Inputs of Increased Inlet Temperature

Figure 5.25 shows that at 10K step increase in inlet temperature, the reactor power decreases to around 264 kW from 312.2 kW. This is about a 15.4% power decrease for 10K increase in inlet temperature. For 30K step increase in inlet temperature, reactor power decreases to about 179 kW. This is an approximate decrease of 43% in reactor power, or 14.2% decrease in reactor power per 10K increase in inlet temperature. Now, if we were to linearly extrapolate either of these gains to a case of 100K increase in inlet temperature, we would get reactor power reduction of >100%. This means that the transfer function model is cooling the fluid rather than heating it. This is quite unphysical. Of course, the reactor (without accounting for decay heat) can only output at minimum 0 kW of power. Any

further increase in temperature beyond this point would produce a no difference in power output. If decay heat is included, then reactor power output cannot physically be lower than the decay heat. This is a second source of non-linearity.

To model these nonlinear responses, we may require a nonlinear model to develop our controllers. These nonlinear models are most definitely more complicated and quite outside the scope of this dissertation. However, it does show that controller development for simulated neutronics feedback would be quite complex if we wanted to capture linear and nonlinear behaviour. Thus, it is quite plausible that we would need to go through several iterations of controller prototypes in order to develop it. This makes digital twins ideal as a testbed because these iterations can be performed on a digital twin before implementation in the real physical system.

For this work, the objective is to demonstrate an iteration of simulated neutronics controller development using digital twins as a testbed. Hence, the focus will not be on developing a nonlinear controller, but rather demonstrate digital twins as a testbed for iterative controller development. The first iteration of the controller would be based on transfer functions because it is simple and quick to produce relative to more complex controllers. We shall complete this iteration by scaling this controller down to CIET and embed it in the digtal twin model of CIET.

5.4 Obtaining Scaled Transfer Function

In this section, we describe how the scaled transfer function for CIET is derived using GeN-Foam simulated data. For this, we use CIET's original scaling methodology [Zweibaum, J E Bickel, et al., 2015; Zweibaum, Guo, et al., 2016; Bardet and Per F Peterson, 2008] since was already established. This scaling methodology for temperature is described in equation 5.60:

$$T_{therminol\ VP-1}(^{\circ}C) = 0.3T_{FLiBe}(^{\circ}C) - 100 \tag{5.60}$$

For time scaling, we will simply scale time by 2/3.

These scaling parameters will be applied directly to the time domain data, only for the $30K$ amplitude PRBS and $30K$ amplitude step response. This is because we already verified that the transfer function can effectively replicate the step response behaviour for step inputs of $30K$ and below.

Scaled Transfer Function from Scaled Frequency Response Data

We repeat the procedure for estimating transfer function using tfest from Matlab for frequency response data to obtain a transfer function. The transfer function is:

$$G(s) = \frac{0.000119s - 2.201 * 10^{-7}}{s^2 + 0.0007903s + 6.667 * 10^{-7}} \tag{5.61}$$

The frequency response of the scaled transfer function is plotted with the scaled frequency response data in Figure 5.26:

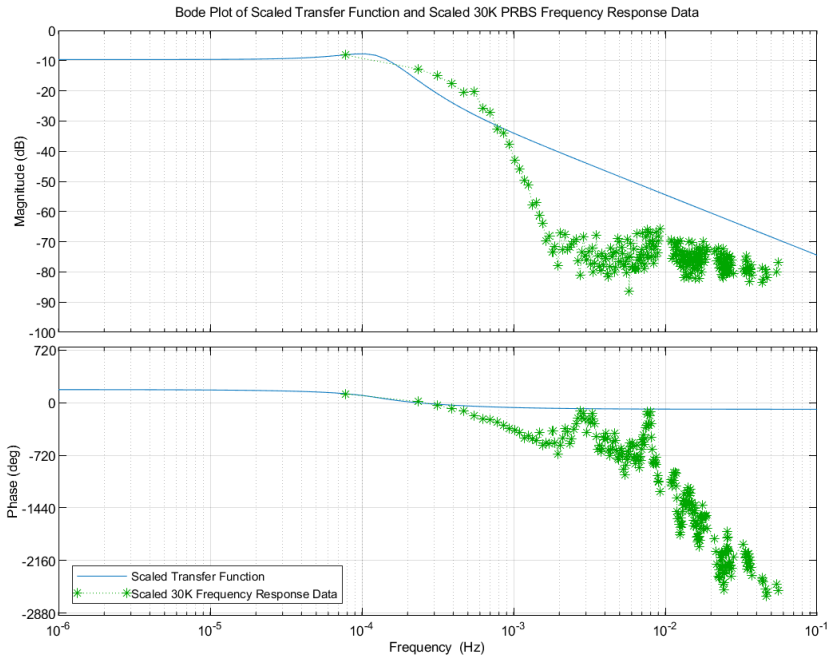


Figure 5.26: Bode Plot of Inlet Temperature to Outlet Temperature Scaled Transfer Function overlaid with Scaled 30K amplitude frequency response data

This time, for the tfest function, we assume that the transfer function has two poles and one zero for Matlab because the transfer function in the FLiBe cooled arbitrary reactor worked well with these settings. As before, I do not make attempts to fit data below -30dB of gain.

Verification of Scaled Transfer Function using Scaled Step Response Data

We also verify if this transfer function fits step response data. To do so, the arbitrary reactor step response data was also scaled using equation 5.60. The scaled step response data was compared to the step response of the transfer function with a scaled step input in inlet temperature. For a 30 K step input, this translates to a $9K$ or 9 $^{\circ}C$ step input for a scaled Therminol VP-1 or Dowtherm A system in CIET. A $9^{\circ}C$ step response for both scaled step response data from GeN-Foam and the scaled transfer function is presented Figure 5.27:

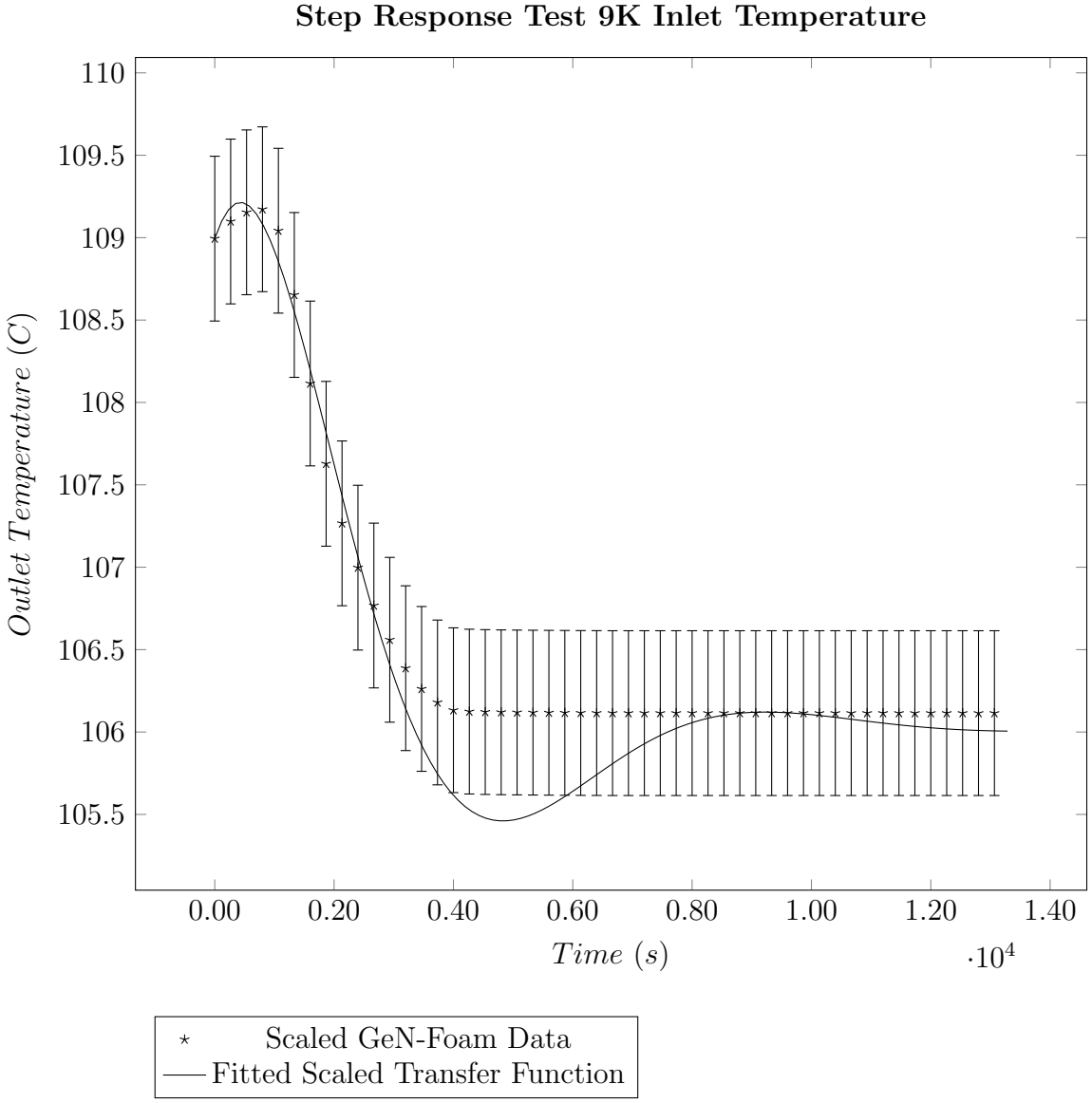


Figure 5.27: Scaled Step Input of applied to Inlet Temperature for Scaled GeN-Foam Data compared to Scaled Transfer Function

In Figure 5.27, we observe deviations between the scaled transfer function and the scaled frequency response data as expected. To ensure that these deviations not cause for concern, we must ascertain if the deviations exceed thermocouple measurement error. To do this, I add error bars to the graph. These error bars are $\pm 0.5°C$ as CIET used T type thermocouples with these measurement uncertainties [Zweibaum, Guo, et al., 2016]. We note that the transfer function has a slight overshoot compared to the scaled GeN-Foam data which becomes statistically significant given the thermocouple measurement uncertainty of $\pm 0.5°C$. This was not a problem before as the error bars before for type K thermocouples

were much larger at $\pm 2.2°C$. Of course, one could argue that this temperature error was not scaled properly for deviations in the scaled system. However, even when we take scaling into consideration, the scaled error for type K thermocouples using equation 5.60 is about $\pm 0.73°C$, which is larger than the T type thermocouple. If we were to use the $\pm 0.73°C$ measurement uncertainty as the measurement error, then the overshoot in Figure 5.27 becomes indistinguishable from measurement error.

Given this fact and that the transfer function followed the data most of the way through, I opted not to further optimise the fitting of the transfer function within the type T thermocouple measurement uncertainties. I consider the transfer function well fitted enough to replicate behaviour of the scaled GeN-Foam data.

5.5 Conclusion

In this chapter, described methods to construct the arbitrary reactor, which is a simple cylindrical shaped FHR without control rods. This is meant to be generate data for the surrogate model (transfer function) so that a simulated neutronics feedback controller may be developed. We then discussed how rationale for designing the relevant GeN-Foam inputs as well as the caveats for this rather simple and crude multiphysics model.

We also obtained transfer functions for the GeN-Foam simulated arbitrary FHR for ULOHS transients. We found that these transfer functions were able to reproduce GeN-Foam reactor transient data in step response tests for $30K$ as well as $10K$ step increases in inlet temperatures. Nonlinearities were observed for $100K$ step inputs and frequency response data.

For simulated neutronics feedback in CIET, we have also derived a scaled transfer function using a similar methodology which can be used to design a controller. This transfer function would then be suitable for programming into CIET's Digital Twin.

5.6 Future Work

The reader should note that the multiphysics model is quite crude in nature. Furthermore, the data driven surrogate model used is a simple SISO transfer function. These simple models are meant to demonstrate the concept of using a data driven surrogate models to construct simulated neutronics feedback controllers. We can improve upon these methods to give us better data driven surrogate models in future. This section discusses possible future work and projects which can be used to improve upon the current models. These works can be done in the hope that an electrical heater in any IET can be modified to behave like a real reactor with real feedback in real time.

Reactor Fidelity

While constructing a high fidelity model is not the objective of this work, we could always improve the fidelity of the multiphysics reactor model to ensure that it behaves closer to a real reactor when disturbances or transients are introduced.

We may modify GeN-Foam's thermal hydrualic models especially in its ability to model graphite reflector structures and alter the pebble heat transfer model to more accurately describe heat transfer in an environment laden with TRISO particles. We may include decay heat, burnup and reactor poisons in future models as well. We could also improve upon the energy or angular discretisation by using SP3 neutronics with more energy groups.

Reactor Design

Of course, in future, we may not necessarily use the arbitrary reactor but use a benchmark in literature such as the gFHR [Kile et al., 2022]. We hope to model control rods as well so reactivity insertion accidents and transients can be simulated as it was for the TMSR-SF and Mark I PB-FHR [X. Wang, 2018].

Improvements for Data Driven Surrogate Models

The data driven surrogate models used for controller design need to be able to account for nonlinearities in future. Hence, nonlinear data driven surrogate models such as those used in the context of AI and deep neural networks can be explored in future.

Chapter 6

Simulated Neutronics Feedback Controller Development

6.1 Introduction

So far, we have constructed both the thermal hydraulics library and a validated model of CIET's heater based on the CIET v2.0 Heater without insulation. We are now ready to test the Digital Twin's effectiveness as a test bed for iterative development of a simulated neutronics feedback (SNF) controller.

To do so, I created graphical user interface (GUI) OPC-UA Rust client with a control systems and transfer simulation simulation toolbox , also written in Rust, to interface with the Digital Twin server over a local area network (LAN). After this, I began testing and developing the SNF controller using this Rust GUI server and client. For testing and development of the SNF controller, I only performed only two design iterations in to get the SNF controller to a roughly reproduce the transient data given by the simulated reactivity transfer function developed in the last chapter. During these design iterations, I roughly recorded the number of times I needed to restart the Rust OPC-UA client as well as the rough amount of time I needed for these design iterations. By doing so, I could gauge how many experiments I would have needed to perform in CIET if I used CIET to perform the SNF controller design iterations. By comparing the difference between the time used for design iterations in the Digital Twin and the estimated amount of time used for design iterations in CIET, I could then calculate how much faster the iterative design process was when it was performed within the Digital Twin.

In this chapter, I first outline how I quantify the speed increase from using the CIET Heater Digital Twin to develop the SNF controller. Next, I outline how the real-time transfer functions and control systems simulator library was developed in Rust and then validated. I then describe how the OPC-UA GUI client was developed in Rust. Lastly, I present the results describing the iterative design process of the SNF controller, how well the controller performs at each iteration stage, and roughly how much time was saved by using the Digital

Twin of CIET's Heater.

6.2 Methods

Quantification of Speed Increase due to use of Digital Twin for SNF Controller Development

The main goal of this chapter is to quantify the speed increase achieved when using a Digital Twin of CIET (or CIET's Heater) as opposed to CIET for iterative development of a SNF controller. To quantify this, I need to know how much time I used to develop the SNF controller when using a Digital Twin and how much time I used to develop the SNF controller in CIET. Unfortunately, it was difficult to develop a SNF controller in CIET due chiefly to the COVID-19 pandemic. Therefore, I had to estimate the amount of time used for an experiment in CIET.

For this, I based my estimates on anecdotal experience. In my experience when using CIET prior to the COVID-19 pandemic, I knew that there was a planning and approval phase prior to performing an experiment. This included creating documentation which outlined the steps required to operate CIET safely. The next step was to vet and obtain approval for the aforementioned documentation for the purposes of Nuclear Quality Assurance (NQA). This was important not only for safety, but also to ensure that the experiment was well documented and repeatable. Otherwise, if certain datasets were missing, or certain procedures were not well recorded, the experiments would then have to be repeated. Thereafter, I would have had to find a partner to go with me to the lab where CIET was for safety reasons. This was because operating CIET meant heating up a fairly large quantity (about 40 kg) of Dowtherm A (otherwise known as Therminol VP-1). Should hot Therminol VP-1 leak or spill, the vapours from this oil could have caused dizziness or even present a fire hazard. Therefore, a buddy system was enforced. Nevertheless, needing to ensure that a lab buddy or partner was present when performing experiments prevented me from performing experiments everyday. Even when I could perform experiments, there were bugs in Labview (including race conditions) which the team had to solve before performing the actual experiment which prevented the actual experiment from being run.

All in all, for every iterative design experiment, I estimated that I needed three days of actual experimental work in order to debug CIET's setup. I also assumed that a lab buddy could only be found for three out of a five day work week. Thus, a rule of thumb would be that one physical experiment in CIET would require about one week to perform successfully. Of course, one might imagine that as I interfaced more with CIET, the time required to solve the bugs would have lessened to the point where I could perform "nth-of-a-kind"(NOAK) experiments. At that point, I may have needed one or two experimental days perhaps to perform one experiment successfully. However, even with NOAK experimental experience, I would have still needed to type out experimental procedures and perform data review and analysis of the experiment. Moreover, I would have needed time to design controllers

in Labview and ensure that they worked in CIET, hopefully without causing accidents. These, along with other potential contingencies such as oil spills, running out of cover gas, maintenance or other issues, would perhaps have accounted for the remaining four days of the week in an experimental run. Also, one might argue that I could do more than one experiment in a day. Unfortunately, the time required to start up and shut down CIET alone (apart from experiments) would have been roughly three hours. This is because I needed time for valve alignment in addition to time for CIET to reach steady state. This would have taken up nearly half the work day. Practically speaking, running more than one experiment a day would have been difficult.

To translate the number of runs in the Digital Twins to the number of experiments I needed in CIET, I used this rough estimate: every one restart of the Rust OPC-UA client for the Digital Twin equates to one full startup and shutdown run of CIET. This is because for SNF controller design iterations, I would have needed to edit the SNF controller, which would have been embedded in the Advanced Reactor Control and Operations (ARCO) system. ARCO would need to be running whenever CIET was running. To edit ARCO and restart ARCO safely, I would have needed to shut down CIET first. This is because restarting CIET's control system (ARCO) while CIET's heater was online is generally an unsafe procedure. Given that this is the case, every restart of CIET's control system (CIET) would necessitate a full shutdown and startup procedure of CIET's heaters, fans and pumps. Since the Rust OPC-UA client was meant to mimic the role of ARCO, restarting the client would effectively mean restarting ARCO and CIET. Therefore, I used this one Rust OPC-UA client restart to one CIET experiment equivalence. Using this relation, I can then estimate the amount of time saved in using the Digital Twin to iteratively develop a SNF controller, at least in the early stages.

Now that we have discussed a method to quantify the relative speed increases, we can now discuss how the SNF controller is designed.

SNF Transfer Function for Coolant Inlet Temperature Changes

Let us begin discussing how our SNF controller is designed by first revisiting the SNF transfer function. As discussed previously, the reactor feedback due to coolant inlet temperature changes is described by transfer function of reactor inlet temperature ΔT_{inlet} to reactor outlet temperature ΔT_{outlet}:

$$G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s) = \frac{0.000119s - 2.201 * 10^{-7}}{s^2 + 0.0007903s + 6.667 * 10^{-7}} \tag{6.1}$$

Sometimes, it is customary to change the constant in the denominator to 1 in what is known as standard form [Seborg et al., 2016]. This makes it simpler to observe what the steady state gain is. If we were to normalise the constant to 1 within the denominator:

$$G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s) = \frac{178.49s - 0.33013}{1.4999 * 10^6 s^2 + 1185.4s + 1} \tag{6.2}$$

From Equation 6.2, we can see that the steady state gain as $s \to 0$ is -0.33013 where s is complex frequency in units of $\frac{1}{time}$. Using Wolfram Alpha [Alpha, 2023], I found that the poles of Equation 6.2 are complimentary complex roots of $s = -0.00039515 \pm 0.000714532i$. Since the, real part of the complex roots are negative, this second order system is stable. The imaginary parts correspond to the oscillation frequency of the exponentially decaying sinusoids. The zero for this transfer function is at $s = 0.0018495$. These are characteristic of an under damped second order system with a real zero. Since we are developing SNF controllers, we need to ascertain if the controllers produce the desired behaviour given user inputs. For iterative SNF controller design trials, we shall attempt to simulate this reactor feedback transfer function in real-time as well and verify if the SNF controllers we developed achieve this goal. Of course, we shall need to test and develop the SNF controllers such that they run in real-time as well. To do so, it is important to derive the expression for the controller transfer function ($G_c(s)$) so that I could ascertain what kind of transfer functions I needed to simulate in real-time.

SNF Controller Modelling Requirements

The SNF controller transfer function, commonly denoted as $G_c(s)$ could take several forms. The first approach that I took to develop the SNF controller was to design it as a feedforward controller. In this case, I designed a controller such that:

$$G_c(s) = G_{CIET\ heater\ inletT\ to\ heaterPower}(s) = \frac{G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s)}{G_{CIET\ heater\ heaterPower\ to\ outletT}(s)}$$

Where $G_{CIET\ heater\ inletT\ to\ heaterPower}(s)$ is the transfer function describing the heater inlet temperature to heater power signal. $G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s)$ represents the scaled transfer function describing the reactivity feedback due to changing inlet coolant temperature at constant flow rate. $G_{CIET\ heater\ heaterPower\ to\ outletT}(s)$ represents the transfer function of the heater power to heater outlet temperature (at BT-12) at a constant flowrate of 0.18 kg/s. In theory, an increased inlet temperature to the heater should have caused the heater power to decrease after some time, and therefore the temperature difference between the inlet and outlet temperature would also decrease accordingly after some time. However, I found out after some initial test runs that in this single input single output (SISO) controller model, I neglected to program the heater to compensate for the increased inlet temperatures. For a working feedforward controller, I would have needed to consider the heater as a multiple input multiple output (MIMO) model and used a state space model rather than just a single SISO transfer function $G_{CIET\ heater\ inletT\ to\ heaterPower}(s)$. Therefore, this single SISO model was insufficient for the SNF controller, a MIMO controller or some other controller would have to be used. Thankfully, I also found out an alternative method of designing a SNF controller from performing early test runs. From those same initial iterative design test runs, the timescales of the SNF transient ($\mathcal{O}(10^4\ s)$) long in comparison to coolant recirculation time recirculation time of about 90 seconds. This was also much longer

than that of the timescales for conjugate heat transfer within the CIET Heater ($< 10^2$ s). Therefore, feedforward control may not be strictly necessary for gradual changes in heater outlet temperature. In such a case, feedback control may have been sufficient. Hence, for the purposes of this work, I deemed that deriving a state space model would have been too much work especially if a simple Proportional, Integral and Derivative (PID) controller sufficed. Nevertheless, I still include the derivation of $G_{CIET\ heater\ inletT\ to\ heaterPower}(s)$ in the methods section because it was used in the first iteration of the reactivity feedback controller. Moreover, $G_{CIET\ heater\ inletT\ to\ heaterPower}(s)$ would have still been useful in future state space space models of $G_c(s)$ for a feedforward controller because it may have been one of the constituent transfer functions within the matrix of transfer functions. Therefore, I still present the derivations in the methods section.

Now, to obtain $G_{CIET\ heater\ inletT\ to\ heaterPower}(s)$, I needed to obtain two transfer functions. Firstly, $G_{CIET\ heater\ heaterPower\ to\ outletT}(s)$ which is the heater power to heater outlet bulk temperature (BT-12). Secondly, I needed $G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s)$, which is the reactor feedback transfer function derived from perturbing the arbitrary FHR constructed in GeN-Foam. Let us first obtain $G_{CIET\ heater\ heaterPower\ to\ outletT}(s)$.

From De Wet's work, an empirical transfer function of the heater power to heater outlet temperature was determined [De Wet and Per F Peterson, 2020]:

$$G(s) = e^{-4s}\frac{3.217 * 10^{-5}s^3 + 6.675 * 10^{-7}s^2 + 1.139 * 10^{-8}s + 2.423 * 10^{-11}}{s^5 + 0.2251s^4 + 0.01688s^3 + 0.0003548s^2 + 3.057 * 10^{-6}s + 1.632 * 10^{-9}}$$

This transfer function is an empirical transfer function of a closed loop response where heater power was varied while CTAH fan frequency was kept constant as opposed to keeping the CTAH outlet temperature constant. Therefore, thermal pulses could traverse the loop several times before the modes died out. Thus, we would observe low frequency modes in the frequency response tests. However, for the purposes of developing $G_c(s)$ for the SNF controller, I was not interested in the thermal pulses traversing the loop over long timescales. I was only interested in the CIET v2.0 Bare Heater. Therefore, I wanted another transfer function which did not include the effect of the rest of CIET's loop. While Poresky has frequency response test data of the insulated heater v2.0 [Poresky, 2017], a transfer function for the heater power to heater outlet temperature has not yet been determined. Therefore, we need to first determine the transfer function for the heater power to heater outlet temperature without considering the lower frequency modes. This can be easily done in Matlab using existing step response data for expediency. If one desires to use a manual method instead, Chen's method in literature can also be used if the system is a second order system [L. Chen, J. Li, and Ding, 2011]. For this work, I simply inspected the Bare Heater's step response data and through trial and error, manually fitted a second order model using LibreOffice Calc because it was convenient. This is because some of the Linux computers I used did not have Matlab.

Now, for the Heaver v2.0 Bare step response data, we could use either the step response data from De Wet's transfer function at least for the short term response in order to generate

a new transfer function. Alternatively, we could use existing step response data from the existing Heater v2.0 Bare Digital Twin in order to develop a new controller. I did the latter since the controller is meant to be calibrated for the heater v2.0 Bare Digital Twin. This Heater v2.0 Bare Step Response data is presented in Table 6.1 for a -500 watt step input to the heater Digital Twin at $t = 100s$:

Time (s)	Heater Outlet Temperature (BT-12) $°C$	Time (s)	Heater Outlet Temperature (BT-12) $°C$
100.8	102.41	123	101.17
102	102.41	124.8	101.12
103.8	102.41	126	101.1
105	102.32	127.8	101.06
106.8	102.22	129	101.04
108	102.13	130.8	101.02
109.8	102	132	101.01
111	101.87	133.8	100.99
112.8	101.72	136.8	100.97
114	101.63	139.8	100.96
115.8	101.5	142.8	100.95
117	101.43	145.8	100.95
118.8	101.33	148.8	100.94
120	101.28	151.8	100.94
121.8	101.21	154.8	100.94
		157.8	100.94

Table 6.1: CIET Heater v2.0 Simulated Step Response Test

The resulting data in Table 6.1 is plotted in Figure 6.1:

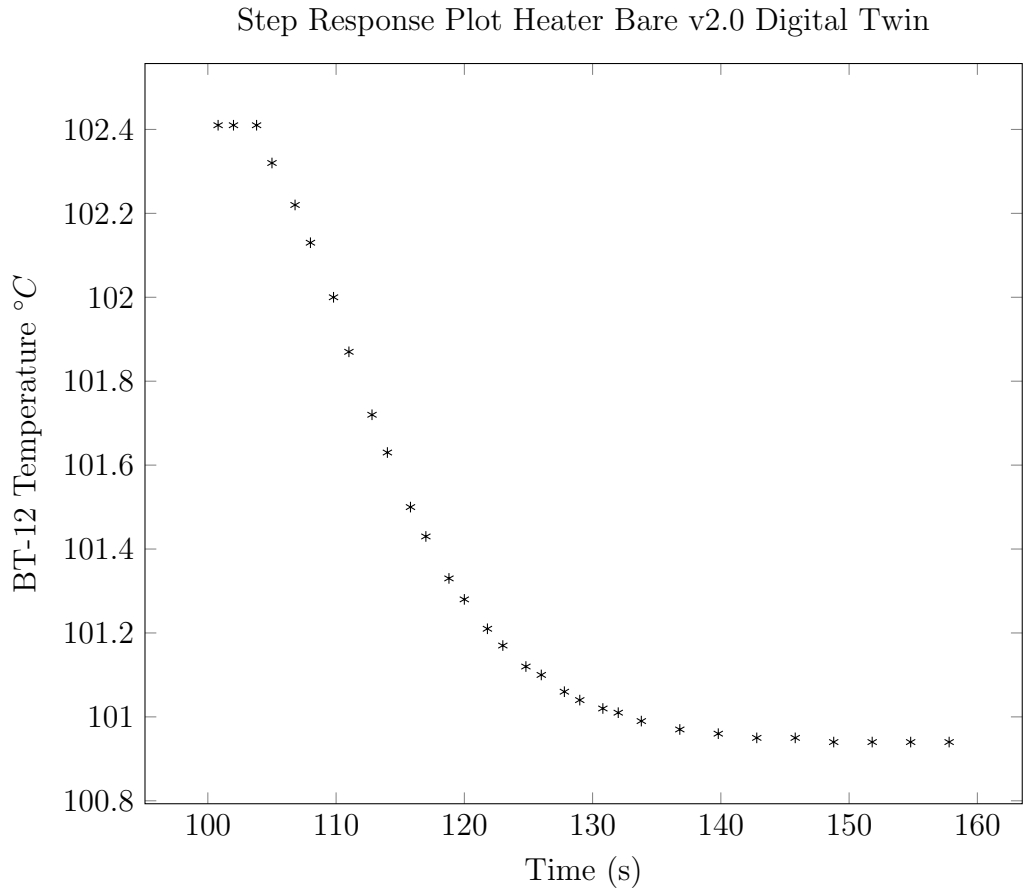


Figure 6.1: Step Response Plot for Heater v2.0 Bare Simulated with 500 Watt Power Step Decrease at t=100s

I then used data in Table 6.1 to obtain an empirical transfer function of the Heater v2.0 Bare to obtain $G_c(s)$. To fit an empirical transfer function to data in Table 6.1 manually in LibreOffice Calc, I first determined the form of the transfer function by observing the shape of the step response. For the heater bare v2.0 Digital Twin, its step response data in Figure 6.1 does not contain an overshoot. This means we could model it using a critically damped or overdamped system plus some dead-time. The simplest method is, of course, to use a First Order Plus Dead Time (FOPDT) system. Based on De Wet's Transfer Function, there is dead-time of approximately four seconds. To find $G_c(s)$, we traditionally invert this FOPDT transfer function. Therefore, we need to somehow invert dead-time. The textbook approach is to use a first order Padé approximation to approximate dead time before inversion [Seborg et al., 2016]. By using this first order Padé approximation, the FOPDT system is approximated as a second order system. Given this case, it is convenient to just fit a second order system to this data rather than fit a FOPDT system that has to be approximated as a second order system anyhow. Therefore, I simply fitted a second order system to the data in Table 6.1. This should be okay because it is common in textbook approaches to approximate

arbitrary transfer functions, including second order transfer functions, as FOPDT.

Now, to fit a second order system to the step response shown in Figure 6.1, I observed that the response in Figure 6.1 has an initial response behaviour. This could indicate a second order system with a zero. Since there is no overshoot, I am expecting an overdamped or critically damped system with two real poles. A transfer function with two real poles and one zero can be represented in the form:

$$G(s) = \frac{K_p(T_3 s + 1)}{(T_2 s + 1)(T_1 s + 1)} \tag{6.3}$$

Where T_1, T_2 and T_3 are time constants and K_p is the process gain. For such a transfer function, the time domain unit step response is [L. Chen, J. Li, and Ding, 2011]:

$$y(t) = K_p \left[1 + \frac{T_3 - T_1}{T_1 - T_2} \exp\left(-\frac{t}{T_1}\right) - \frac{T_3 - T_2}{T_1 - T_2} \exp\left(-\frac{t}{T_2}\right)\right] \tag{6.4}$$

In Equation 6.4, $y(t)$ represents the deviation in the output from some steady state and t is the time elapsed from the start of the step response. We can adapt this to fit data in Table 6.1 given that the steady state temperature is $102.41°C$. In adapting Equation 6.4 for our data, we can obtain an expression for temperature profile given a step response with magnitude a_0:

$$T(t)(°C) = 102.41 + a_0 K_p \left[1 + \frac{T_3 - T_1}{T_1 - T_2} \exp\left(-\frac{t}{T_1}\right) - \frac{T_3 - T_2}{T_1 - T_2} \exp\left(-\frac{t}{T_2}\right)\right] \tag{6.5}$$

Now, for this step response in Figure 6.1, $a_0 K_p = -1.47°C$. $a_0 = -500\ watts$, therefore, $K_p = 0.00294 \frac{°C}{watt}$. With this in mind, I fitted a transfer function with two poles and one zero using LibreOffice Calc. Again, while Chen has proposed methods for fitting a second order transfer function [L. Chen, J. Li, and Ding, 2011], however, for the use case in this dissertation, I found more expedient, and convenient to perform manual trial and error in Libreoffice Calc until suitable parameters were obtained. In doing so, I obtained $T_1 = 8.5\ seconds$, $T_2 = 5.5\ seconds$, and $T_3 = -1.5\ seconds$. Therefore, if we fit a step the step response data using these parameters:

$$T(t)(°C) = 102.41 + a_0 K_p \left[1 + \frac{-1.5 - 8.5}{8.5 - 5.5} \exp\left(-\frac{t}{8.5}\right) - \frac{-1.5 - 5.5}{8.5 - 5.5} \exp\left(-\frac{t}{5.5}\right)\right]$$

$$T(t)(°C) = 102.41 + a_0(watts)0.00294 \left[1 - \frac{10}{3} \exp\left(-\frac{t}{8.5}\right) + \frac{7}{3} \exp\left(-\frac{t}{5.5}\right)\right] \tag{6.6}$$

In the frequency domain, the transfer function is:

$$G(s)_{CIET\ heater\ heaterPower\ to\ outletT}\left(\frac{^{\circ}C}{watt}\right) = 0.00294\frac{-1.5s+1}{(s+5.5)(s+8.5)} \tag{6.7}$$

To validate this empirical transfer function, I performed a step response of this transfer function in Scilab [Merzlikina and Prochina, 2020] using a step amplitude of -500 *watts* and recorded its step response. I then superimposed the step response of this fitted transfer function onto the experimental.

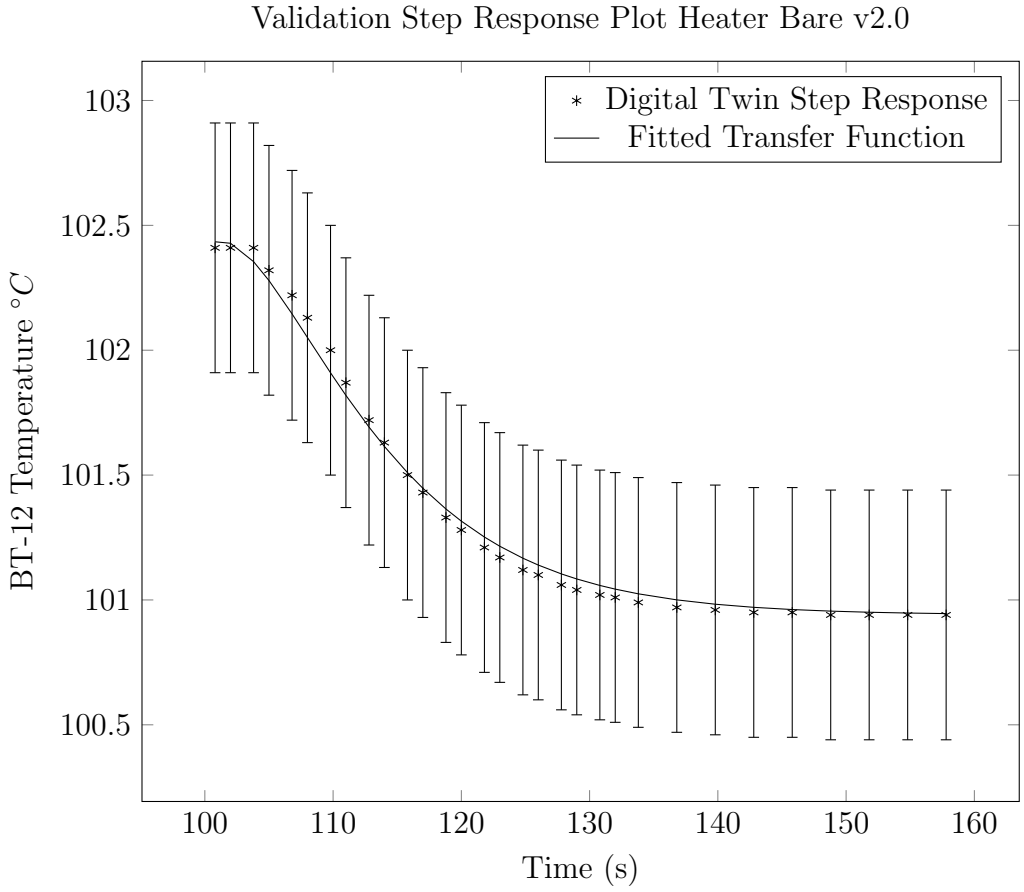


Figure 6.2: Step Response Plot for Heater v2.0 Bare Simulated with 500 Watt Power Step Decrease at t=100s with fitted Transfer Function

Error bars of $\pm 0.5\ K$ are added to Figure 6.2 in order to show that the magnitude of the deviation between transfer function and Heater Digital Twin Data is small in comparison to the typical Type T thermocouple measurement error encountered in CIET [Nicolas Zweibaum, 2015]. Therefore, I deemed this transfer function, $G_{CIET\ heater\ heaterPower\ to\ outletT}(s)$, to have fitted well enough to experimental data. With both $G_{CIET\ heater\ heaterPower\ to\ outletT}(s)$ as well as $G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s)$, I could find $G_{CIET\ heater\ inletT\ to\ heaterPower}(s)$.

This would be the transfer function for my first iteration of SNF controller. I obtain $G_{CIET\ heater\ inletT\ to\ heaterPower}(s)$ as follows:

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s) = \frac{G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s)}{G_{CIET\ heater\ heaterPower\ to\ outletT}(s)}$$

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s) = \frac{\frac{0.000119s - 2.201*10^{-7}}{s^2+0.0007903s+6.667*10^{-7}}}{0.00294\frac{-1.5s+1}{(s+5.5)(s+8.5)}}$$

Therefore,

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s)\left(\frac{^\circ C}{watt}\right) = 340.136\frac{(s+5.5)(s+8.5)}{-1.5s+1}\frac{0.000119s - 2.201*10^{-7}}{(s^2+0.0007903s+6.667*10^{-7})}$$

Alternatively, we can write:

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s)\left(\frac{^\circ C}{watt}\right) = 340.136\frac{(s+5.5)(s+8.5)}{-1.5s+1}\frac{178.49s - 0.33013}{1.4999*10^6 s^2+1185.4s+1}$$

Now, unfortunately, this transfer function is potentially inherently unstable due to the $(-1.5s+1)$ pole. However, when performing internal model control, it is customary to simply remove the unstable pole (colloquially known as the "bad part") of the controller [Bequette, 1999]:

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s)\left(\frac{^\circ C}{watt}\right) = 340.136\frac{(s+5.5)(s+8.5)}{1}\frac{(178.49s - 0.33013)}{1.4999*10^6 s^2+1185.4s+1}$$

When I removed the unstable pole, I ran into another issue: this controller is not physically realisable as it has two poles and three zeroes. To solve this, I added a low pass filter to it. This is okay because it is customary also to add low pass filters to physically non realisable transfer functions, such as transfer functions describing derivative controllers

($G(s) = s$), so that the resulting transfer functions are physically realisable [Seborg et al., 2016]. If we added a filter with characteristic timescale for the low pass filter, τ_{filter}, the resulting transfer function is:

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s)\left(\frac{^{\circ}C}{watt}\right) = 340.136\frac{(s+5.5)(s+8.5)}{\tau_{filter}s+1}\frac{(178.49s-0.33013)}{1.4999*10^6s^2+1185.4s+1}$$

With a physically realisable transfer function for the controller, I could then determine τ_{filter} and design the appropriate programming classes and structures for the transfer function. For this work, I simply used $\tau_{filter} = 0.1\ s$ because, at least for a derivative controller with derivative time τ_d, the appropriate derivative controller with filter has a transfer function [Seborg et al., 2016]:

$$G(s) = \frac{\tau_d s}{\tau_d \alpha s + 1}$$

Where α is usually around 0.1 [Seborg et al., 2016]. Now, since the coefficient of s for both factors in $(s + 5.5)(s + 8.5)$ is 1, $\tau_d = 1\ s$ I decided that $\alpha\tau_d$ should be 0.1 seconds. Since $\tau_{filter} = \alpha\tau_d$, $\tau_{filter} = 0.1\ s$. Hence, we leave ourselves with my first iteration of the SNF controller transfer function:

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s)\left(\frac{^{\circ}C}{watt}\right) = 340.136\frac{(s+5.5)(s+8.5)}{0.1s+1}\frac{(178.49s-0.33013)}{1.4999*10^6s^2+1185.4s+1}$$

With this first iteration, I found out that this transfer function, meant to be designed as a feedforward controller, was not able to reject the initial increase in heater outlet temperature brought about by increase in heater temperature. Therefore, a second iteration using some other control schemes were required. In particular, I choose Proportional, Integral and Derivative (PID) feedback control for my second iteration of SNF controller due to its simplicity. The set point of which is the desired outlet temperature of the scaled down version of the arbitrary FHR. For this, I needed the capability to both simulate the transfer function for the scaled arbitrary FHR, and to enable PID control based on real-time readings of the OPC-UA server from the OPC-UA client. Now, PID controllers in textbooks are applied to the error ε where $\varepsilon = y(t) - y_{sp}(t)$. Here $y(t)$ is the controlled variable and $y_{sp}(t)$ is the user specified set point (sp) of the controlled variable. However, implementing PID control in this manner often causes proportional and derivative kicks or instabilities. To eliminate

them, the derivative portion of the controller is usually applied in the feedback loop to $y(t)$ rather than the error ε [Seborg et al., 2016; Perry and Green, 2015]. The controller output, which is the input to the process $u(t)$ takes the following form in the time domain [Perry and Green, 2015]:

$$u(t) = K_c \left(\varepsilon + \frac{1}{\tau_I} \int \varepsilon dt - \tau_d \frac{dy(t)}{dt} \right)$$

Of course, the same derivative filter described earlier can be applied to the derivative portion of the controller so that it is physically realisable [Seborg et al., 2016]. Based on these two controller designs for the SNF controller meant to run within the OPC-UA client, the OPC-UA client would have to simulate transfer functions as well as PID controllers in real-time. To ensure that this is done well, I took the following preliminary steps:

Firstly, I needed to obtain functionality to simulate transfer functions in real-time using real-time inputs. This allowed me to construct the low pass filters or transfer functions that enabled the heater to simulate the simulated neutronics feedback. For this work, I used a single input single output (SISO) model. However, the SISO model can be extended to multiple input multiple output (MIMO) state space models in future work.

Secondly, I also needed to construct a real-time OPC-UA server for the CIET Heater. Now, given prior experience, I adapted the isothermally simulated CIET Digital Twin Server to simulate the Heater in real-time given a constant flowrate and a user-adjustable heater power input and heater inlet temperature.

Thirdly, I needed to construct a real-time OPC-UA client which is not only able to request and modify OPC-UA variables hosted on the server, but is also able to pass those values through simulated transfer functions and PID controllers in real-time. In fact, this third requirement is a combination of the first and second requirements. The key additional constraint for this third requirement is that both the OPC-UA server and the transfer function and PID simulator must be run in tandem in real-time.

Given these requirements, let us consider how to best to obtain these capabilities.

Programming Language Choice for Writing the Transfer Function Simulator

For the OPC-UA client, it must be designed such that it is able to obtain information from the OPC-UA server in real-time, pass it through various low pass filters or transfer functions, and translate it into a suitable power signal for the heater.

I had thought initially to use Labview given that CIET was using Labview in the past [Poresky et al., 2022]. Additionally, Labview has a Control Design and Simulation module [Cansalar, Maviş, and Kasnakoğlu, 2015] which is able to simulate transfer functions in real-time such that these transfer functions are able to respond to real-time user input. This has worked well in literature for small control loops [Cansalar, Maviş, and Kasnakoğlu, 2015]. Moreover, Labview has an OPC-UA client module which is able to request and modify OPC-

UA server variables in real-time. Now, at the time of writing, I did not have access to the OPC-UA module. However, Labview also has a Python module which allows me to code an OPC-UA client using the Python implementation. I could then build an interface between the OPC-UA client and Python code and have it communicate with Labview in real-time.

I found this process, however, to be extremely cumbersome because the Python module was built to simulate short functions or scripts. What I needed instead was for the Python code to run the OPC-UA client asynchronously while Labview was also running. Working this out proved to be cumbersome. A second roadblock I faced was latency. Every interface between Labview and the Python OPC-UA client seemed to have a noticeable performance cost. This is because every request to read an OPC-UA variable on the server in real-time took about 20 to 30 ms or even 300 ms. This time taken was too long, and therefore, I was not inclined to use this Python-Labview interface. Moreover, I wanted the Control Design and Simulation loops to run asynchronously with Python server code and share variables. Even getting this started was not trivial. Now, I could use other programs such as Simulink within Matlab [Cansalar, Maviş, and Kasnakoğlu, 2015]. However, I was unfamiliar with getting Simulink in Matlab to communicate with the OPC-UA client within Matlab. Therefore, even the prospect of Matlab and Simulink was not desirable for me. Finally, the closed source nature of Labview's and Matlab's object code further deterred me from using proprietary software. This is because I could not modify my code for follow up work if I wanted to in future. This is because my access (or lack thereof) to Labview was highly dependent on where I worked in future. Moreover, any research I performed using Labview could only be repeated by researchers with access to Labview. Since I already had to expend significant effort to get a Python OPC-UA client working with closed source Labview code, I thought that the effort was better spent on developing Free and Open Source (FOSS) code. Therefore, I wanted to shift to a FOSS version of OPC-UA server and client for the digital twin as well as the transfer function and PID simulation module. This effort to move to FOSS software was very much in line with work done in ARCO-CIET in 2021 where the OPC-UA server and client moved away from Labview only to one that was more Open Source like so that "ARCO- CIET can generate lessons and offer use cases that are relevant and actionable for many advanced reactor technologies" [Poresky et al., 2022]. Given that this was the case for ARCO-CIET, it would be natural to move to FOSS software for the Digital Twins of ARCO-CIET as well, or at least its heater.

Given all these considerations, I decided that this effort was better spent writing a Rust OPC-UA client with a graphical user interface (GUI) as well as a real-time control and simulation module which could run asynchronously with the OPC-UA client and GUI app. This would be a challenge since GUI programming in Rust was unfamiliar territory to me. However, as I was already familiar with some of the programming constructs used in Rust when developing the thermal hydraulics library, this barrier to entry was significantly lower to me. The biggest advantage of developing a GUI OPC-UA server, client and process control simulator in Rust would be customisability of the code due to it being FOSS.

Since I decided upon a FOSS approach for programming the transfer function simulator, PID controller and the OPC-UA server and GUI client. I will outline the theory of how

I write each of these components. Additionally, code verification, which tests if the code is functioning correctly [Avramova and Ivanov, 2010], is important for the newly written transfer function simulator and PID controller. Hence, some code to code verification results will also be presented. The validation part, which tests if the simulated transfer function replicates behaviour of experimental data [Avramova and Ivanov, 2010], is not done since the purpose of the transfer function simulator is to produce real-time output for transfer functions rather than experimental data.

Transfer Function Simulator Construction

To write a real-time FOSS transfer function simulator, I needed to consider the algorithms to be used for simulating transfer functions in real-time. This means the user should be able to give the transfer function a real-time varying input, and the transfer function should give a real-time varying output. Now, the user is not inclined to give a mathematically symmetrical or elegant input. The user should be free to give any input he or she wants in real-time. This is best modelled by a series of Heaviside functions or step functions with various amplitudes.

Therefore, the transfer function must be able to receive a series of Heaviside or step inputs of any amplitude and produce the appropriate output. Now, for a single Heaviside function, we should be able to produce the equivalent output in the time domain. However, for multiple Heaviside functions, we can just use the principle of superposition for linear time invariant (LTI) systems so that the output is the sum of several Heaviside functions at different times. To illustrate how this is done, I will first go through some derivations for first order systems, and extend it to second order systems. For simplicity, I will also limit my discussion to analog transfer function controllers using Laplace Transforms rather than discrete time digital controllers using the "z-transform".

First Order Plus Dead Time Example

FOPDT Transfer Functions in Standard Form To illustrate how one can program a transfer function simulator, let us begin with a simple example of a first-order plus dead time (FOPDT) transfer function with gain K_p, process time τ_p and dead-time τ_d:

$$G_{1st\ order}(s) = \exp(-\tau_d s)\frac{K_p}{\tau_p s + 1} \tag{6.8}$$

This class of transfer functions is important because they serve as the basis not just for first order transfer functions, but filters and derivative controllers as well.

Now, if we were to subject the FOPDT system to one step response of amplitude a_0 at time t_0, the time-domain expression of the output $y(t)$ with dead time, the expression after t_0 is:

$$y(t) = a_0 K_p \; u(t - t_0 - t_d) \left[1 - \exp\left(-\frac{t - t_0 - t_d}{\tau_p}\right)\right] \tag{6.9}$$

Of course, Equation 6.9 is valid for all values of t as its value of $y(t)$ at values prior to $t_0 + t_d$ are rightly zero. Now, suppose at t_1 the user gives another step input with amplitude a_1. The time domain response should be:

$$\begin{aligned} y(t) = a_0 K_p \; u(t - t_0 - t_d) \left[1 - \exp\left(-\frac{t - t_0 - t_d}{\tau_p}\right)\right] \\ + a_1 K_p \; u(t - t_1 - t_d) \left[1 - \exp\left(-\frac{t - t_1 - t_d}{\tau_p}\right)\right] \end{aligned}$$

Now, suppose the user keeps adding step inputs, it is easy to see that the equation keeps growing longer and longer. If one unit of computer memory was allocated to account for each user input, we would see that memory would continuously be allocated and never deallocated. Thus, the required computer memory required to simulate this transfer function increases over time. When memory is continuously allocated by a program and not deallocated, we call this a "memory leak" [Xie and Aiken, 2005]. Such memory leaks are detrimental to performance and can even cause system crashes if left unchecked [Xie and Aiken, 2005]. Obviously, this is not desirable. To prevent this, we must find a way to deallocate memory. Thankfully, these exponentials do decay away, and after some time, they practically reach a steady state. After that time, there is no more use in calculating the exponentially decaying function. To quantify when the equation reaches steady state, I specified that $\exp\left(-\frac{t-t_1-t_d}{\tau_p}\right)$ should be on the order of one part per billion or 10^{-9}. Now, this is because for FHRs, the temperature ranges are on the order of 1000K at most. The thermocouple uncertainty is at least $\pm 0.5°C$ as seen in CIET, though for thermocouples with higher temperature tolerances, the uncertainties tend to be larger. For this, the uncertainty is on the order of 10^{-4}. Deviations smaller than this tend not to be noticeable. I chose 10^{-9} out of some degree of conservatism. For this, $\exp(-20)$ is about $2 * 10^{-9}$. Therefore, after this time, the transfer functions are deemed to have reached steady state, and I can deallocate memory to prevent memory leaks.

Let us illustrate how this works supposing that the first input has reached steady state:

$$y(t) = a_0 K_p + a_1 K_p \; u(t - t_1 - t_d) \left[1 - \exp\left(-\frac{t - t_1 - t_d}{\tau_p}\right)\right]$$

I would then define an offset value C_{offset} in my program and add $a_0 K_p$ to this offset value:

$$y(t) = C_{offset} + a_1 K_p \; u(t - t_1 - t_d) \left[1 - \exp\left(-\frac{t - t_1 - t_d}{\tau_p}\right)\right]$$

Where:

$$C_{offset} = a_o K_p$$

Now if the second input reached steady state, I would define:

$$C_{offset} = a_o K_p + a_1 K_p$$

This is so that $y(t)$ just remains a constant:

$$y(t) = C_{offset}$$

Now, only one unit of memory is used to represent C_{offset} no matter the value of C_{offset}. Thus, after the exponential values reach steady state, I effectively set a new value of C_{offset} and deallocate memory used for the exponential function. In this manner, I solve my memory leak issue, and this would allow the computational demands not need to grow exponentially with time. Now, this manner of preventing memory leaks works only for stable transfer functions where we observe exponential decays. For unstable transfer or undamped transfer functions where these inputs do not decay away, we may have to find some other method of ensuring that we have no memory leaks.

Now, this manner of programming a stable FOPDT transfer function is relatively intuitive. Moreover, since the schemes are analytically integrated, we need not worry about numerical integration or stiff problems causing numerical instability. We also need not worry about approximation errors from numerical integration other than the steady state approximation we made earlier.

FOPDT Transfer Functions with Zeroes in Non Standard Form Now of course, even for first order stable systems, we may not merely encounter systems with one pole and no zeroes, it is also common to come across first order systems in non standard form where zeroes exist. Suppose that a first order physically realisable transfer function comes in the form:

$$G(s) = \frac{a_1 s + b_1}{a_2 s + b_2} \tag{6.10}$$

We should then be able to deal with these transfer functions. Of course, while the user could in theory convert this transfer function to a superposition of two FOPDT simulations, it would be more convenient for us to execute this process programmatically.

To convert equation 6.10 into standard form, we shall need to obtain K_p and τ_p from these coefficients a_1, a_2, b_1 and b_2. If we were to ignore the zero, this is quite straightforward:

$$K_p = \frac{b_1}{b_2} \tag{6.11}$$

And

$$\tau_p = \frac{a_2}{b_2} \tag{6.12}$$

In SI units, τ_p should be in time units (s) and a_2 should also be in time units (s). Assuming b_2 is dimensionless, we see unit consistency. Likewise, K_p is in units of b_1. Now, with K_p and τ_p defined, we can rewrite equation 6.10, separating the simple FOPDT component away from the filtered derivative (s) term with the coefficient $\frac{a_1}{b_2}$:

$$G(s) = \frac{\frac{a_1}{b_2}s + K_p}{\tau_p s + 1} = \frac{K_p}{\tau_p s + 1} + \frac{\frac{a_1}{b_2}s}{\tau_p s + 1}$$

Now, thankfully for us, the filtered derivative term can also be converted into a first order transfer function. Therefore, we can essentially reuse code meant for FOPDT functions and adapt it for generic FOPDT transfer functions with one zero.

$$\begin{aligned} G(s) &= \frac{K_p}{\tau_p s + 1} + \frac{\frac{a_1}{b_2}s}{\tau_p s + 1} \\ &= \frac{K_p}{\tau_p s + 1} + \frac{a_1}{b_2 \tau_p}\frac{\tau_p s}{\tau_p s + 1} \\ &= \frac{K_p}{\tau_p s + 1} + \frac{a_1}{b_2 \tau_p}\left[\frac{\tau_p s + 1 - 1}{\tau_p s + 1}\right] \\ &= \frac{K_p}{\tau_p s + 1} + \frac{a_1}{b_2 \tau_p}\left[1 - \frac{1}{\tau_p s + 1}\right] \end{aligned}$$

Again doing some SI unit checks, b_2 is dimensionless, a_1 and τ_p are in units of s assuming that $G(s)$ is dimensionless.

Code to Code Verification Now that we have derived the FOPDT transfer functions, we need to perform code to code verification. To do so, I used a stable FOPDT transfer function and simulated it in the FOSS software Scilab Xcos [Merzlikina and Prochina, 2020]. The transfer function is shown in Equation 6.13:

$$G(s) = \exp(-2s)\frac{5 - 2s}{4s + 2} \tag{6.13}$$

I chose Scilab because it is licensed under GNU General Public License (GPL) v2.0 and it is also because it was, at the time of writing, a suitable FOSS alternative to Matlab [Mikac, Logožar, and Horvatić, 2022]. Now, the block digram of the Scilab simulation is presented in Figure 6.3:

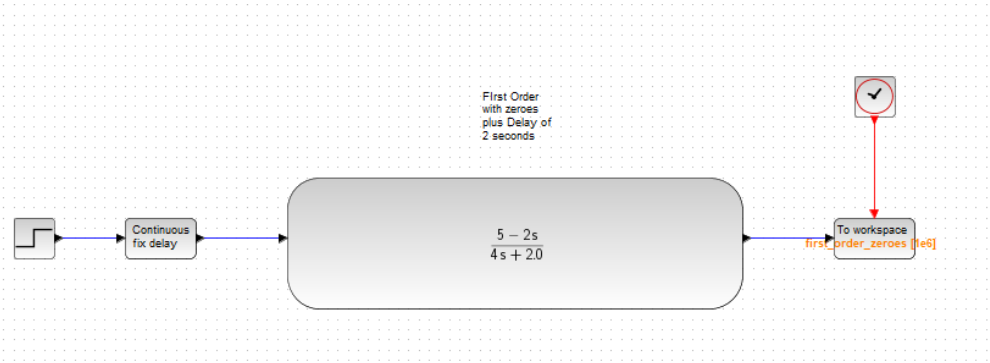


Figure 6.3: fopdt system $[g(s) = \exp(-2s)\frac{5-2s}{4s+2}]$ block diagram in scilab xcos

For this transfer function, I simulated an input of 9 at $t = 0$ for both Scilab and my Transfer Function Library written in Rust. This was done with a time step of $t = 0.1s$ for my Transfer Function Library and done using an implicit auto-timestepped Runge-Kutta 45 method in Scilab. The results are plotted in Figure 6.4:

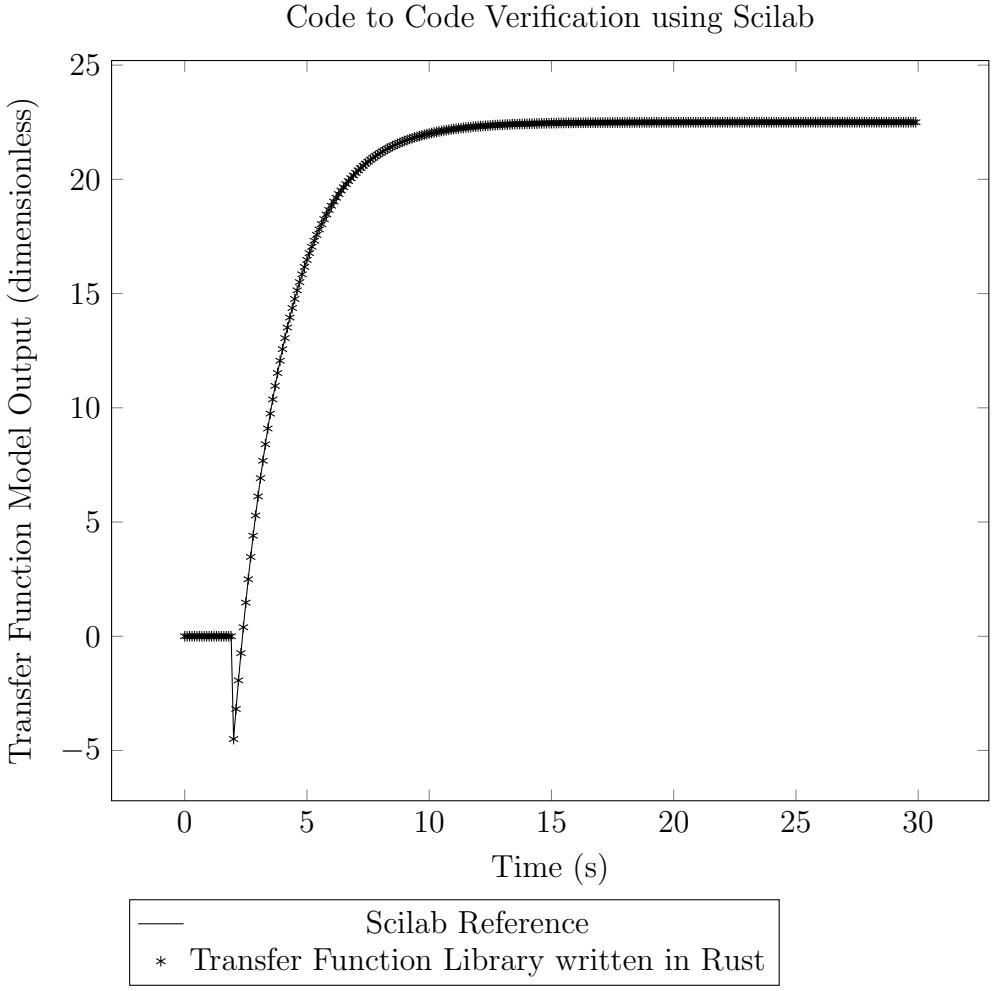


Figure 6.4: Step Input of 9 units (dimensionless) given at $t = 0s$

Figure 6.4 seems to show good agreement between Scilab and the Transfer Function Library written in Rust. Of course, to verify that the results match within satisfactory tolerances, it is useful to look into the residual plots.

To calculate these, can we look at the fractional deviation or error (ε) between the Scilab outputs $y_{scilab}(t)$ and my library outputs written in Rust $y_{rust-library}(t)$. This was calculated using Equation 6.14:

$$\varepsilon_{fractional} = \left| \frac{y_{scilab}(t) - y_{rust\ library}(t)}{y_{scilab}(t)} \right| = \left| \frac{\varepsilon_{absolute}}{y_{scilab}(t)} \right| \tag{6.14}$$

The maximum value of $\varepsilon_{fractional}$ in this context was 8.2×10^{-5}. It may seem small compared to a 1% error, but we still need to put this relative error in context. To do so, I compare $\varepsilon_{absolute}$ against a typical truncation error. This is because the truncation error should give us a suitable scale of how far numerical errors should deviate from their analytical

solution. Since we are using Runge-Kutta 45 (RK-45) solvers in Scilab, it would be natural to use an appropriate Runge-Kutta truncation error estimate. Such error estimation methods are well known in literature [Butcher and Johnston, 1993; Tsitouras and Papakostas, 1999]. However, I have opted for something much more crude given that the error is sufficiently small that it is smaller than typical uncertainties in experiments. For example, in FHR context, we deal with ΔT of around $100°C$ and thermocouple measurement uncertainty of at least $\pm 0.5°C$. This translates to a rough uncertainty percentage of around $\pm 0.5\%$. Thus, any numerical error below this value won't be distinguishable from thermocouple uncertainty. For our purposes, this transfer function simulation is good enough.

Of course, it will would still be useful to measure if the simulation performs well enough in general. To do that, the residuals should ideally be compared to the RK-45 truncation error. Given that this is tedious and that the results are already good enough for simulating transfer functions, I thought that using a crude estimate the typical truncation error would suffice. This crude truncation error estimate comes from the Euler method. For this, we consider a Taylor Series expansion of a function $f(x)$. We can write this as [Perry and Green, 2015]:

$$f(x + \Delta h) \approx f(x) + f'(x)\Delta h + \frac{1}{2}f''(x)(\Delta h)^2 + \cdots$$

Where $f'(x)$ and $f''(x)$ are the first and second order derivatives respectively, and Δh is the stepsize. In terms of time t and time step Δt:

$$f(t + \Delta t) \approx f(t) + f'(t)\Delta t + \frac{1}{2}f''(t)(\Delta t)^2 + \cdots$$

Now, a lower bound for the truncation error term should be:

$$\frac{1}{2}f''(t)(\Delta t)^2$$

Now, for $f''(t)$, we can estimate its value using the central difference approximation [Perry and Green, 2015]:

$$f''(t) = \frac{f(t + \Delta t) - 2f(t) + f(t - \Delta t)}{(\Delta t)^2}$$

Thus, we are left with Equation 6.15 as a crude estimation for the truncation error:

$$\frac{1}{2}\left[f(t + \Delta t) - 2f(t) + f(t - \Delta t)\right] \tag{6.15}$$

Of course, there may be discontinuities where it is inconvenient to use Equation 6.15. However, as I am only doing a rough estimate of the truncation error, I'm okay with using forward and backward estimation methods for truncation error as well in Equation 6.16 and Equation 6.17:

$$\frac{1}{2}\left[f(t+2\Delta t) - 2f(t+\Delta t) + f(t)\right] \tag{6.16}$$

$$\frac{1}{2}\left[f(t-2\Delta t) - 2f(t-\Delta t) + f(t)\right] \tag{6.17}$$

Also, since I am only concerned with the order of magnitude of the error, I also neglected the $\frac{1}{2}$ term when calculating the error bars. The timestep $\Delta t = 0.1s$, and that was used to estimate the typical truncation error. Given this, I plot the residuals in Figure 6.5:

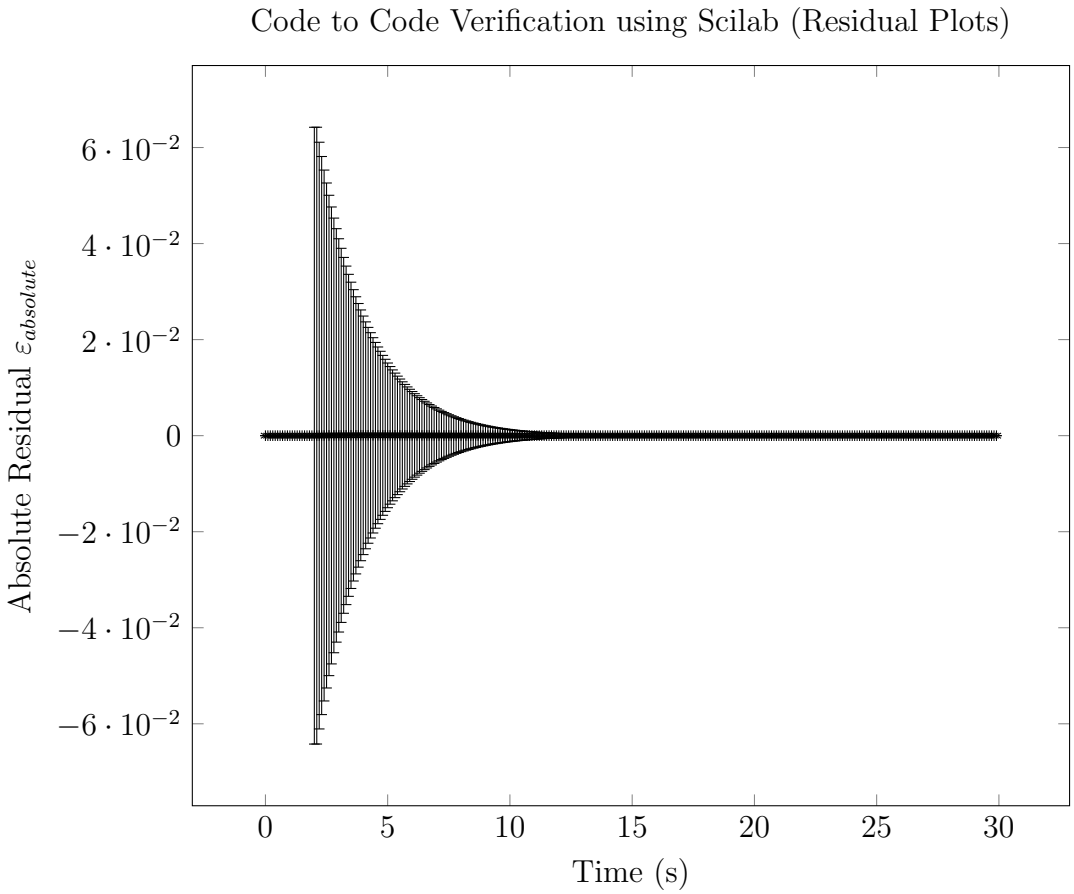


Figure 6.5: Step Input of 9 units (dimensionless) given at $t = 0s$

In Figure 6.5, the residuals all seem to be smaller than the crudely estimated truncation error. For the purposes of this work, this is good enough.

Second Order Systems

The FOPDT algorithm described earlier is not necessarily applicable beyond the simplest first order systems. For more complicated transfer functions, the analytical integration process for

each transfer function becomes non trivial. Of course, for most transfer functions of interest, we can split the transfer function up into several first and second order transfer functions connected in series and then perform inverse Laplace Transforms on them. However, I won't do this for an nth order transfer function in this work as this would take a fair amount of development time. For now, I am only interested in simulating the transfer function of the SNF controller (or at least its first iteration):

$$G_{CIET\ heater\ inletT\ to\ heaterPower}(s)\left(\frac{watt}{^{\circ}C}\right) = 340.136\frac{(s+5.5)(s+8.5)}{\tau_{filter}s+1}\frac{(178.49s-0.33013)}{1.4999*10^{6}s^{2}+1185.4s+1}$$

Of course, we need to ensure that the controller causes the heater to behave correctly. Therefore, we also need to simulate the expected outlet temperature of the heater in real-time. The inlet temperature to outlet temperature transfer function for GeN-Foam scaled to CIET is:

$$G(s) = \frac{0.000119s - 2.201*10^{-7}}{s^{2}+0.0007903s+6.667*10^{-7}}$$

We can see that it is a second-order transfer function. As mentioned previously, it is a stable transfer function with complex roots. For both transfer functions, we can see that there is at least a second order transfer function involved. For the first controller iteration, it is technically a third order transfer function, but it can be modelled by a first order transfer function and a second order transfer function combined in series. Given that we need to simulate second order transfer functions either way, we will at least need to discuss a theoretical framework for simulating second order transfer functions in real-time. These second order transfer functions can include underdamped, overdamped or critically damped systems. Of course, we also have unstable systems where the system response grows exponentially. However, when building SNF controllers, we generally want to avoid unstable controller designs. Hence, for now, we will not develop code which simulates an unstable transfer function. This is out of scope for this work.

Now, to simulate a stable second order transfer function in real-time, it is important to obtain the response of this transfer function to a step input. We shall then apply a similar algorithm to the FOPDT algorithm discussed earlier in order to simulate this transfer function in real-time. The output $Y(s)$ in the frequency domain to a step input $1/s$ [Irvine, 2011; Perry and Green, 2015] is:

$$Y(s) = \frac{1}{s}\frac{0.000119s - 2.201*10^{-7}}{s^{2}+0.0007903s+6.667*10^{-7}} \tag{6.18}$$

$$Y(s) = \frac{0.000119}{s^2 + 0.0007903s + 6.667 * 10^{-7}} - \frac{1}{s}\frac{2.201 * 10^{-7}}{s^2 + 0.0007903s + 6.667 * 10^{-7}} \tag{6.19}$$

The first term here has a convenient inverse Laplace transform in the form $\exp(-at) * \sin(\omega t)$ [Irvine, 2011; Perry and Green, 2015]:

$$\mathcal{L}\left\{\exp(-at)\sin(\omega t)\right\} = \frac{\omega}{(s+a)^2 + \omega^2}$$

We can see that having zeroes in the numerator of transfer function would generally induce more decaying sinusoids in underdamped systems. Now, should there be an s^2 term in the numerator of the second order system, we may have to involve cosines in calculating the step response. The Laplace Transforms relevant for this are [Irvine, 2011; Perry and Green, 2015]:

$$\mathcal{L}\left\{\exp(-at)\cos(\omega t)\right\} = \frac{s+a}{(s+a)^2 + \omega^2}$$

Since $\cos(\omega_d t)$ is of $\mathcal{O}(1)$, we can simply use the same approximation as FOPDT in order to ascertain if it has reached steady state. That is to say, for a function $\exp(-at)\cos(\omega t)$, we can say that the function has reached steady state at about $at = 20$. The latter term is a prototypical 2nd order lowpass step response. Again, I used a similar algorithm to check if the solution has reached steady state can be used to ascertain if responses have reached steady state. These two components can be calculated separately and summed together. To calculate the latter term in Equation 6.19, let us discuss the step response of second order transfer functions in general.

Classifications based on Damping Factor The transfer functions for second order systems can be written in standard form as [Seborg et al., 2016]:

$$Y(s) = \frac{K_p}{\tau_p^2 s^2 + 2\zeta\tau_p s + 1} \tag{6.20}$$

Where τ_p the process timescale. While τ_p is defined also for the first order system, one should note that τ_p in a second order system is not equivalent to τ_p in a first order system. ζ is a dimensionless damping factor and K_p is the steady state process gain. For second order systems, there are well known analytical solutions for a systems response to a step input. For an overdamped system, the analytical solution for a step response of amplitude a_0 at time $t = 0$ can be written as [Seborg et al., 2016]:

$$y(t) = a_0 K_p \left\{1 - \exp\left(-\zeta\frac{t}{\tau_p}\right)\left[\cosh\left(\frac{\sqrt{\zeta^2-1}}{\tau_p}t\right) + \frac{\zeta}{\sqrt{\zeta^2-1}}\sinh\left(\frac{\sqrt{\zeta^2-1}}{\tau_p}t\right)\right]\right\} \tag{6.21}$$

Where again, ζ is the damping factor, necessarily greater than 1 in overdamped system, t is elapsed time, τ_p is process time and K_p is process gain. For underdamped systems, the hyperbolic cosines and sines are replaced by their non-hyperbolic counterparts, in addition to some sign changes with ζ [Seborg et al., 2016][1]:

$$y(t) = a_0 K_p \left\{ 1 - \exp\left(-\zeta \frac{t}{\tau_p}\right) \left[\cos\left(\frac{\sqrt{1-\zeta^2}}{\tau_p} t \right) + \frac{\zeta}{\sqrt{1-\zeta^2}} \sin\left(\frac{\sqrt{1-\zeta^2}}{\tau_p} t \right) \right] \right\} \quad (6.22)$$

Where $0 \leq \zeta < 1$. For critically damped systems, $\zeta = 1$ and [Seborg et al., 2016]:

$$y(t) = a_0 K_p \left\{ 1 - \left[1 + \frac{t}{\tau_p} \right] \exp\left(-\frac{t}{\tau_p} \right) \right\} \quad (6.23)$$

Of course, there are many more complex systems than second order systems. The user can refer to plenty of Laplace Transform tables[Irvine, 2011] in order to see the time domain response of the system to a step change.

Steady State Values of Decaying Exponential Sinusoids The time domain response can be readily programmed into the client side of Rust with similar algorithms to determine when the responses reach steady state. For underdamped systems, if $\zeta \frac{t}{\tau_p} > 20.0$, we can consider that the exponent $\exp(-\zeta \frac{t}{\tau_p})$ has essentially reached steady state.

For the critically damped system, however, the time taken for the system to reach steady state is slightly different as the exponential term changes:

$$-\left[1 + \frac{t}{\tau_p} \right] \exp\left(-\frac{t}{\tau_p} \right)$$

Thus we have a term in the form $x \exp(-x)$ that needs to decay away. For this, we can perform simple calculations to check when this will reach on the order of 10^{-9}. For this, note that a suitable value for x is about 23 or 24. At $x = 23$, $x \exp(-x) = 2.36 \times 10^{-9}$. This is of a suitable magnitude for it to be considered negligible. Hence, a slightly larger x (23 as opposed to 20) is required to for the exponential terms to reach 10^{-9} than in the FOPDT system. Hence, for second order systems in general, we can just set it such that $\frac{t}{\tau_p} = 23$ in order for the system to reach steady state.

For overdamped systems, it is difficult to see when steady state is reached due to the presence of hyperbolic sines and cosines. Hence, we will need to expand out the hyperbolic sines and cosines in order to inspect when the transient components decay away Again, we start with:

$$y(t) = a_0 K_p \left\{ 1 - \exp\left(-\zeta \frac{t}{\tau_p}\right) \left[\cosh\left(\frac{\sqrt{\zeta^2-1}}{\tau_p} t \right) + \frac{\zeta}{\sqrt{\zeta^2-1}} \sinh\left(\frac{\sqrt{\zeta^2-1}}{\tau_p} t \right) \right] \right\}$$

[1] Process Dynamics and Control, 2nd Edition Chapter and Section: 5.4.1, Page 117

We can now examine the hyperbolic sines and cosine terms:

$$\begin{aligned}
&\exp\left(-\zeta\frac{t}{\tau_p}\right)\cosh\left(\frac{\sqrt{\zeta^2-1}}{\tau_p}\right)\\
&=\exp\left(-\zeta\frac{t}{\tau_p}\right)\left(0.5\exp\left(\frac{\sqrt{\zeta^2-1}}{\tau_p}t\right)+0.5\exp\left(-\frac{\sqrt{\zeta^2-1}}{\tau_p}t\right)\right)\\
&=0.5\exp\left(\frac{\sqrt{\zeta^2-1}}{\tau_p}t-\zeta\frac{t}{\tau_p}\right)+0.5\exp\left(-\frac{\sqrt{\zeta^2-1}}{\tau_p}t-\zeta\frac{t}{\tau_p}\right)\\
&=0.5\exp\left(\frac{t}{\tau_p}\left[\sqrt{\zeta^2-1}-\zeta\right]\right)+0.5\exp\left(-\frac{t}{\tau_p}\left[\sqrt{\zeta^2-1}+\zeta\right]\right)
\end{aligned}$$

For the hyperbolic cosine terms, we see that $\frac{t}{\tau_p}\left[\sqrt{\zeta^2-1}-\zeta\right]$ and $-\frac{t}{\tau_p}\left[\sqrt{\zeta^2-1}+\zeta\right]$ are both to consider to see if an exponential term decays sufficiently for us to assume it reaches steady state. Now, we need not worry about an unbound response because $\frac{t}{\tau_p}\left[\sqrt{\zeta^2-1}-\zeta\right]<0$. This in turn is is because $\sqrt{\zeta^2-1}<\sqrt{\zeta^2}$. With that out of the way, we can now consider the two time scales and see which mode decays more slowly. Firstly, we consider that:

$$|\sqrt{\zeta^2-1}+\zeta|>|\sqrt{\zeta^2-1}-\zeta|$$

Given this, we know that the more slowly decaying mode is based on $(\sqrt{\zeta^2-1}-\zeta)\frac{t}{\tau_p}$. This is because if we consider two time decaying modes, $\exp(-20t)$ and $\exp(-0.5t)$, the $\exp(-20t)$ mode decays faster than $\exp(-0.5t)$. To check if the exponential term reaches more or less steady state, we need only consider $\exp(-0.5t)$ to obtain a suitable time scale. Therefore, we need only consider $(\sqrt{\zeta^2-1}-\zeta)\frac{t}{\tau_p}$. The time for overdamped systems where we consider it to have reached steady state is obtained from:

$$(\zeta-\sqrt{\zeta^2-1})\frac{t}{\tau_p}>20 \tag{6.24}$$

Now of course, it is trivial to add both decay modes in and let the program check for us if both modes have decayed out. We only need to add the condition that:

$$(\zeta+\sqrt{\zeta^2-1})\frac{t}{\tau_p}>20 \tag{6.25}$$

I do not perceive this to be excessively expensive computationally, and will add it in first out out of an abundance of caution.

To obtain more decay modes, we can do a similar analysis for the hyperbolic sine, but it would still yield the same result in terms of which time scales are important for determining if we have reached steady state.

Second Order Systems in Non Standard Form Now, second order systems come in several shapes and forms. And it can get quite cumbersome to manually obtain ζ and τ_p for a given polynomial. Therefore, it would be good to programmatically obtain ζ, τ_p and the gain from the second order system. This can get inconvenient if done repeatedly.

Suppose we had a polynomial transfer function:

$$G(s) = \frac{a_1 s^2 + b_1 s + c_1}{a_2 s^2 + b_2 s + c_2} \tag{6.26}$$

The step response would be:

$$\begin{aligned} Y(s) &= \frac{1}{s} \frac{a_1 s^2 + b_1 s + c_1}{a_2 s^2 + b_2 s + c_2} \\ &= \frac{a_1 s + b_1 + c_1/s}{a_2 s^2 + b_2 s + c_2} \\ &= \frac{a_1 s + b_1}{a_2 s^2 + b_2 s + c_2} + \frac{1}{s} \frac{c_1}{a_2 s^2 + b_2 s + c_2} \end{aligned}$$

We can see that the $a_1 s + b_1$ term essentially has an inverse Laplace Transform which converts it into a decaying exponential term with a steady state value of zero. Only the c_1 term has a persistent steady state change after an extended amount of time. I refer to this as a "Second Order Non Zero Steady State Mode". Let us first consider how to deal with Second Order Non Zero Steady State Modes. Whereas the $a_1 s + b_1$ terms essentially decay to zero over time. Therefore, I refer to them as the "Second Order Decaying Modes" for Stable Second Order Transfer Functions.

Simulation of Second Order Non Zero Steady State Modes To simulate $Y(s)$, we must simulate its Second Order Non Zero Steady State Modes. For this purpose, the first order of business would be to obtain τ_p and ζ. The denominator should be in the form:

$$\tau_p^2 s^2 + 2\zeta\tau_p s + 1$$

Just by comparing coefficients, we can find τ_p:

$$\tau_p = \sqrt{\frac{a_2}{c_2}}$$

τ_p should have units of time or (seconds in SI). We see that if we use SI units, a_2 has units of s^2 and c_2 is dimensionless. Therefore the units are correct.

For ζ:

$$
\begin{aligned}
2\zeta\tau_p &= \frac{b_2}{c_2} \\
\zeta &= 0.5\frac{1}{\tau_p}\frac{b_2}{c_2} \\
\zeta &= 0.5\frac{1}{\sqrt{\frac{a_2}{c_2}}}\frac{b_2}{c_2} \\
\zeta &= \frac{0.5b_2}{\sqrt{a_2c_2}}
\end{aligned}
$$

Now, ζ is a ratio, and so it should be dimensionless, we see that $\sqrt{a_2c_2}$ is units of $\sqrt{s^2}$, and b_2 is in units of s if we use SI units. So ζ is indeed dimensionless.

Now, the steady state gain, K_p is taken by limiting $s \to 0$:

$$K_p = \frac{c_1}{c_2} \tag{6.27}$$

K_p should be in units of c_1. Since c_2 is dimensionless, the units are consistent.

If the damping factor $\zeta \leq 0$, we will tentatively return an error in the Transfer Function library because for this dissertation, we only consider $\zeta > 0$.

Second Order Decaying Modes Next, we must convert the decaying $a_1s + b_1$ term into a suitable form. Let's first convert the denominator into a suitable form:

$$
\begin{aligned}
s^2 + \frac{b_2}{a_2}s + \frac{c_2}{a_2} &= (s+\lambda)^2 + \omega^2 \\
s^2 + \frac{b_2}{a_2}s + \frac{c_2}{a_2} &= s^2 + 2\lambda s + \lambda^2 + \omega^2
\end{aligned}
$$

Now, $\lambda > 0$ because $\zeta > 0$. Hence we obtain:

$$\lambda = 0.5\frac{b_2}{a_2} \tag{6.28}$$

Now, λ, which is a decay constant of sorts like those seen in radioactive decay, should be in units of frequency, or s^{-1} in SI units. If we consider SI units, b_2 is also in units of s and a_2 is in units of s^2. Therefore the units check out.

And consequently, we must have $b_2 > 0$ for a stable system. This agrees with our earlier observations. Also $a_2 > 0$ because $\lambda > 0$. Given this, $c_2 > 0$ also. Now, not all systems are

underdamped. In fact, only $0 < \zeta < 1$ constitutes an underdamped system. To ascertain if we will have oscillations, we must look at the discriminant $b_2^2 - 4a_2c_2$.

$$\begin{cases} 0 < \zeta < 1 & \text{if } b_2^2 - 4a_2c_2 < 0 \\ \zeta = 1 & \text{if } b_2^2 - 4a_2c_2 = 0 \\ \zeta > 1 & \text{if } b_2^2 - 4a_2c_2 > 0 \end{cases} \tag{6.29}$$

Underdamped Case Now, only in the case of underdamping, where $0 < \zeta < 1$ we can calculate the decaying oscillation frequencies:

$$\begin{aligned} \frac{c_2}{a_2} &= \lambda^2 + \omega^2 \\ \omega &= \sqrt{\frac{c_2}{a_2} - \lambda^2} \\ \omega &= \sqrt{\frac{c_2}{a_2} - 0.25\frac{b_2^2}{a_2^2}} \end{aligned}$$

Now, ω, like λ is a frequency unit, or s^{-1} in SI. If use use SI units, we see that c_2 is dimensionless while a_2 has units of s^2. Therefore this term is consistent. Likewise b_2^2 has units of s^2 and a_2^2 has units of s^4. Therefore, the units check out.

Now suppose λ and ω are calculated, we shall need to use it to ascertain the first $a_1 s + b$. Ideally, we would want to massage the terms in such a way that the inverse Laplace Transforms are trivial:

$$\begin{aligned} \frac{a_1 s + b_1}{a_2 s^2 + b_2 s + c_2} &= \frac{a_1}{a_2}\frac{s + \frac{b_1}{a_1}}{(s+\lambda)^2 + \omega^2} \\ &= \frac{a_1}{a_2}\frac{s + \lambda - \lambda + \frac{b_1}{a_1}}{(s+\lambda)^2 + \omega^2} \\ &= \frac{a_1}{a_2}\frac{s + \lambda}{(s+\lambda)^2 + \omega^2} - \frac{a_1}{a_2}\frac{\lambda - \frac{b_1}{a_1}}{(s+\lambda)^2 + \omega^2} \\ &= \frac{a_1}{a_2}\frac{s + \lambda}{(s+\lambda)^2 + \omega^2} - \frac{a_1(\lambda - \frac{b_1}{a_1})}{a_2\omega}\frac{\omega}{(s+\lambda)^2 + \omega^2} \end{aligned}$$

Now, we take the inverse Laplace Transforms of the expression,

$$\begin{aligned}\mathcal{L}^{-1}\left[\frac{a_1 s + b_1}{a_2 s^2 + b_2 s + c_2}\right] &= \mathcal{L}^{-1}\left[\frac{a_1}{a_2}\frac{s+\lambda}{(s+\lambda)^2+\omega^2} - \frac{a_1(\lambda - \frac{b_1}{a_1})}{a_2\omega}\frac{\omega}{(s+\lambda)^2+\omega^2}\right] \\ &= \frac{a_1}{a_2}\exp(-\lambda t)\cos(\omega t) - \frac{a_1(\lambda - \frac{b_1}{a_1})}{a_2\omega}\exp(-\lambda t)\sin(\omega t) \\ &= \frac{a_1}{a_2}\exp(-\lambda t)\cos(\omega t) - \frac{a_1\lambda}{a_2\omega}\exp(-\lambda t)\sin(\omega t) \\ &+ \frac{b_1}{a_2\omega}\exp(-\lambda t)\sin(\omega t)\end{aligned}$$

Now, we can clearly see that $\frac{b_1}{a_1}$ is of units $\frac{1}{s}$ and so it is the same as units of λ and ω. Therefore, both units are in units of $\frac{a_1}{a_2}$. Hence, there is unit consistency.

Thus, given the polynomial expressions, we should be able to obtain the time domain step responses the second order transfer functions with zeros.

Critically Damped Case In the critical damping case where $\zeta = 1$, $\omega = 0$, and $\lambda = 0.5\frac{b_2}{a_2}$, we have two repeated roots in the denominator. Then we shall have [Seborg et al., 2016] :

$$\mathcal{L}\left\{\frac{1}{(s+\lambda)^2}\right\} = t\exp(-\lambda t) \tag{6.30}$$

As mentioned earlier, we can consider the mode to have reached steady state at around $\lambda t > 23$ if we consider the term $\lambda t\exp(-\lambda t)$. This is a more stringent than the requirement for $\lambda t > 20$ where $\exp(-\lambda t)$ decays away. Nevertheless, based on the Laplace Transform, there remains a term dependent on $t\exp(-\lambda t)$ for which we cannot easily assign a threshold λt value and expect it to work for all cases. What we can do here is to apply a second condition in addition to $\lambda t > 23$. The base condition for this is where the exponential term decays to around 10^{-9} times of the original value or maximum value. The maximum value can be found by obtaining the stationary point:

$$\frac{d}{dt}t\exp(-\lambda t) = -\lambda t\exp(-\lambda t) + \exp(-\lambda t)$$

The relevant stationary (local maximum) point in this case is $\lambda t = 1$, where the function evaluates to $\frac{1}{e}$. Since this of of $\mathcal{O}(1)$, we can simply set the criteria to be $t\exp(-\lambda t) < 10^{-9}$ and $\lambda t > 1$ for the term to decay away to steady state of 0.

However, to calculate the second order time domain step response signals, we need to consider $a_1 s + b_1$ in the numerator.

$$\begin{aligned}
\mathcal{L}\left\{\frac{a_1 s + b_1}{a_2 s^2 + b_2 s + c_2}\right\} &= \frac{a_1}{a_2}\mathcal{L}\left\{\frac{s + \frac{b_1}{a_1}}{(s+\lambda)^2}\right\} \\
&= \frac{a_1}{a_2}\mathcal{L}\left\{\frac{s + \lambda - \lambda + \frac{b_1}{a_1}}{(s+\lambda)^2}\right\} \\
&= \frac{a_1}{a_2}\mathcal{L}\left\{\frac{s+\lambda}{(s+\lambda)^2} + \frac{\frac{b_1}{a_1} - \lambda}{(s+\lambda)^2}\right\} \\
&= \frac{a_1}{a_2}\mathcal{L}\left\{\frac{1}{s+\lambda} + \frac{\frac{b_1}{a_1} - \lambda}{(s+\lambda)^2}\right\} \\
&= \frac{a_1}{a_2}\mathcal{L}\left\{\frac{1}{s+\lambda} + \left(\frac{b_1}{a_1} - \lambda\right)\frac{1}{(s+\lambda)^2}\right\} \\
&= \frac{a_1}{a_2}\left\{\exp(-\lambda t) + \left(\frac{b_1}{a_1} - \lambda\right) t\exp(-\lambda t)\right\} \\
&= \frac{a_1}{a_2}\exp(-\lambda t) + \left(\frac{a_1}{a_2}\frac{b_1}{a_1} - \lambda\frac{a_1}{a_2}\right) t\exp(-\lambda t) \\
&= \frac{a_1}{a_2}\exp(-\lambda t) + \left(\frac{b_1}{a_2} - \lambda\frac{a_1}{a_2}\right) t\exp(-\lambda t)
\end{aligned}$$

Now, let us once again check the units using SI convention, noting that $\exp(-\lambda t)$ is dimensionless, both $\frac{b_1}{a_1}$ and λ are in units of $\frac{1}{s}$, and t is in s. We see that $\left(\frac{b_1}{a_1} - \lambda\right) t$ is also dimensionless. Therefore, the units check out. Based on this expression, we do see a term with $\lambda t \exp(-\lambda t)$. Therefore, $\lambda t > 23.0$ should be one of the criteria of consideration for decay to steady state. The next criteria is where $\frac{b_1}{a_1} t \exp(-\lambda t) < 10^{-9}$. This should not be too computationally expensive to compute. Therefore, I will not perform further numerical analysis to obtain a further simplified criteria for when this exponential term decays to steady state.

Overdamped Case Now for overdamped modes, we essentially have two real roots in the denominator. The formula for such is given by the classic quadratic root finding formula [Perry and Green, 2015]:

$$a_2 s^2 + b_2 s + c_2 = a_2(s+\alpha)(s+\beta) \tag{6.31}$$

Where:

$$\alpha = \frac{-b_2 + \sqrt{b_2^2 - 4a_2c_2}}{-2a_2}$$
$$\beta = \frac{-b_2 - \sqrt{b_2^2 - 4a_2c_2}}{-2a_2}$$

Therefore, if we were to go back to the Laplace Transforms:

$$\begin{aligned}\mathcal{L}\left\{\frac{a_1 s + b_1}{a_2(s+\alpha)(s+\beta)}\right\} &= \frac{a_1}{a_2}\mathcal{L}\left\{\frac{s + \frac{b_1}{a_1}}{(s+\alpha)(s+\beta)}\right\} \\ &= \frac{a_1}{a_2}\mathcal{L}\left\{\frac{A}{s+\alpha} + \frac{B}{s+\beta}\right\} \\ &= \frac{a_1}{a_2}A\exp(-\alpha t) + \frac{a_1}{a_2}B\exp(-\beta t)\end{aligned}$$

So far, the units seem to check out, because A and B should be of the same unit. Now we need to determine the coefficients A and B:

$$s + \frac{b_1}{a_1} = A(s+\beta) + B(s+\alpha)$$
$$\text{Using the Cover up Rule} \begin{cases} -\beta + \frac{b_1}{a_1} = B(-\beta + \alpha), & \text{if } s = -\beta \\ -\alpha + \frac{b_1}{a_1} = A(-\alpha + \beta), & \text{if } s = -\alpha \end{cases}$$

Now let's perform some unit checks in SI units again. We see that $A = \frac{-\alpha + \frac{b_1}{a_1}}{-\alpha+\beta}$, and $B = \frac{-\beta + \frac{b_1}{a_1}}{-\beta+\alpha}$. We note that the denominator is in units of α and β which are in s^{-1}. If the numerator of the transfer function is dimensionless, then b_1/a_1 is in units of s^{-1}. Here, we have unit consistency between A and B, both of which are dimensionless.

The final expression for overdamped systems is:

$$\mathcal{L}\left\{\frac{a_1 s + b_1}{a_2(s+\alpha)(s+\beta)}\right\} = \frac{a_1}{a_2}\left(\frac{-\alpha + \frac{b_1}{a_1}}{-\alpha+\beta}\right)\exp(-\alpha t) + \frac{a_1}{a_2}\left(\frac{-\beta + \frac{b_1}{a_1}}{-\beta+\alpha}\right)\exp(-\beta t)$$

Code to Code Verification Tests For code verification, we want to subject our code to some limited tests. For code to code verification, I will first obtain step tests for some

transfer functions using SciLab and Xcos [Campbell et al., 2010; Merzlikina and Prochina, 2020]. Then I will compare the results to that generated by the coded transfer function libraries in Rust. Firstly, the heater inlet temperature to heater outlet temperature transfer function was tested:

$$G_{arbitrary\ FHR\ scaled\ inletT\ to\ outletT}(s) = \frac{0.000119s - 2.201 * 10^{-7}}{s^2 + 0.0007903s + 6.667 * 10^{-7}}$$

This is a second order underdamped system with one zero. For this transfer function, a 9 K step input was given at $t = 0$. Data was generated using the Xcos control systems module within Scilab with timestep of $20s$ using a Runge-Kutta 4th Order Implicit Scheme. I used this to validate the transfer function libraries I built.

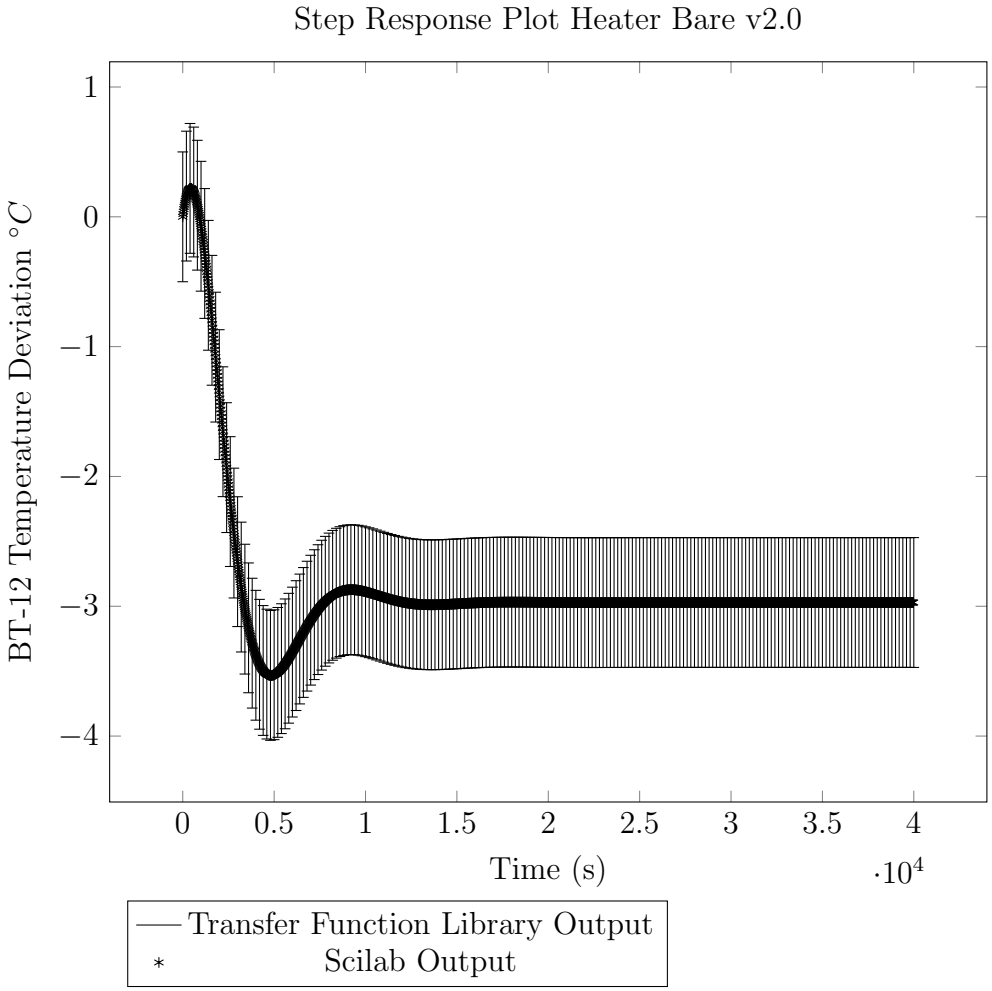


Figure 6.6: Transfer Function Rust Library Code to Code Verification using SciLab [Merzlikina and Prochina, 2020]

We can see from Figure 6.6 that the library written in Rust has almost the same output as the Scilab. There are also well within the $\pm 0.5K$ error bars. Thus, for the purposes of heater simulation, the simulation library has shown satisfactory results.

However, we also want to ensure that the second order underdamped response works well in general. For this, we consider the underdamped second order transfer function:

$$G(s) = \exp(-3s)\frac{2.5s^2 - 0.5s + 1}{3s^2 + 4s + 4}$$

The same transfer function was put in both Scilab and the Transfer Function Library and subject to the same transient of 9 units at $t = 0s$. The results are plotted in Figure 6.7:

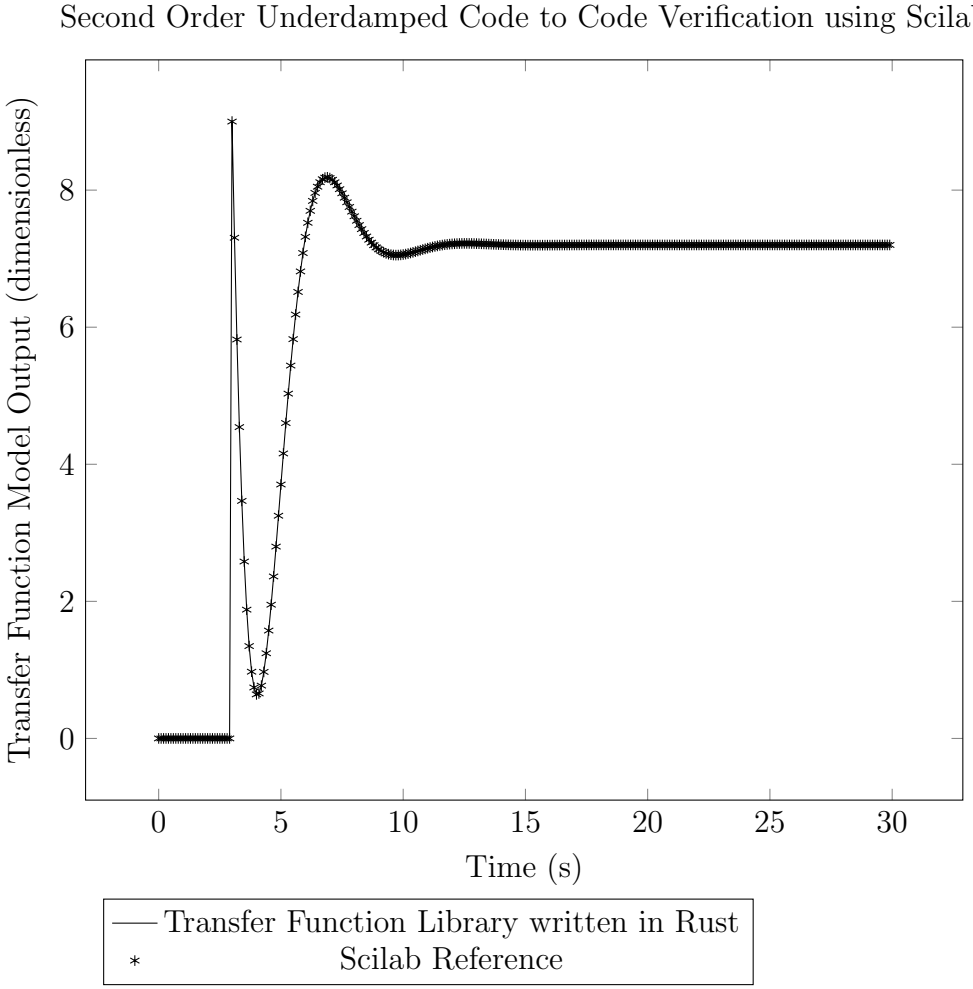


Figure 6.7: Step Input of 9 units (dimensionless) given at $t = 0s$ for Second Order Underdamped System

For Figure 6.7, I present the residual plots using the same crude truncation error estimation methods described for the FOPDT system in Figure 6.8:

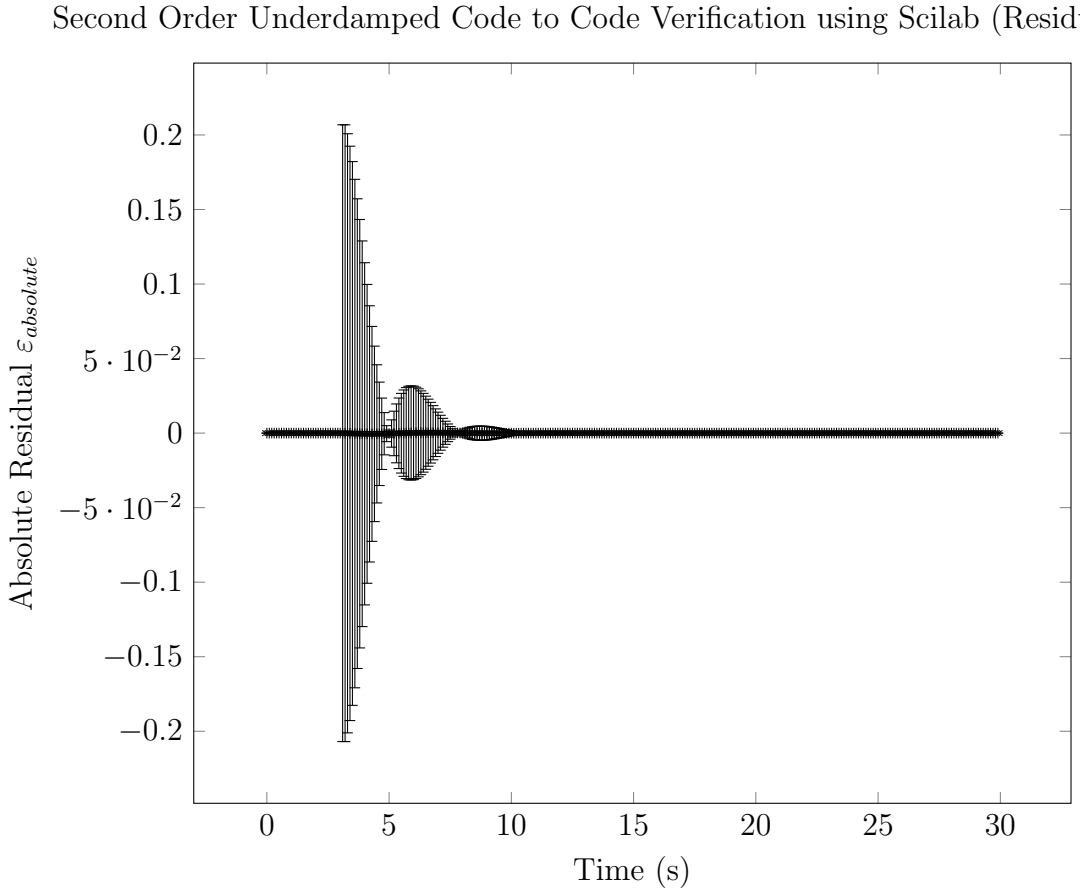


Figure 6.8: Residual Plot for Second Order Underdamped System with Step Input of 9 units (dimensionless) given at $t = 0s$

For most points within Figure 6.8, the points fit within truncation error. However, there are anomalous points where the residual exceeds the truncation error by as much as 1400 times. On closer examination, I found that the absolute values and relative values of these errors were small. The absolute residual had the magnitude of about 1.3×10^{-8}. When compared to the output of Scilab, these were about 2×10^{-9} as much as the output of Scilab. These are too small to even observe on the residual plot. We run into such issues because there are points of inflection in this transfer function simulation where $y''(t) \to 0$. Thus, the truncation error term also approaches zero at this point. Therefore, we observe such large values of residuals in comparison to the truncation error even though the truncation error is small. In my opinion, these errors are small enough to ignore for the purposes of this work, and for the purposes most engineering simulations.

Now, we want to deal with overdamped and critically damped systems.

We deal with the following critically damped transfer function:

$$G(s) = \frac{5s^2 - 2s + 1}{(s+2)(s+2)} = \frac{5s^2 - 2s + 1}{s^2 + 4s + 4}$$

And the following overdamped transfer function:

$$G(s) = \frac{5s^2 - 2s + 1}{(s+1)(s+2)} = \frac{5s^2 - 2s + 1}{s^2 + 3s + 2}$$

Results for the critically damped and overdamped transfer functions are presented in Figure 6.9:

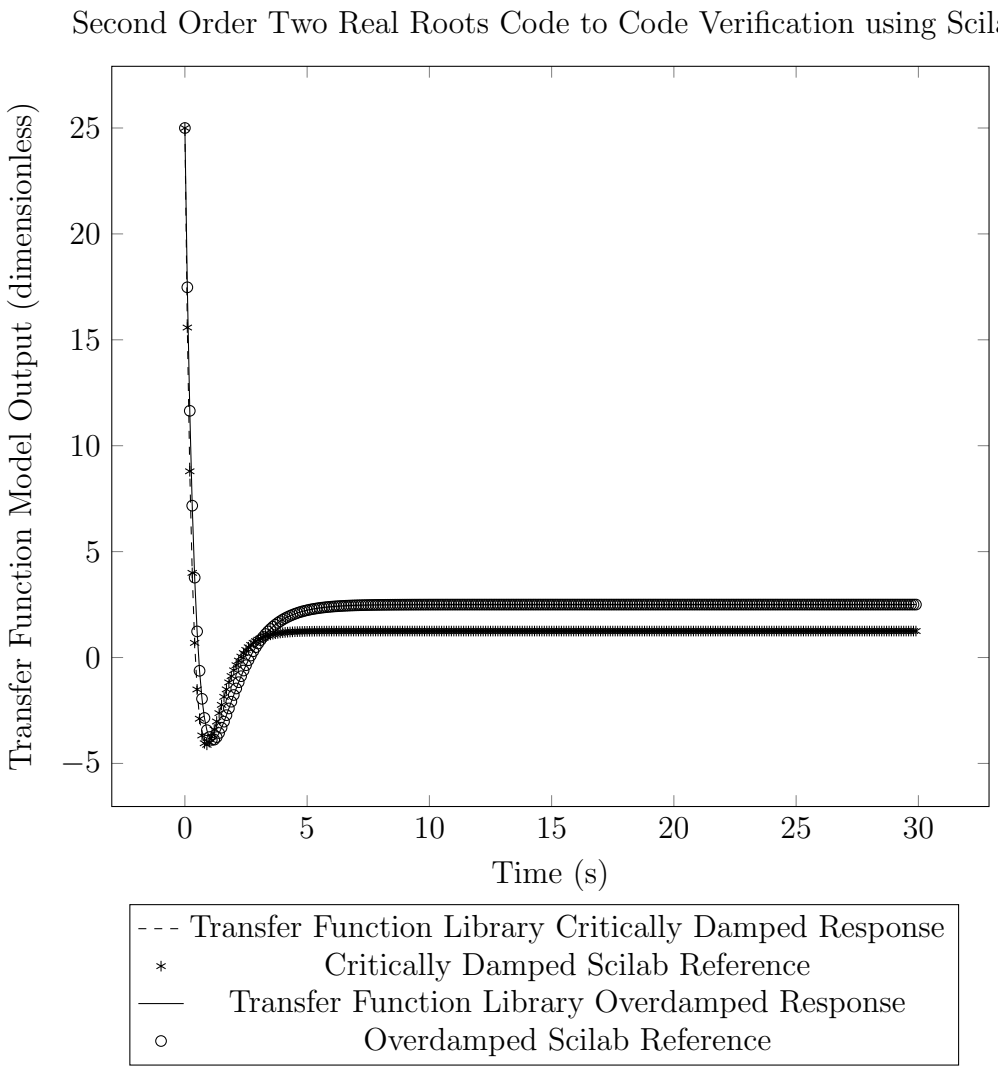


Figure 6.9: Step Input of 5 units (dimensionless) given at $t = 0s$ for Second Order Damped System with Two Real Roots

We can see from Figure 6.9 that the Transfer Function Library generally matches well with the Scilab reference for both the critically damped and overdamped transfer functions. In fact, the largest absolute error in the critically damped and the overdamped transfer function was less than 1% of the truncation error. In regions where the residual was larger than the truncation error, the residual was typically on the order of 10^{-8}. This is quite similar to the trends shown in Figure 6.8 where the residuals are too small to even be seen on the residual plot in comparison to a the typical truncation error scale. Now, overdamped and critically damped transfer functions are not used for this SNF controller, therefore I will not produce the residual plots for the sake of brevity. Suffice to say, however, that the transfer function library is able to produce stable second order behaviour similar enough to behaviour shown in Scilab. Therefore, the code to code verification effort has been successful.

Controller Simulator Construction

Besides transfer functions, it would also be important to have Proportional, Integral and Derivative (PID) controllers in the toolbox so that PID can be simulated if required. This was important for use in the second iteration of SNF controller. From experience gained from the first iteration, I found that the timescales for transients due to change in inlet coolant temperature were much longer than the timescales needed for the heater to respond. Therefore, I could construct a controller where the simulated neutronics transfer function determined the expected set point of the heater outlet temperature. Based on rudimentary feedback control, I could use a PI and PD controller to adjust the heater power such that its outlet met that set point. I used this approach in my second iteration of the SNF controller.

Proportional Controllers

Proportional controllers are trivial to design and code. They take the input in real-time and multiply by a controller gain K_c to give an output. One would only need to add extra programming if one thinks about dead-time. For the sake of development speed, I simply used the transfer function $G(s) = \frac{s+1}{s+1} = 1$ to represent proportional controllers. This is because prior code for first order transfer functions have already been developed. While this would have been a computationally heavier and somewhat overcomplicated representation of a proportional controller, using this implementation would not be an issue if computational speed is not a problem. Moreover, from a developer's perspective, the first order transfer function has already been tested and developed. Therefore, reusing the first order transfer function would be more expedient than writing a proportional controller with delay time function from scratch. Therefore, this was the approach I took. In this manner, I did not have to re-verify if the proportional controller matched the Scilab reference.

However, it would be good to see the behaviour of proportional controllers in comparison to Scilab when these are placed in a feedback loop as shown in Figure 6.10:

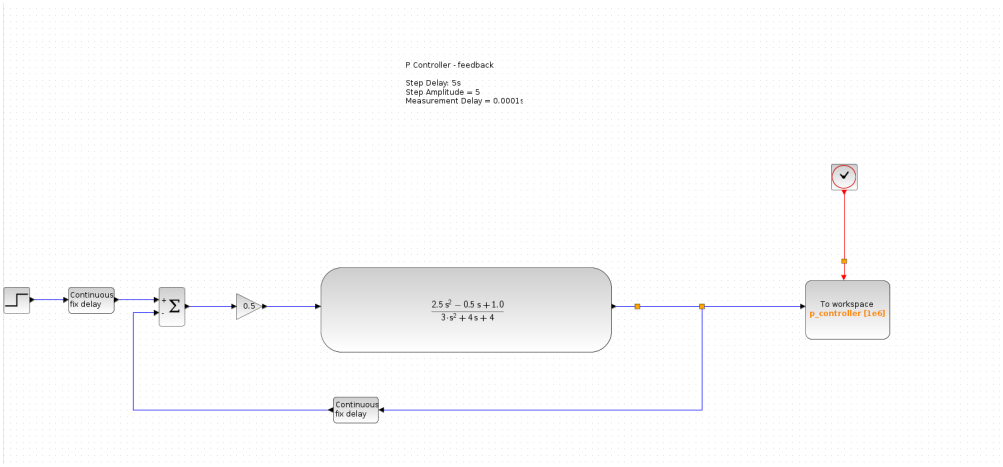


Figure 6.10: Proportional Controller Feedback Loop Block Diagram in Scilab Xcos

The "continuous-fix-delay" block in Figure 6.10 provided a small delay in time. The goal of this was to mitigate an "algebraic loop error". This is an error given by Scilab when attempting to solve feedback loops for all their variables simultaneously. Normally, at one time step, a sequence of events occurs. Firstly, the error signal is sent into the controller, the controller then sends a control signal into the transfer function to obtain an output at the first timestep. However, to calculate the error signal $y_{sp}(t) - y(t)$, a value for $y(t)$ must be supplied. Therefore, we need $y(t)$ in the error term to calculate $y(t)$. This is the essence of the "algebraic loop error". To solve this error, we can supply an initial value of $y(t)$ to calculate $y(t + \Delta t)$. To achieve this, I looked in online forums and found that I needed to introduce a small delay in the feedback loop [rupakrokade, 2018]. This is not unreasonable since in reality, the measuring device would take some finite amount of time to measure a signal. Therefore, this delay required for the solver could represent some sort of measurement delay. However, I set this measurement time to as small a value as possible so that I could emulate, as far as possible, an ideal measurement transfer function with no delay. This measurement delay was set at 0.1 ms to mimic perhaps a typical measurement delay time or latency by the controller. In reality, typical latency over local area networks (LAN) is on the order of 1 ms. The LAN setup was used by CIET to connect its sensors to ARCO via the PXI-e. Therefore, such a latency is not unrealistic. I then simulated the same idealised control loop in the transfer function library. However, no additional delay was required because I used the previous time step's value of $y(t)$ to calculate the error signal for the controller. This achieved a similar result as putting a delay in Scilab. Nevertheless, the time step became important in feedback loops despite my control blocks being analytically integrated because these signals were now discretised in time. Using controllers designed using Laplace Transforms in discrete time systems generally performs worse than using controllers designed using the z-transforms as the latter is designed with discrete time stepping in mind. However, for the purposes of this work, designing a controller from the ground up using a z-transform was not strictly required and the controller still performed well enough compared to Scilab.

I plotted the results of this system in Figure 6.11:

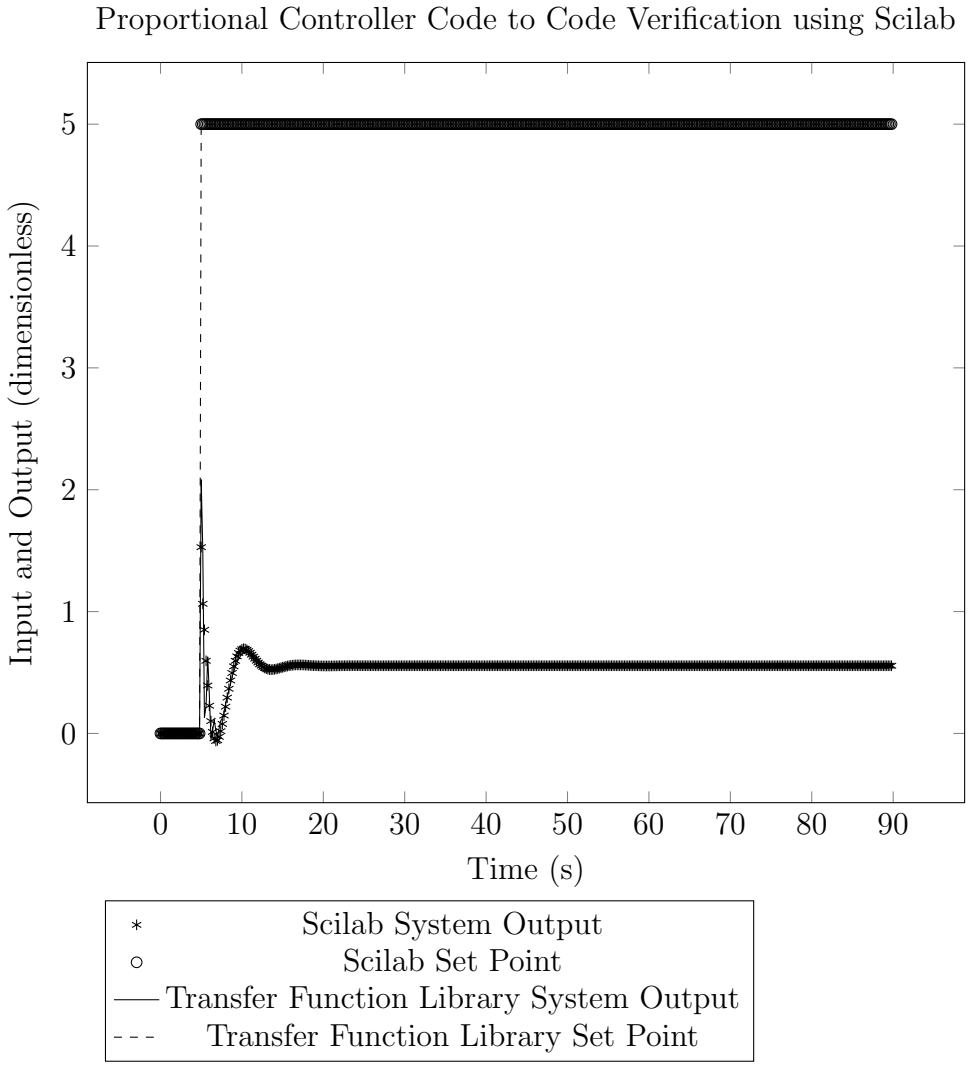


Figure 6.11: Proportional Controller Feedback Control, Scilab and Transfer Function Library Comparison

Generally speaking, the proportional controller in my transfer function library performs quite similarly to the controller block in Scilab even with the minor time delay. However, the transfer function library tends to produce a more jerky output as compared to Scilab. Now, Scilab simulates the controller block system using an automatically determined time step perhaps so that the feedback control system is simulated as close to a continuous time simulation as possible. However, for the transfer function library, I set the time step to $\Delta t = 0.2s$. Therefore, it may behave more like a discrete time feedback control system. This may be a significant contributing factor for the deviations as shown in Figure 6.12:

Proportional Feedback Controller Code to Code Verification using Scilab (Residual Plots)

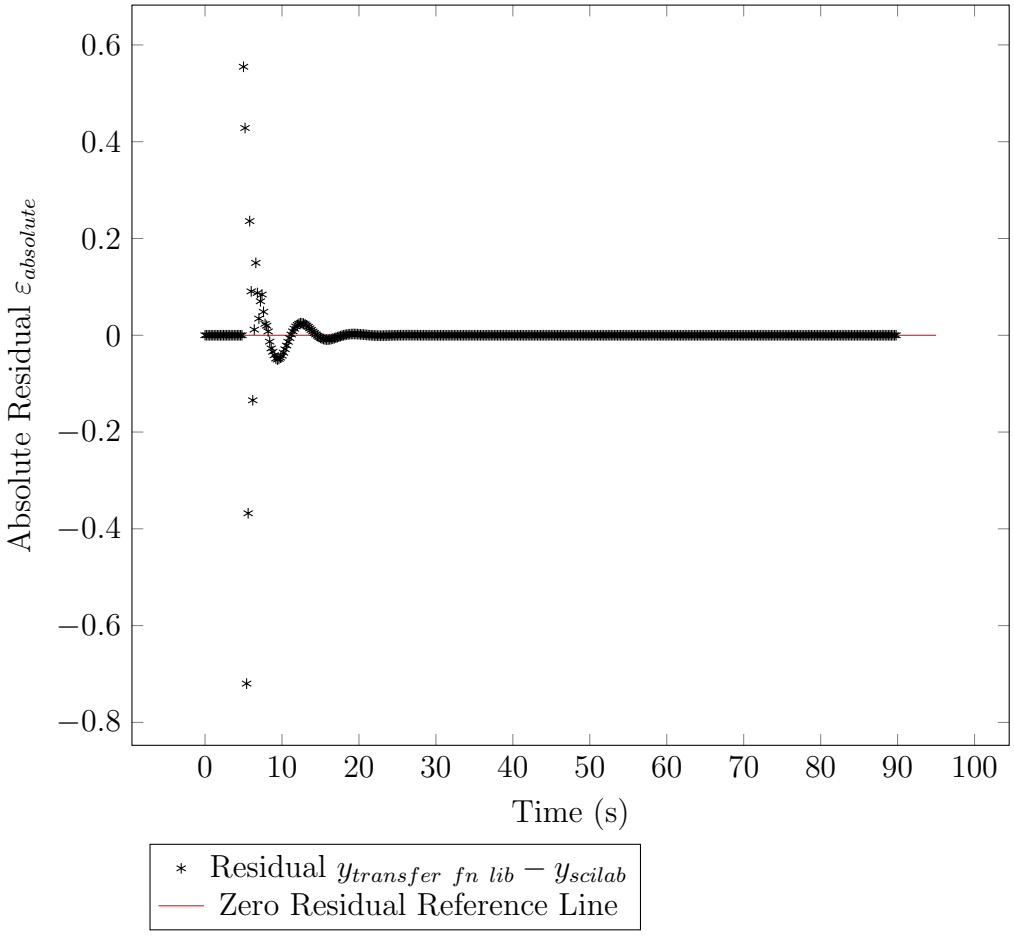


Figure 6.12: Residual Plot for Feedback Control Test with Proportional Controller

These residuals are not small in comparison to the step input and the final steady state output. This has important implications for the feedback control mechanism. For example, if at $t = 5s$, the set point is set to $y_{sp} = 5$, the error signal should have been 5 at $t = 5s$. The next time step to calculate would be for $t = 5.2s$. However, for analogue control, we should have a marked decrease in error between $t = 5s$ and $t = 5.2s$ because the system output would have increased. For digital control, the error would have been set at 5 for the duration of $\Delta t = 0.2s$. Therefore, the system response would have been greater as compared to an analog control scheme. Nevertheless, we see that the steady state outputs of the feedback loop in both Scilab and my transfer function library are virtually equal at 0.556 with a residual of 1.6×10^{-12}. If we were to use the package with real-world components, we may have issues as the additional deviation may damage components. However, for the purposes of creating a development sandbox for SNF controller development, this deviation is acceptable since the steady state output is similar.

Integral Controllers

The next item to design is an integral controller because I want to progress from a P feedback controller to a PI feedback controller. Integral controllers are rather challenging because they do not have modes that decay out. Therefore, we cannot use the same algorithms we used for the stable transfer functions to avoid memory leaks.

To start developing code for integral controllers, we start with the transfer function for integral controllers and obtain its response to a step input. The transfer function for an integral controller is:

$$G(s) = \frac{K_c}{\tau_I s}$$

Where τ_I is the integral time. The step response of an integral controller is simply the ramp function $\frac{1}{s^2}$ scaled by $\frac{K_c}{\tau_I}$. The ramp function is:

$$y(t) = t$$

Suppose $y(t)$ is zero initially, and there were two step inputs given by the user, one at t_1 and the other at t_2 with amplitudes a_1 and a_2.

After the first step input:

$$y(t) = a_1 \frac{K_c}{\tau_I} (t - t_1)$$

After the second input:

$$\begin{aligned} y(t) &= a_1 \frac{K_c}{\tau_I} (t - t_1) + a_2 \frac{K_c}{\tau_I} (t - t_2) \\ &= \left[a_1 \frac{K_c}{\tau_I} + a_2 \frac{K_c}{\tau_I} \right] t - \frac{K_c}{\tau_I} [a_1 t_1 + a_2 t_2] \end{aligned}$$

For a system with N inputs,

$$y(t) = t \sum_{i=1}^{N} \left[a_i \frac{K_c}{\tau_I} \right] - \sum_{i=1}^{N} \left[a_i t_i \frac{K_c}{\tau_I} \right]$$

The units here seem to check out as t and τ_I are of the same units, and $y(t)$ should be in units of $a_i K_c$. Hence, there is no need for vectors here. There should be an offset where:

$$\text{offset} = - \sum_{i=1}^{N} \left[a_i t_i \frac{K_c}{\tau_I} \right]$$

And a gradient:

$$\text{gradient} = \sum_{i=1}^{N} \left[a_i \frac{K_c}{\tau_I} \right]$$

Given this expression, one can simply program two mutable variables which are changed and every time an input is registered. The change to offset and gradient can be calculated and put into the object code representing the integral controller. To ascertain if the PI controller worked, I created a simple test case that I ran in both Scilab and the Transfer Function Library, and I compared the results. The block diagram of this test case is in Figure 6.13:

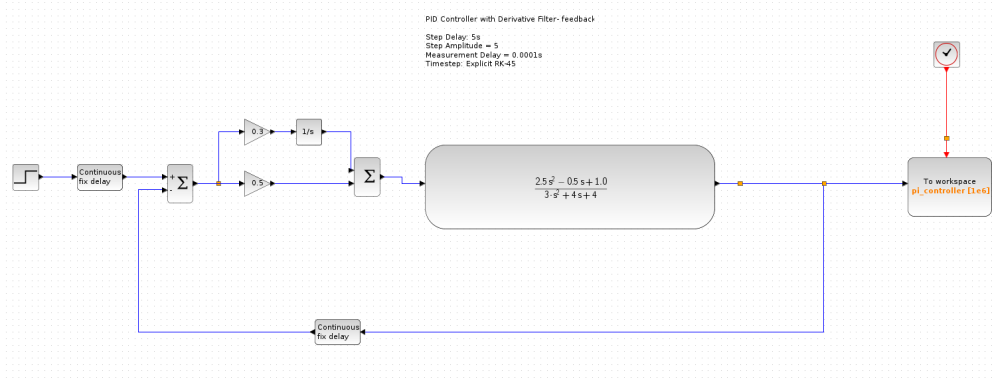


Figure 6.13: PI Feedback Controller Block Diagram

The results in Figure 6.14, show that the Transfer Function Library was able to cause the closed loop behaviour to behave in a generally similar way to its Scilab counterpart:

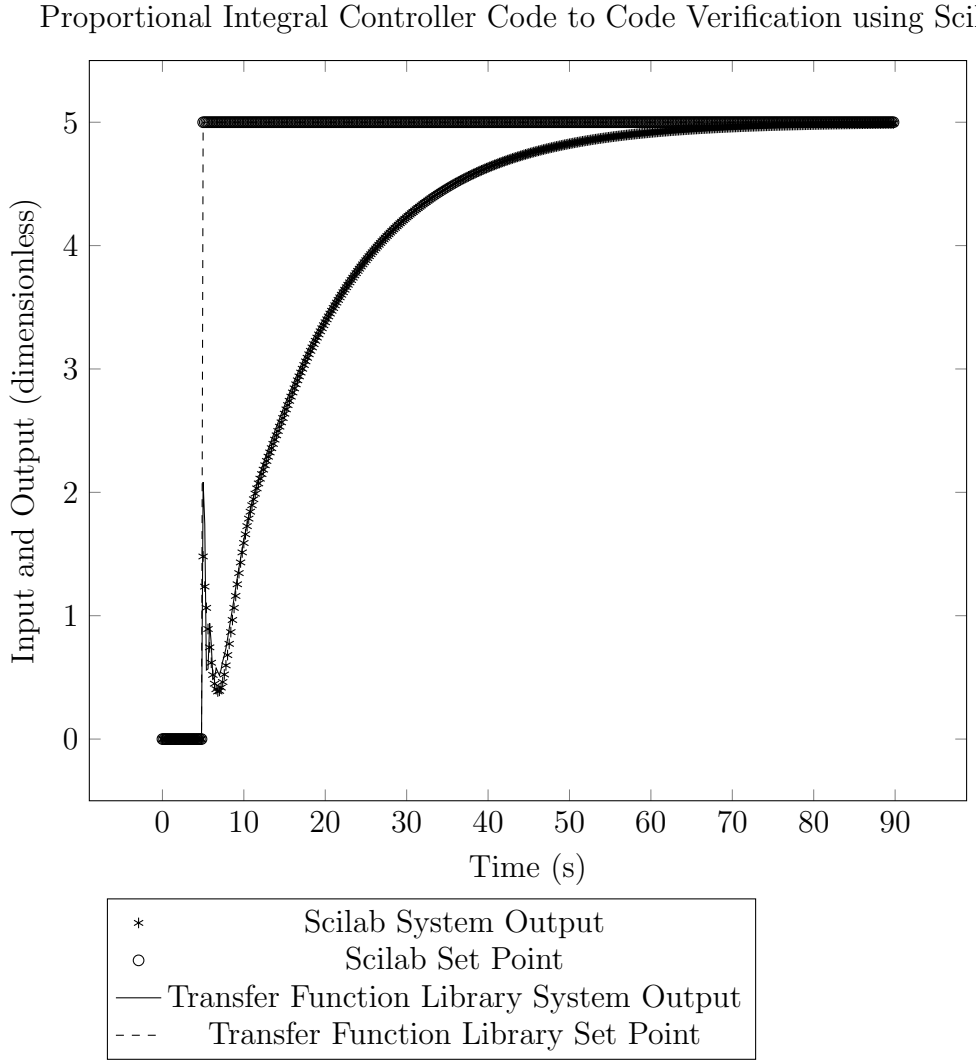


Figure 6.14: Proportional Integral Controller Feedback Control, Scilab and Transfer Function Library Comparison

While Figure 6.14 shows a generally close match in closed loop output between Scilab and the Transfer Function Library, , we do not get an exact match. This is shown in Figure 6.15:

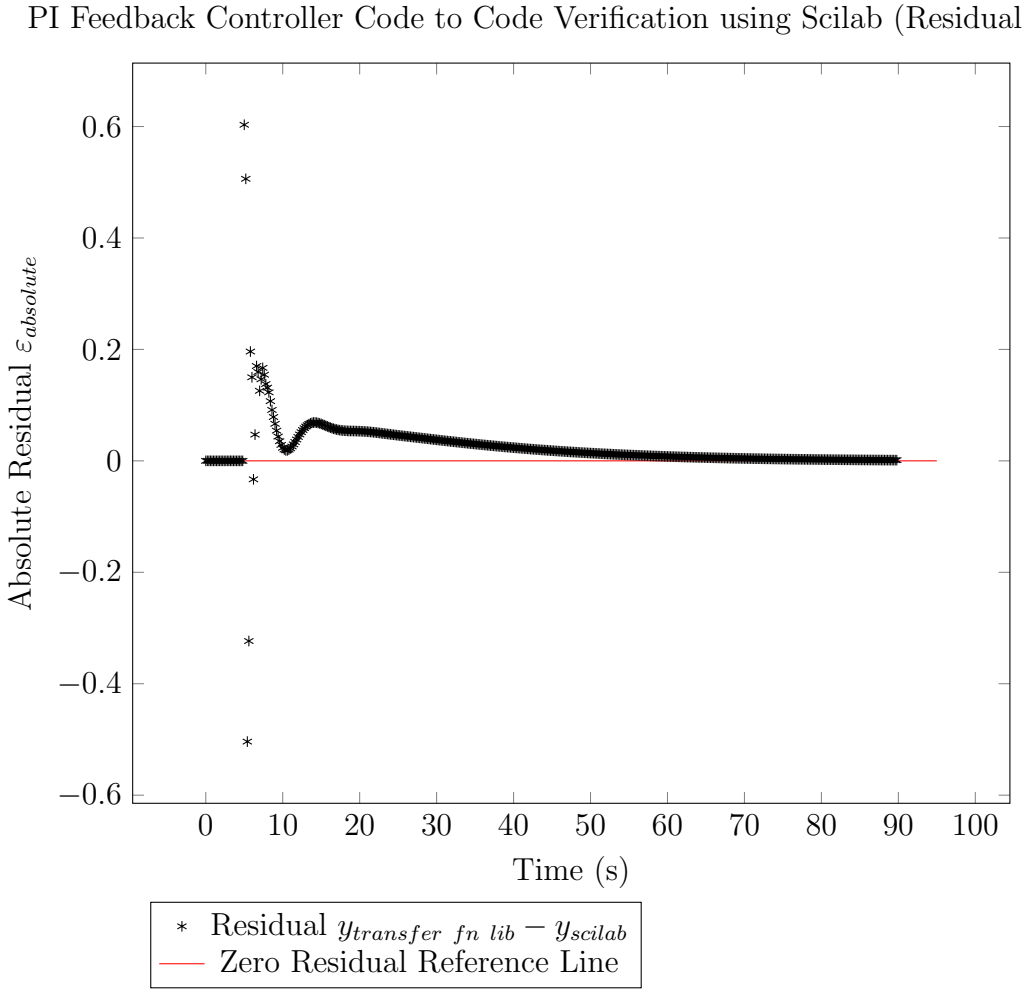


Figure 6.15: Residual Plot for Feedback Control Test with Proportional Integral (PI) Controller

Nevertheless, the PI controller does its job by bringing the system to the desired set point, and the residual tends to zero as $t \to \infty$. Again, these deviations are likely due to the time discretisation effects where $\Delta t = 0.2s$ as described earlier, and care should be taken if using such controllers for real process control. However, for the purposes of this work, the behaviour of the PI controller suffices. Therefore, I did not tweak the code further.

Derivative Controllers

To construct a full PID controller, I needed to design the derivative controller as well. Derivative controllers are slightly more challenging because its transfer function is s. This is not physically realisable and challenging to model. However, for the purposes of this work, we need to model its input to a step function as before.

Again, the transfer function of a derivative controller is s and the frequency domain representation of a step function is $\frac{1}{s}$ [Seborg et al., 2016]. The corresponding output in the frequency domain to a unit step input is 1. For this, the inverse Laplace Transform is [Seborg et al., 2016]:

$$\mathcal{L}^{-1}\{1\} = \delta(t)$$

Now, it would be rather challenging to simulate the delta function $\delta(t)$. There are two options though. First is to approximate it using two Heaviside functions. The first Heaviside function is step up and the other is a step down. The time between the two Heaviside functions is determined by the timestep. Consequently, since the area under the step is 1, the amplitude $a_{heaviside}$ is determined by:

$$1 = \Delta t * a_{heaviside} \tag{6.32}$$

The problem with this approach is that the time step Δt must be known beforehand. I wish not to make this assumption, and therefore I looked for another method. The second alternative is to use a filtered derivative controller, with the transfer function [Seborg et al., 2016]:

$$G(s) = K_c \frac{\tau_d s}{\alpha \tau_d s + 1}$$

Where τ_d is the derivative time, and α is known in literature as a derivative filter constant [A. Isaksson and Graebe, 2002]. This is commonly used It is typical for α to have values of 0.1 [Seborg et al., 2016]. However, it has been suggested in literature that α should be a fourth parameter in PID controller design besides K_c, τ_I and τ_d [A. Isaksson and Graebe, 2002]. Therefore α should be up to the user to decide. In terms of programming, it is essentially a first order system with one zero at $s = 0$. We can then reuse code from the first order transfer function for this.

Now, it is good to compare the differences in behaviour between filtered PID and non filtered PID controllers. Therefore a comparison was performed in Scilab to ascertain the differences in closed loop response between these two PID controllers. The Scilab block diagram for the unfiltered PID controller is presented in Figure 6.16:

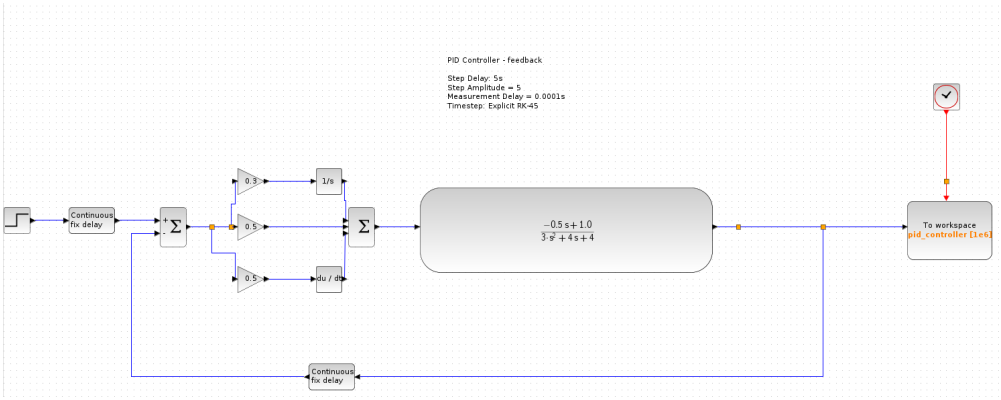


Figure 6.16: Scilab Block Diagram for Unfiltered PID Controller

In contrast, the filtered PID controller setup is presented in Figure 6.17:

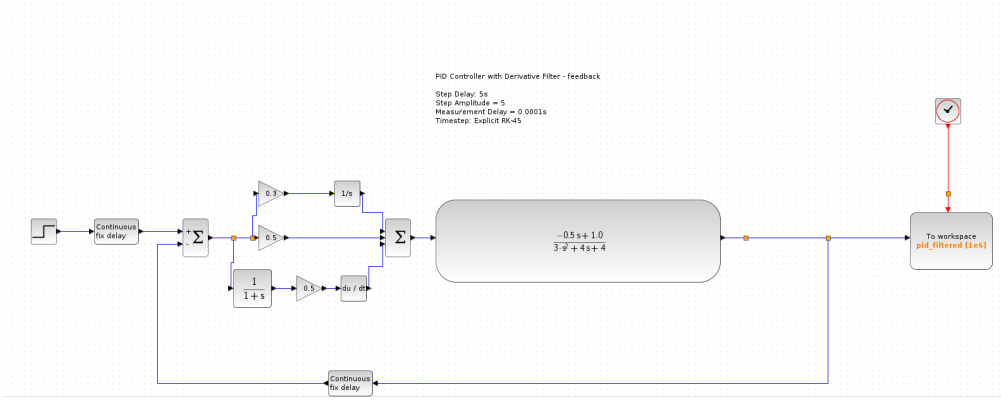


Figure 6.17: Scilab Block Diagram for Filtered PID Controller

In Figure 6.17, essentially a low pass filter with transfer function $\frac{1}{s+1}$ was added to the derivative controller. When both control loops were subject to a set point change of 5 units at $t = 5s$, the responses were recorded and presented in Figure 6.18:

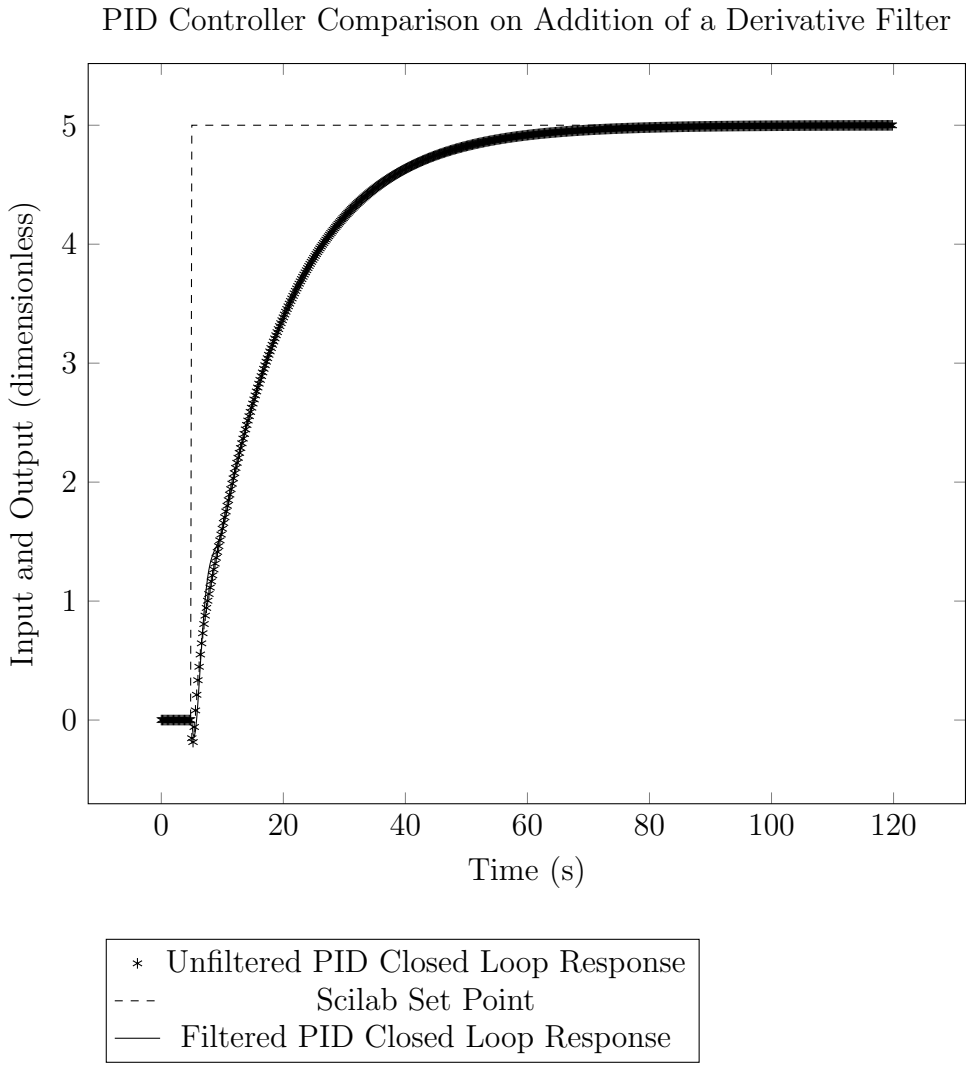


Figure 6.18: Proportional Integral Derivative (PID) Controller Feedback Control, Comparison between filtered and unfiltered Variants

Based on the general trend of Figure 6.18, it made little difference whether the filter was added to the derivative controller. Since the behaviour is more or less similar, I refrained from implementing a pure derivative controller within my program.

Now that we have discussed derivative filters, we shall move on to the verification case study. The closed loop system in the Scilab block is shown in Figure 6.19:

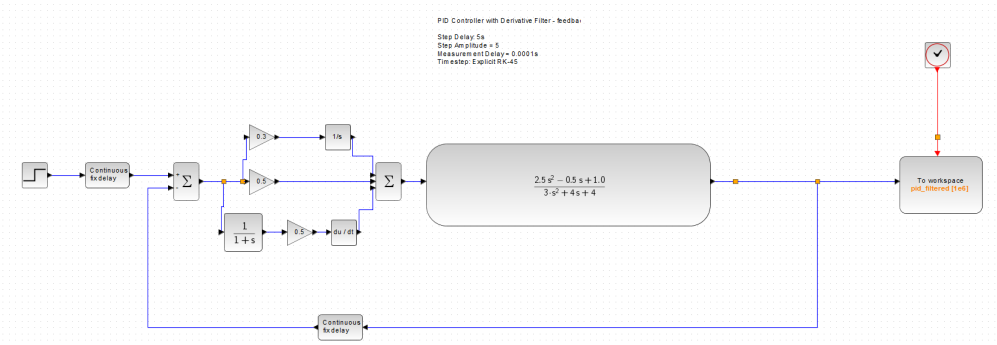


Figure 6.19: Scilab Block Diagram for Filtered PID Code Verification Test Case

This closed loop feedback control system with a filtered PID controller again simulated in Scilab and the transfer function library written in Rust. The results are presented in Figure 6.20:

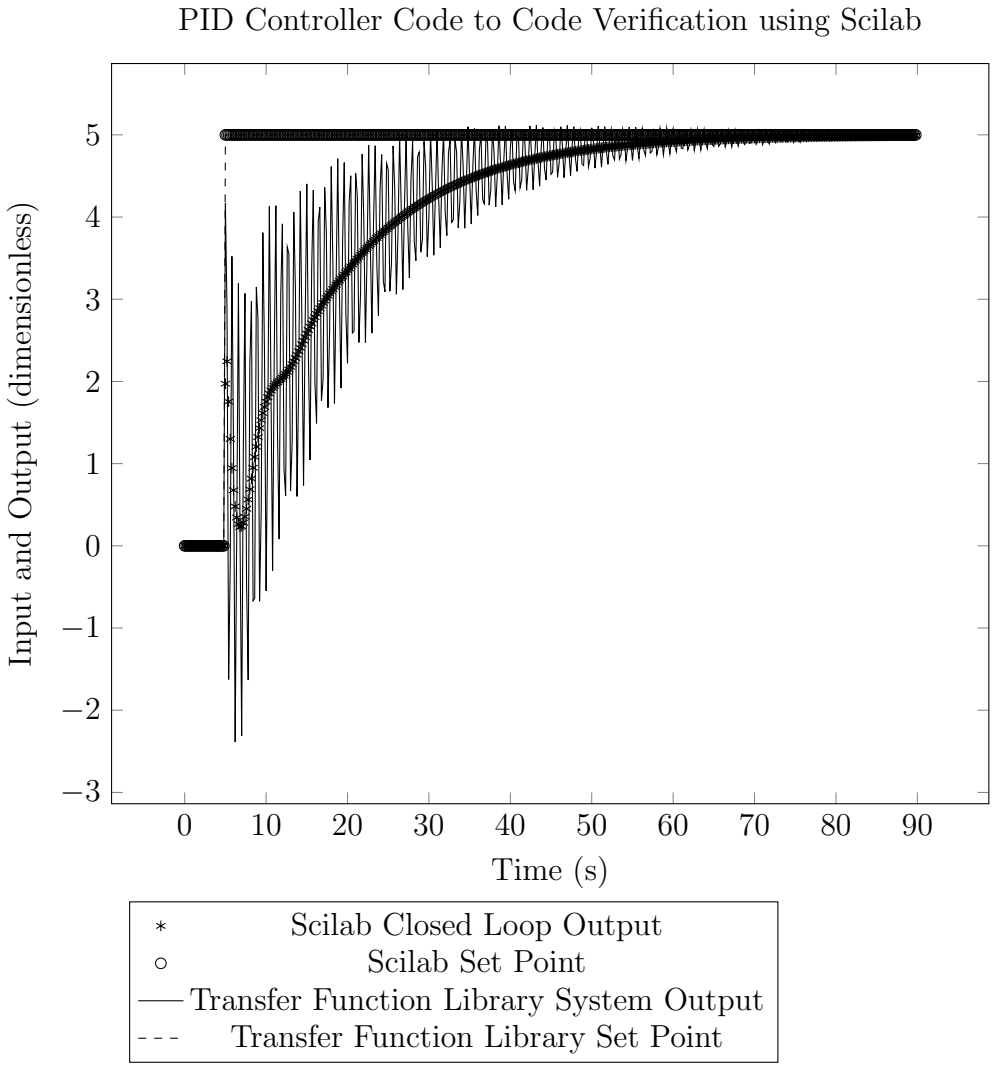


Figure 6.20: Proportional Integral Derivative (PID) Controller Feedback Control, Scilab and Transfer Function Library Comparison with $\Delta t = 0.2s$

From Figure 6.20, we can see that there is generally poor agreement between the feedback controller using the transfer function library and that of Scilab. One possible reason is due to me using a time step of $\Delta t = 0.2s$ in the transfer function library simulation. In contrast, the Scilab simulation, once again, used an automatically determined time step. To verify if this time discretisation was indeed the issue, I repeated the simulation using a $\Delta t = 0.02$. However, I plotted data I sampled every 0.2 seconds. This is shown in Figure 6.21:

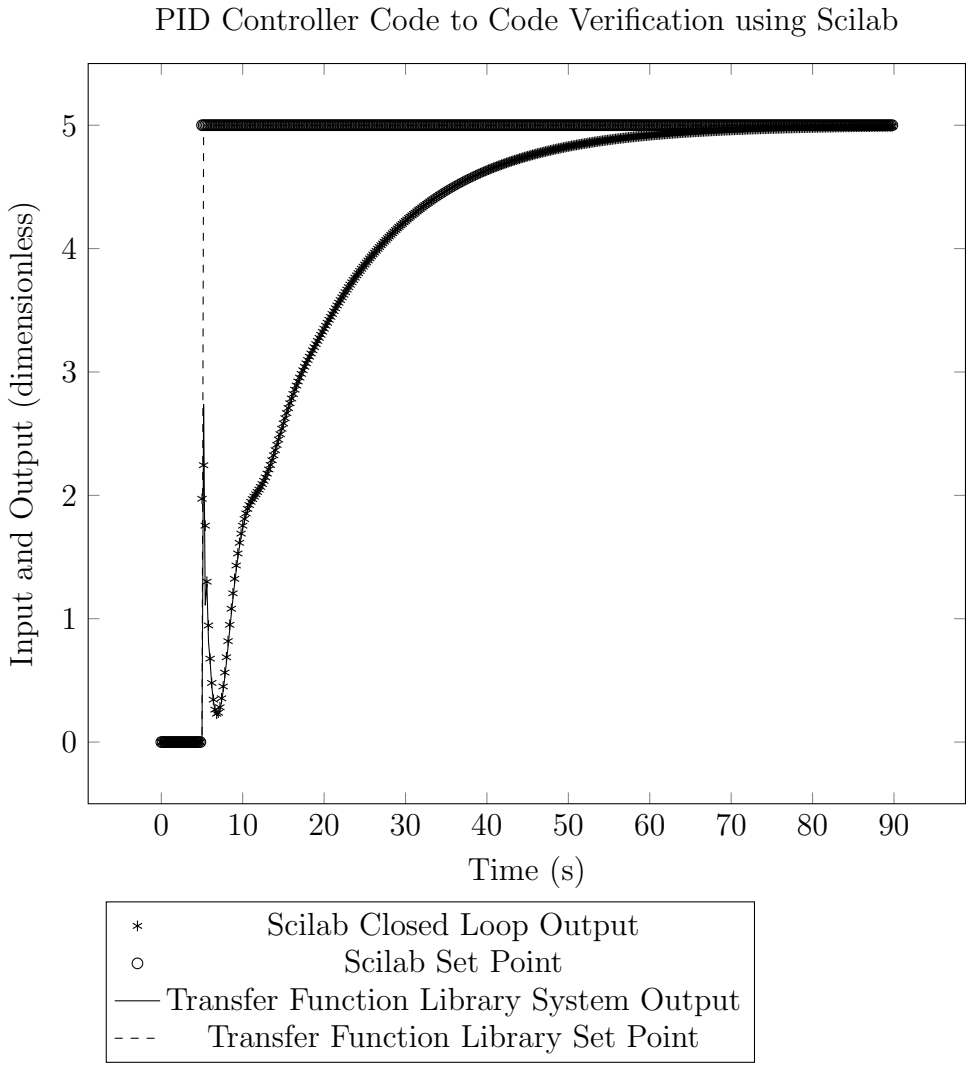


Figure 6.21: Proportional Integral Derivative (PID) Controller Feedback Control, Scilab and Transfer Function Library Comparison with $\Delta t = 0.02s$

In Figure 6.21, the agreement is generally much better than in Figure 6.20. The residuals are shown in Figure 6.22:

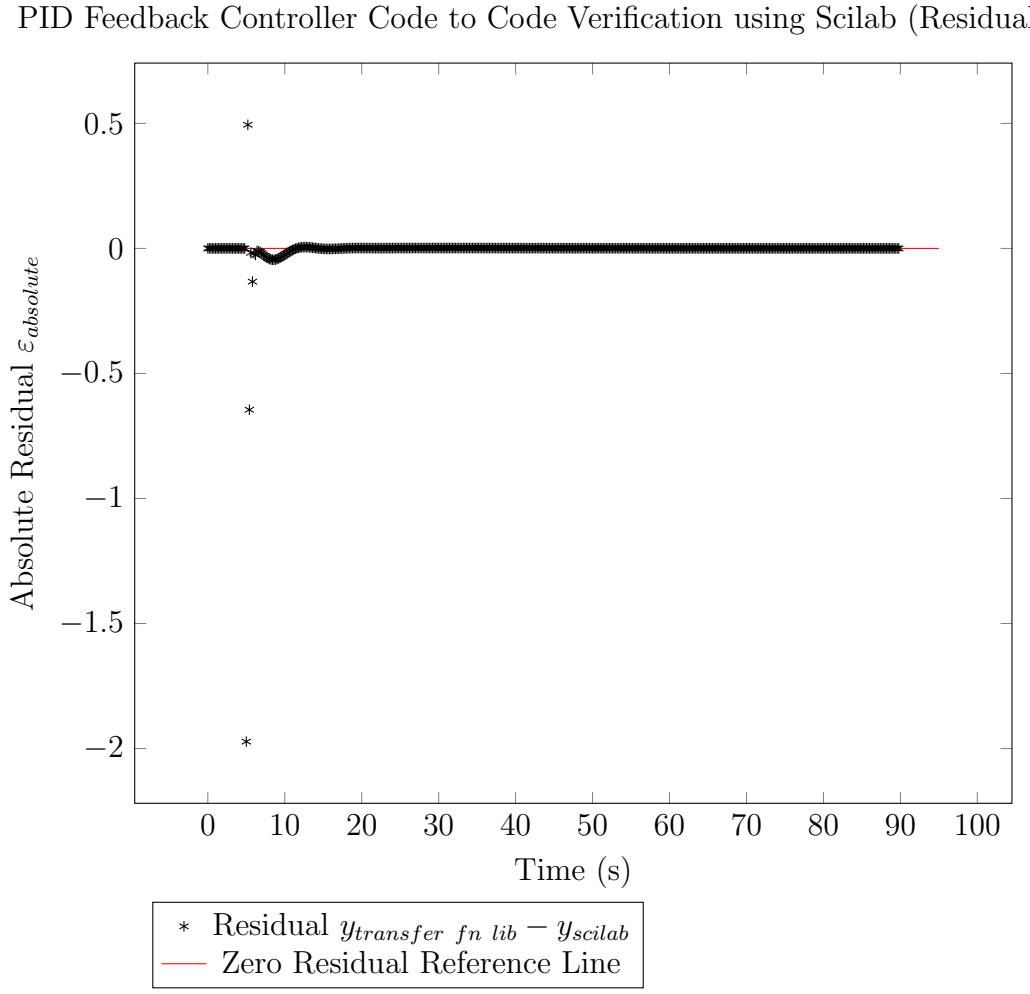


Figure 6.22: Residual Plot for Feedback Control Test with Proportional Integral Derivative (PID) Controller

Figure 6.22 shows that if timestep was not an issue, then the closed loop behaviour of the PID controller simulated in the transfer function library generally matches the data of the Scilab data except for a few data points. Therefore, for most of the simulation, the closed loop behaviour of the PID controller in the transfer function library matches that of Scilab. The large residuals seen in Figure 6.22 can be attributed to a some reasons. The most important of which was that the simulation in the transfer function library was started one timestep (0.02 seconds) later than that of the Scilab simulation. Hence, the simulation step input in y_{sp} started at 5.02 seconds for the transfer function library and 5.00 seconds for the Scilab dataset due to the way I programmed the simulation loop, and not the way I programmed the transfer function library. This small delay caused a significant residual from $t = 5s$ to $t = 6s$ in the simulation. However, it did not significantly impact the overall closed loop behaviour for the rest of the simulation despite this small deviation. Given that

this is the case, I decided not to spend further time in repeating the experiment. Figure 6.21 and Figure 6.22 show that the PID behaviour has been sufficiently replicated in the transfer function library written in Rust.

However, the user should be cautious when using the analogue PID controllers in discrete time systems. If the timestep is too large, we will then observe the oscillatory behaviour in Figure 6.20. In real-time systems such as the digital control system in ARCO-CIET, it may not be possible to obtain data at such small timesteps. Therefore, using filtered PID controllers in this manner, we ought to expect oscillations to occur. These oscillations that come about due to sudden changes in set point can be mitigated by placing the derivative controller portion in the feedback portion of the loop [Seborg et al., 2016], thus eliminating the proportional and derivative kick due to sudden changes in y_{sp} [Seborg et al., 2016]. This issue aside, I considered the transfer function library in this form sufficiently capable for the purposes of this thesis. Therefore I released it under the Rust crates directory as "chem-eng-real-time-process-control-simulator" version 0.0.4. This is a nod to when I first learnt process control as a chemical engineer in my undergraduate work, and also because the majority of references I used for this library were chemical engineering textbooks [Seborg et al., 2016] and Perry's Chemical Engineer's Handbook [Perry and Green, 2015].

Graphical User Interface (GUI) and Postprocessing

Now, for the Type I Digital Twin of CIET, it is quite necessary to have a graphical user interface (GUI) so that the operator can change the power levels of CIET as if he or she was controlling CIET via ARCO-CIET. Therefore, I needed to construct a GUI as a frontend for the Type I Digital Twin preferably in Rust. It may not be as aesthetically well developed or professional looking as Labview, but it got the job done. This GUI was based on the eframe GUI framework and the closely related egui crate [Ernerfeldt, 2023a], both of which were written in Rust. Both are also published under Free and Open Source (FOSS) Licenses such as Apache 2.0 and the MIT license. Therefore, this makes the code highly accessible to all users. I also used an eframe template made by Ernerfedlt [Ernerfeldt, 2023b] as the baseline with which to start building the tabs for the GUI. Additionally, I used Litvin's "rs-value-plotter" [Litvin, 2023] as a template with which to build graphs which could respond in real-time to user input. Figure 6.23 shows one iteration of this GUI used for testing and development.

Figure 6.23: GUI Demo using the egui Framework

This GUI made it possible for me to interact with the Digital Twin Server in real-time using the OPC-UA Client written in Rust. Additionally, it enabled me to integrate my "chem-eng-real-time-process-control-simulator" crate with the client seamlessly as it was also written in Rust. This made testing and development of the Digital Twin simpler because only one programming language (Rust) was used as opposed to a mixture of Rust and Python, or Rust, Python and Javascript. This simplicity greatly sped up the testing and development process of the SNF controller.

For postprocessing, however, I did not set up an interactive GUI so that the user clicks a button to start recording data. Instead, the OPC-UA client has a thread that runs so that comma separated values (csv) files are produced. For this, I used the "csv" crate written in Rust [Gallant, 2023].

SNF Controller Development Process

Now that we have constructed our transfer function simulator in the form of the "chem-eng-real-time-process-control-simulator" crate written in Rust, we can now outline the SNF controller development process within the context of the Type I Digital Twin for CIET's Heater. This is to demonstrate how effective the Type I Digital Twin is for speeding up development of SNF controllers. Since we have previously mentioned how the speed increase was to be quantified, we now focus on the details of how SNF Controllers can be developed for the purposes of this work.

Simulated Neutronics and Thermal Inertia Controllers

Now, there are several approaches for SNF Controller Designs we can consider for this work. As mentioned in the literature review, the Desire Loop SNF Controller utilised PRKE based transfer functions for [Kok and Van der Hagen, 1999]. These would measure the actual void fraction using a sensor within the DESIRE IET loop. Then the void feedback signal would

be passed through a digital controller which would then control the DESIRE loop power supplies [Kok and Van der Hagen, 1999]. The key assumption for SNF facilities such as the DESIRE loop is that there is thermal hydraulics similitude between the actual reactor and the scaled SNF facility [Kok and Van der Hagen, 1999]. In this manner, we only need to add some transfer functions in order for the void-reactivity feedback to be accurately simulated [Kok and Van der Hagen, 1999].

Unfortunately, for CIET, it was scaled originally to natural circulation flow for decay heat removal within the Pebble Bed Advanced High Temperature Reactor (PB-AHTR) rather than the Mark I PB-FHR Design [De Wet and Per F Peterson, 2020]. However, it was then adapted for use, with some scaling distortions, for the Mark I PB-FHR designs as it still matched the thermal hydraulics behaviour of the Mark I relatively closely [I. M. B. Johnson, 2022; De Wet and Per F Peterson, 2020; Nicolas Zweibaum, 2015]. The alternative would be to construct another IET scaled to a more recent reactor design. However, reactor designs change and evolve. Therefore, the use case of CIET may evolve over time as well. In this case, we are simulating a forced flow unprotected transient for an arbitrary FHR rather than a natural circulation flow for decay decay heat removal for the PB-AHTR or Mark I. We could, in theory, construct another IET for this purpose. However, constructing an IET in and of itself is not a small feat for an academic research laboratory. The more convenient option is to make do with the experimental facilities available. Therefore, for this dissertation especially, similitude does not exist for thermal hydraulics since we base our reactor feedback on CIET. Therefore, the SNF controller would not only have to simulate the neutronics feedback via various means, but also the thermal inertia of reactor of interest. This adds an extra layer of challenge especially if the disparity in thermal hydraulics is large.

One way this may be possible is that of a "black-box" model approach. In this regard, we do not worry about the specifics of the thermal hydraulics within the reactor. Instead, we only concern ourselves to design a controller so that the scaled inlet temperature of the arbitrary FHR matches that of the inlet temperature of the CIET Heater, and the scaled outlet temperature of the arbitrary FHR matches that of the outlet temperature of CIET's Heater. If the arbitrary FHR has high thermal inertia, then its outlet temperature would change extremely slowly. If CIET's Heater has a low enough thermal inertia, one could design a controller such that CIET's Heater produces the desired outlet temperature of the arbitrary FHR. If the arbitrary FHR is meant to have a slow increase in outlet temperature, the controller can send a slowly increasing power signal to CIET's Heater to match this behaviour after scaling. The same can be said for when an outlet temperature decrease is expected. One could program the controller to slowly reduce the power of CIET's Heater such that the decrease in the outlet temperature of CIET's Heater matches that of the arbitrary FHR. In this case, the heater power is not scaled to the reactor power, and therefore its value does not hold as much physical meaning as before. Therefore, it is not as important to monitor except that it operates within safety limits and that the inlet temperature to outlet temperature transfer function remains the same as the scaled arbitrary FHR. However, there is one problem with this approach: for some transients, the outlet temperature may be lower than the inlet temperature. To illustrate this, I once again present the Step Response test

of the arbitrary FHR to a 100K increase in inlet temperature in Figure 6.24:

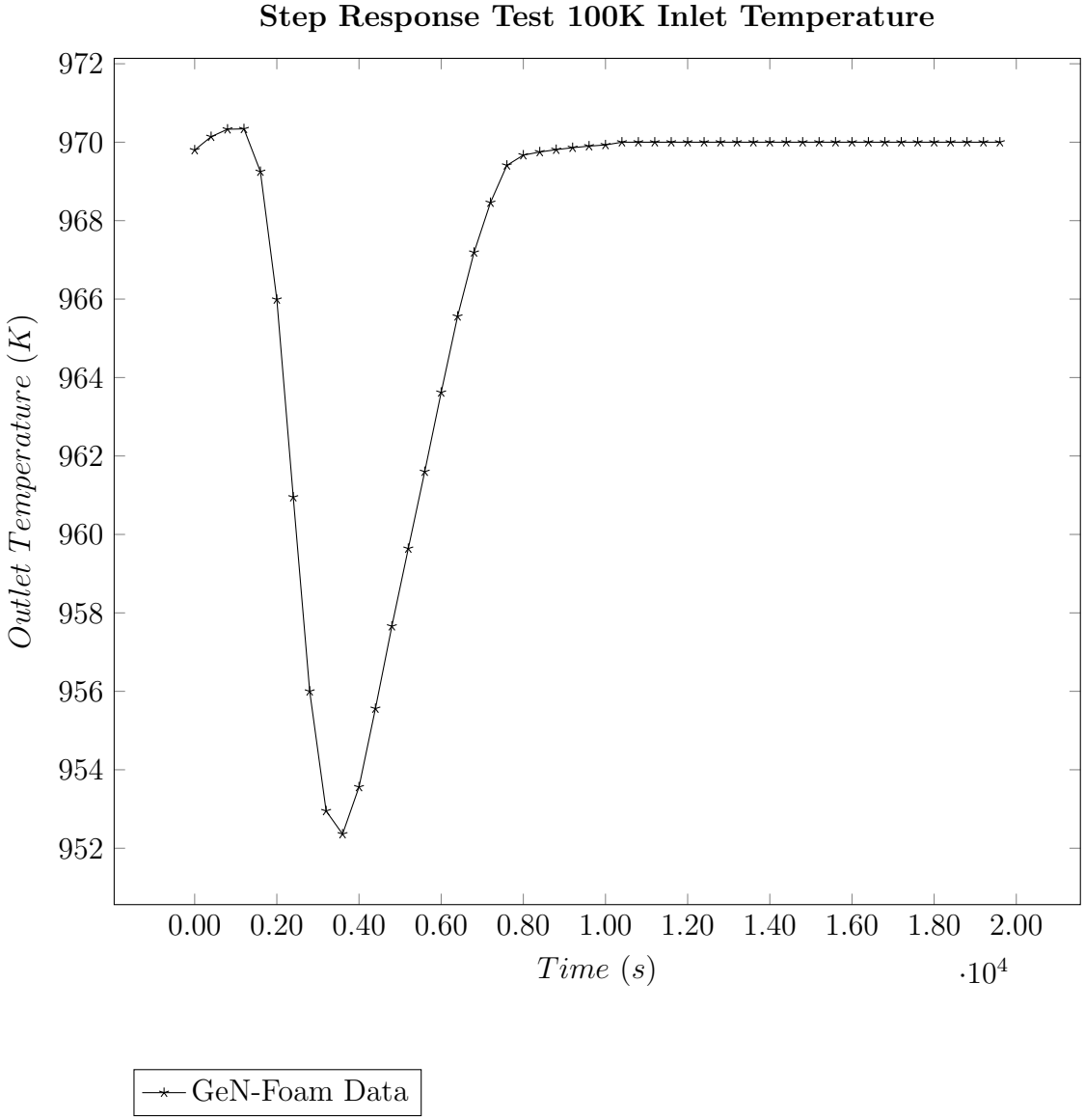


Figure 6.24: Step Input of 100K applied to Inlet Temperature for GeN-Foam compared to Derived Transfer Function

In Figure 6.24, the arbitrary reactor was brought to a steady state where its inlet temperature was 873 K and its outlet temperature was around 970 K. The transient of a 100K step increase in inlet temperature was done at $t = 0$ seconds so that the inlet temperature was 973 K throughout. This effectively shut down the arbitrary FHR. Therefore, for a few hours, the outlet temperature of the arbitrary FHR was lower than its inlet temperature likely due to negligible heat generation and cooler fluid still remaining within the arbitrary FHR.

Therefore, the fluid residence time and thermal inertia contributed to the outlet coolant temperature being significantly cooler than the inlet coolant temperature from $t = 2000\ s$ to $t = 6000\ s$. For a SNF Controller with thermal inertia simulation capabilities, this means that the outlet temperature must be colder than the inlet temperature for at least one hour. In this regard, parasitic heat losses for the heater are quite necessary in order to reproduce this effect. Therefore, the heated section must also be able to cool the liquid in addition to heating the liquid up. In CIET's present configuration, this is not possible even with parasitic heat present when the heater was turned off. Hence, only a subset of transients can be simulated with CIET's Heater in its present state. Therefore, if SNF facilities were designed to simulate thermal inertia in addition to neutronics feedback, then we should design a heater and cooler hybrid to account for situations such as those in Figure 6.24. Designing a new facility for the purposes of SNF and simulated thermal inertia is out of scope of this work, but its design process would likely benefit from using Type I Digital Twins as well.

Design Options for SNF Controllers in this Dissertation

For now, CIET's Heater, or its digital twin, is able to simulate a subset of transients which require SNF capability given the right SNF controller. In this dissertation, I chose to use the Type I Digital Twin to test analogue feedforward controllers since this was the most conceptually straightforward. I also tested feedback PID controllers since the time scale for SNF due to increased inlet coolant temperatures was much longer than the timescale for the heater thermal inertia and residence time. These two design options did not fully reproduce the desired SNF Controller behaviour. However, designing a working SNF Controller is outside the scope of this dissertation. The main point is to show how the Type I Digital Twin of the Heater sped up the design iteration process. For this purpose, two design iterations were sufficient. Hence, I did not perform further design iterations for the SNF Controller. This is left for future work.

6.3 Results

After performing the two design iterations for SNF Controller using the Type I Digital Twin of CIET's heater, I was able to demonstrate two working iterations of the SNF Controller. These did not reproduce the SNF behaviour completely, but the PID controller was able to follow the reference SNF behaviour quite closely for most of the transient where there were no sudden changes in heater inlet temperature (BT-11). These two design iterations took about 21 OPC-UA Client restarts to develop in total over two or three working days. This equated to about 24 man hours of work. Assuming one OPC-UA client restart equates to one successfully run CIET iteration experiment that took about three working days within one week to complete, we would require about 21 hours per person per successful experiment. This assumes that one working day consisted of seven hours of time spent in CIET and spent debugging CIET. Since CIET required two people to operate, we can assume that about 42

man hours of work was required per experiment. If this iterative process was done over 21 experiments, then we would require about 882 man hours of work. Based on this estimate, designing the SNF Controller using the Type I Digital Twin of CIET's Heater was roughly 21 times more efficient in terms of working hours. Also, roughly a total of 861 man hours were saved when performing the first two design iterations on the Type I Digital Twin. Of course, designing a SNF Controller would require many more iterations than these two design iterations, but this would only further prove that the Type I Digital Twin is an essential part of the design process if one wanted to be efficient with time.

Of course, to write code for this Type I Digital Twin from the ground up, it took roughly two years of trial and error. Nevertheless, I conjecture that the time spent in these two years would not only help to save my own time should I want to continue writing SNF controllers, but would also help others if they want to do something similar. Thus, this was time well spent. Additionally, several lessons were learnt during the first two design iterations. These lessons, along with the design iteration results are presented in the following subsections.

SNF Development Initial Results

Let us now discuss some of the initial results for how the SNF controller performed and some of the important learning points. Using the aforementioned methodology, I started developing my SNF controller using the Type I Digital Twin Rust Server and the Rust OPC-UA client. I took approximately 13 iterations to get my initial results as I had to debug various issues and configure the workings of the transfer function library and csv exports correctly. This was meant to mimic the development process I would have taken if I used ARCO-CIET to program my SNF controller.

Unsurprisingly, my first results for the SNF Controller were not particularly successful. When I decreased the inlet temperature of the heater (BT-11), the outlet temperature (BT-12) ought to have increased after sometime due to higher reactor power output. Nevertheless, erroneous behaviour and various bugs caused the SNF controller to switch off the heater after some time despite the fact that the heater was meant to increase power output. This behaviour is shown in Figure 6.25:

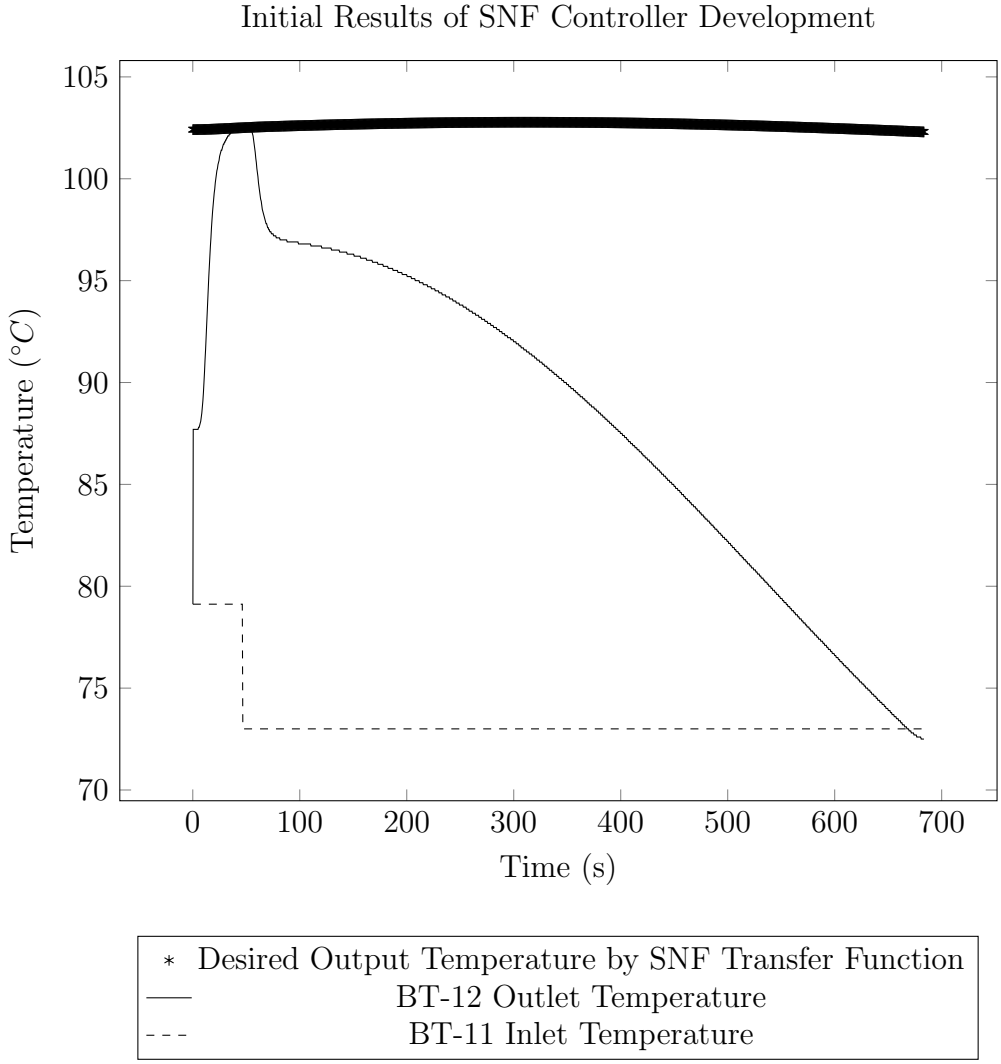


Figure 6.25: Initial Postprocessing Results after 13 Iterations

In Figure 6.25, the heater outlet temperature was initially set to around 80 $°C$. However, it took about 50 seconds for the heater to reach a steady state coolant outlet temperature of 102.24$°C$. At roughly 100 seconds, I reduced the inlet temperature of the coolant. This should have gotten the outlet coolant temperature to increase due to increased reactor power output. However, the SNF controller was not able to compensate for the decrease in inlet temperature. This is because the SNF Controller transfer function did not account for the decrease in inlet coolant temperature. This was an oversight on my part, but it showed the need for iterative design processes. Additionally, after the initial response, the SNF controller essentially lowered the heater power until it was 0 kW. This was puzzling, but at least the basic data logging functions and other basic functions were working correctly. I did not manage to debug the root cause of this issue until the second design iteration was

complete.

These first 13 iterations took approximately one day. If this iteration process was done in ARCO-CIET, the approximate time, assuming one could find the right schedules and people for the experiment, would have been roughly 13 weeks or almost one semester. While the SNF controller was not successful in that it did not produce the correct SNF behaviour, it showed that the timescales for the SNF transients due to changes in inlet coolant temperature were much longer than the timescales for transient conjugate heat transfer (CHT) within the heater. This is because the thermal inertia of the heater is very low in comparison to the thermal inertia of the fluid, at least compared to typical ratios found within FHRs and the arbitrary FHR. Given that this is the case, I opted to use a simpler PID controller in order to produce the desired SNF behaviour rather than debug the root cause of this error.

The key learning point for this design iteration was that I needed to consider a state space representation for the SNF Controller transfer function or state space model. This would have allowed for a control scheme with multiple inputs, and allowed me one more transfer function to try and reject the decrease in outlet temperature brought about by decrease in inlet temperature.

PID Type SNF Controller

Given this state of affairs, I decided to try the PID type SNF controller. In this setup, the SNF transfer function would determine the set point for the heater outlet temperature BT-12. Given such a set point, the controller would control the heater power so that the error between the Digital Twin's BT-12 temperature and the set point was minimised. In this regard, the SNF behaviour would have been adequately replicated because the timescales were so long. To attain the results for the second design using PID, approximately eight attempts were needed. This took around one to two working days because of the several hour long times for the transients.

When I iteratively designed the PID controller, I found severe limitations when using PID controllers acting on the error. As expected, the PID controller produced severe oscillations seen during the verification tests because of the time discretisation issue. In ARCO-CIET and the Type I Digital Twin, temperature samples were taken every 0.1 seconds when using digital control. Therefore, it behaved more like a discretised (digital) time system simulation rather than a continuous time system simulation. This caused oscillatory behaviour within the Type I Digital Twin of the heater during early iterations of the PID controller shown in Figure 6.26:

Figure 6.26: Oscillatory Behaviour Observed during Early Iterations of PID Control based SNF

Moreover, Figure 6.26 showed inherent limitations in having the heater power at a maximum of 10 kW. This is because transients were performed on the heater when it was operating at a steady state of 8 kW. At 8 kW, the heater's behaviour was validated with experimental data. Therefore, I could perform transients from this steady state knowing that the heater behaved reasonably according to experimental data. Ideally, I would have operated the heater at a lower power and flowrate, around 4 or 5 kW, and performed transient studies from that state. However, this would require a different set of experimental validation studies for the Type I Digital Twin. I left this to future work as this did not directly contribute to the purpose of this dissertation.

Now, to remove this oscillatory behaviour, I decided to follow Seborg's suggestion and place the derivative element within the feedback portion of the control loop in a PI - PD version of the PID controller [Seborg et al., 2016]. The block diagram of which is shown in Figure 6.27:

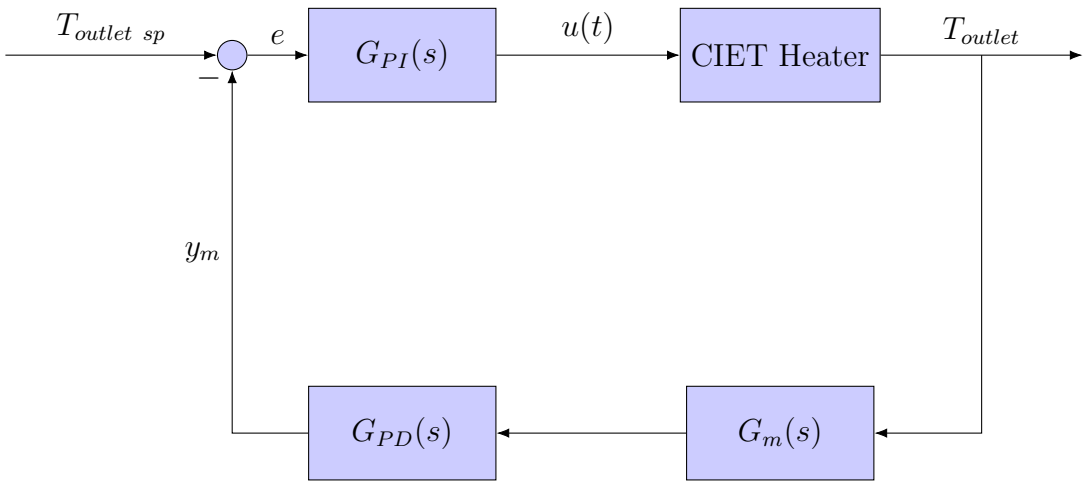


Figure 6.27: PI-PD Feedback Controller Diagram

In Figure 6.27, $T_{outlet\ sp}$ is the outlet temperature set point determined by the SNF transfer function. T_{outlet} is the Heater Outlet Temperature BT-12. $G_m(s)$ is the measurement transfer function which should be 1 if we were to ignore the lags and delays between OPC-UA server and client. Based on Figure 6.27, I define a transfer function for the PI and PD controller:

$$G_{PD} = \frac{3.339s}{0.1 \times 3.339s + 1} + 1 \tag{6.33}$$

$$G_{PI} = 80.0\frac{watts}{K}\left(\frac{1}{7s} + 1\right) \tag{6.34}$$

In both equations 6.33 and 6.34, the proportional and integral time were in units of seconds. Both equations 6.33 and 6.34 were initially estimated using Chien and Fruehauf's internal model control (IMC) tuning correlations [Fruehauf, Chien, and Lauritsen, 1994] for PID controller tunings found in textbooks [Seborg et al., 2016]. While Chien and Fruehauf's correlations were meant for PID controllers in parallel forms, they served as decent initial estimates for the PI-PD controller that I developed. This PI-PD controller approach worked well in removing oscillations within the outlet temperature. Early versions of the PI-PD version of the PID controller showed that the controller was able to get the outlet temperature of BT-12 to follow the desired output given by the transfer function at least for the longer term. However, the PI-PD controller was not able to reject the initial disturbances in the simulated BT-12 temperatures due to increased heater inlet temperatures (BT-11). This is shown in Figure 6.28:

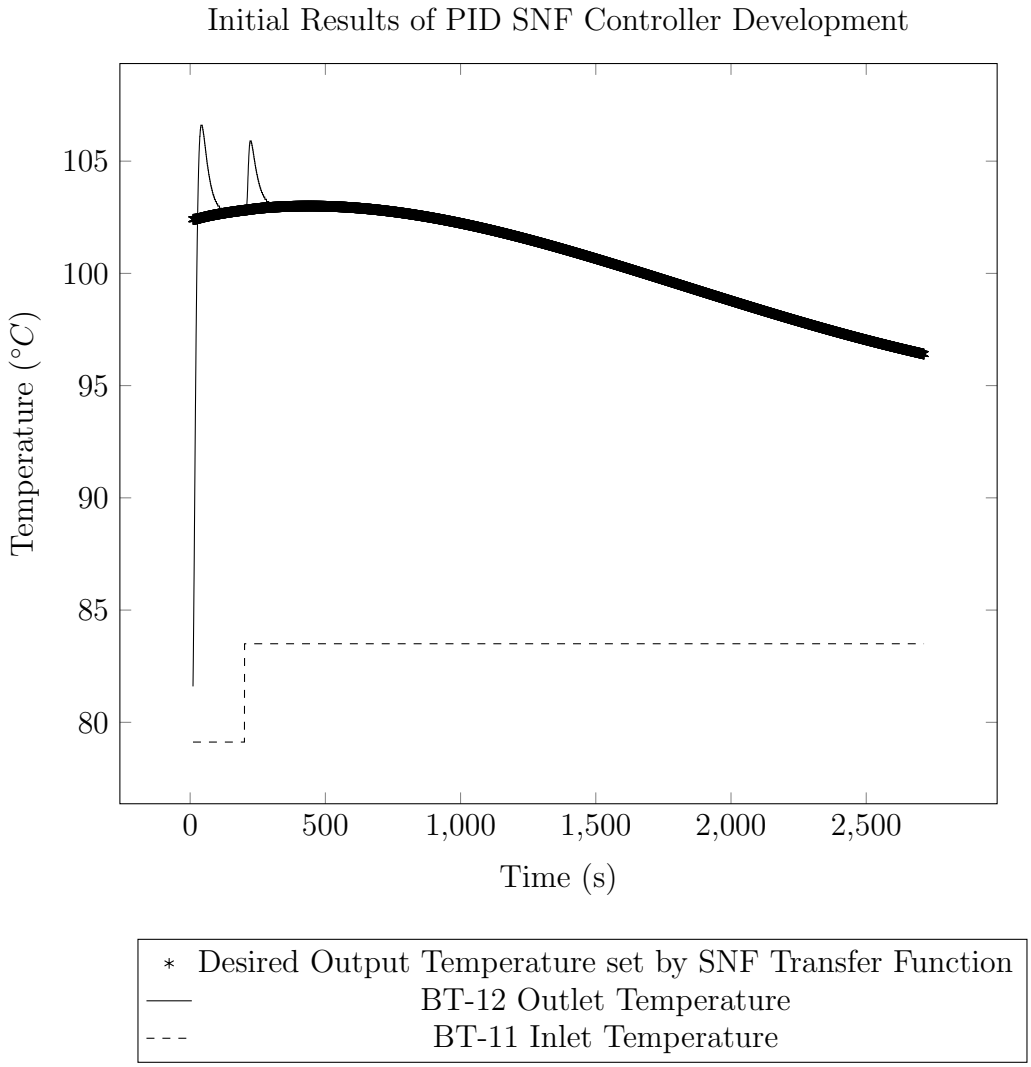


Figure 6.28: Initial Postprocessing Results for PI-PD Controller

The initial overshoot was due to the heater requiring some settling time as it reached an initial steady state. The latter overshoot was due to an increase in heater inlet temperature where the feedback controller was unable to reject the disturbance. Moreover, the transient timescale in Figure 6.28 was wrong because of a bug. The simulated reactivity feedback transfer function was mistakenly using the difference between the heater outlet temperature and steady state inlet temperature reading $T_{outlet} - T_{inlet\ steady\ state}$ to calculate the desired heater outlet temperature. This means that the transient shown was a result of mistakenly overpredicting simulated reactivity feedback. Thus, the transient time scale in Figure 6.28 was significantly shorter than the timescale of the correct SNF behaviour. For correct SNF behaviour, we should expect the time scale to dip after several hours rather than within one hour.

Despite this, Figure 6.28 showed that the PI-PD SNF controller was able to keep pace with the calculated set point from the SNF transfer function scaled down from GeN-Foam for the most part of the transient. Therefore, aside from the $T_{outlet} - T_{inlet\ steady\ state}$ issue, I was largely satisfied with the design. I then took some action to correct the $T_{outlet} - T_{inlet\ steady\ state}$ issue. This was largely successful, and a second transient was plotted in Figure 6.29:

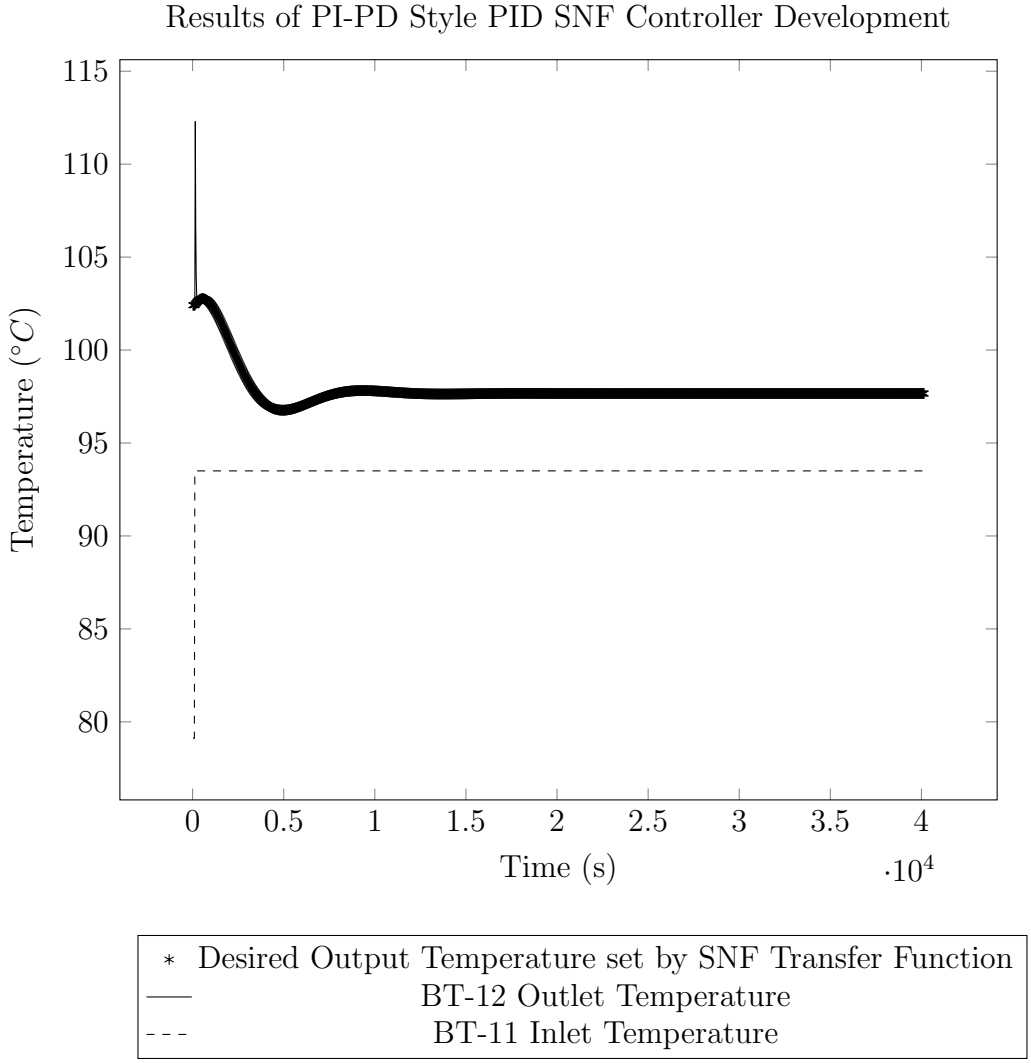


Figure 6.29: Postprocessing Results for PI-PD Controller

Figure 6.29 shows the correct SNF behaviour over the correct timescale. Other than the inability of the controller to reject the initial increase in outlet temperature, the SNF controller has generally performed well. While, I could make a third design iteration to eliminate the initial increase in outlet temperature, designing a SNF controller that perfectly mimics the SNF behaviour is not necessary for this work. Instead, I was only aiming to show that this iterative design process for the SNF controller was expedited by the Type I Digital

Twin. This goal has been achieved, and therefore, I decided to leave the actual design of the SNF controller to future work.

6.4 Future Work

Based on the experiences of the two previous design iterations of the SNF controller, we can clearly see that the Type I Digital Twin of CIET's heater performs a vital role in expediting the process. Given that we now have the tools with which to iteratively design the SNF controller, we could perform more design iterations so that the SNF controller is able to reject the initial increase in outlet temperature due to increase in heater inlet temperature. Additionally, we might explore other designs which incorporate control rod movement and other feedback mechanisms into the SNF controller.

Moreover, based on the time scale of the SNF transient, we may even upgrade the arbitrary FHR simulation to include decay heat and Xenon-135 transients, or even base these simulations on the gFHR [Kile et al., 2022]. Additionally, the surrogate models we use for modelling SNF behaviour could also be improved beyond transfer functions and state space models. We could use something more sophisticated such as neural networks or even a low fidelity deterministic lattice code.

Chapter 7

Conclusion

7.1 TL;DR

In this dissertation, we have discussed the construction of Type I Digital Twin libraries for incompressible flow heat transfer and validated the Type I Digital Twin of CIET's Heater against experimental data. This Digital Twin was then used as a testbed for iterative development of simulated neutronics feedback (SNF) controllers which could, at least partially, simulate reactor thermal inertia as well. We have also demonstrated a methodology of constructing a surrogate model based on a higher fidelity deterministic multiphysics simulation in GeN-Foam for use in the SNF controller. To simulate this SNF controller, a transfer function and control systems library was built in the Rust programming language. We also have demonstrated code to code verification with this control systems library using Scilab to show that the library was functioning correctly for this dissertation. With these tools, we demonstrated the capability of the testbed in expediting the development process of the SNF controller in Integral Effects Tests (IET) facilities. This is important because the Type I Digital Twin has helped speed up the initial iterative development process for SNF Controllers for use in electrically heated IET facilities.

7.2 Future Work

The work in this dissertation is useful for designing SNF Controllers within electrically heated IET facilities such as the Compact Integral Effects Test (CIET). In the longer term, these SNF Controllers were meant to be coupled with experimental facilities in real-time so that the IETs can be used for transient tests with SNF. Experimental data from these tests can help us ascertain the impact of transients on the primary loop. Additionally, if we wanted to ascertain the impact of certain transients on the reactor vessel and fuel in real-time, we could improve upon this hardware in the loop simulation such that the reactor vessel and its components are simulated in real-time. This is similar to how a Type III Digital Twin would be coupled both ways in real-time with the IET hardware. However, the model being

synchronised in real-time with the IET hardware is a real-time reactor model rather than the IET model. Building and synchronising such a reactor model with its IET hardware is a much more daunting task than a simple SNF Controller. For such a task, using a Type I Digital Twin test bed for iterative model development will prove to be essential.

Bibliography

Zweibaum, Nicolas (2015). *Experimental Validation of Passive Safety System Models: Application to Design and Optimization of Fluoride-Salt-Cooled, High-Temperature Reactors*. University of California, Berkeley. ISBN: 1339216086.

Lichtenstein, T et al. (2022). *Thermochemical Property Measurements of FLiNaK and FLiBe in FY 2020*. Tech. rep. Argonne National Lab.(ANL), Argonne, IL (United States).

Ong, Theodore Kay Chen (2023). "Development of an Isothermal-Flow Digital Twin for the Compact Integral Effects Test". MA thesis. University of California, Berkeley.

De Wet, Dane and Per F Peterson (2020). "A Frequency Domain Approach to Characterizing and Modeling Single Phased, Forced Circulation Advanced Nuclear Reactor Designs". Dissertation. University of California, Berkeley.

Kile, Robert et al. (2022). *Initial Development of a Generic Fluoride Salt-Cooled Reactor Model*. Tech. rep. Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States).

Larsen, Kristian Pontoppidan (Oct. 2022). *graphreader.com - Online tool for reading graph image values and save as CSV JSON*. Webpage. URL: `http://www.graphreader.com/`.

Chadwick, M.B. et al. (Dec. 2006). "ENDF/B-VII.0: Next Generation Evaluated Nuclear Data Library for Nuclear Science and Technology". In: *Nuclear Data Sheets* 107.12. Ed. by P. Obložinský and M. Herman, pp. 2931–3118. DOI: `10.1016/j.nds.2006.11.001`. URL: `https://www.sciencedirect.com/science/journal/00903752`.

Merzlikina, EI and OG Prochina (2020). "Laboratory works on control theory using scilab/xcos". In: *2020 V International Conference on Information Technologies in Engineering Education (Inforino)*. IEEE, pp. 1–4.

Zou, Ling, Rui Hu, and Anne Charpentier (2019). *SAM Code Validation using the Compact Integral Effects Test (CIET) Experimental Data*. Tech. rep. Argonne National Lab.(ANL), Argonne, IL (United States).

Parker, WJ et al. (1961). "Flash method of determining thermal diffusivity, heat capacity, and thermal conductivity". In: *Journal of applied physics* 32.9, pp. 1679–1684.

Zhao, Dong and Fenghua Guan (2022). "Research and Application Innovation of Digital Twin Technologies for Key Equipment of Nuclear Power Plants". In: *International Conference on Nuclear Engineering*. Vol. 86472. American Society of Mechanical Engineers, V012T12A044.

Helmuth, John A. (Oct. 1988). "Nuclear Power Plant Capital Costs and Turnkey Estimates". en. In: *The American Economist* 32.2, pp. 66–70. ISSN: 0569-4345, 2328-1235. DOI: 10.1177/056943458803200211. URL: http://journals.sagepub.com/doi/10.1177/056943458803200211 (visited on 06/22/2023).

Bickel, Jeffrey E, Nicholas Zweibaum, and Per F Peterson (2014). *Design, Fabrication and Startup Testing in the Compact Integral Effects Test (CIET 1.0) Facility in Support of Fluoride-Salt-Cooled, High-Temperature Reactor Technology*. University of California, Berkeley.

Blandford, Edward et al. (2020). "Kairos power thermal hydraulics research and development". In: *Nuclear Engineering and Design* 364, p. 110636.

Wang, Xin (2018). "Coupled neutronics and thermal-hydraulics modeling for pebble-bed Fluoride-Salt-Cooled, High-Temperature Reactor (FHR)". PhD thesis. URL: internal-pdf://236.20.125.107/Coupled%20neutronics%20and%20thermal-hydraulics%20mode.pdf.

Kholopov, VA et al. (2019). "Digital twins in manufacturing". In: *Russian Engineering Research* 39.12, pp. 1014–1020.

Bardet, Philippe M and Per F Peterson (Sept. 2008). "Options for Scaled Experiments for High Temperature Liquid Salt and Helium Fluid Mechanics and Convective Heat Transfer". In: *Nuclear Technology* 163.3, pp. 344–357. ISSN: 0029-5450. DOI: 10.13182/NT163-344. URL: https://doi.org/10.13182/NT163-344.

De wet, Dane, Per F. Peterson, and Michael Scott Greenwood (Aug. 2019). *A FREQUENCY RESPONSE APPROACH TO MODEL VALIDATION FOR THE COMPACT INTEGRAL EFFECTS TEST FACILITY IN TRANSFORM*. English. Tech. rep. URL: https://www.osti.gov/biblio/1570910 (visited on 06/23/2023).

Sohal, Manohar S et al. (2010). *Engineering database of liquid salt thermophysical and thermochemical properties*. Tech. rep. Idaho National Lab.(INL), Idaho Falls, ID (United States).

Romatoski, Rebecca Rose and Lin-Wen Hu (2017). "Fluoride salt coolant properties for nuclear reactor applications: A review". In: *Annals of Nuclear Energy* 109, pp. 635–647.

Perry, Robert H and Don W Green (2015). "Perry's chemical engineers' handbook". In: *Mc Graw*.

Bejan, Adrian (2013). *Convection heat transfer*. John wiley & sons. ISBN: 1118330080.

Todreas, Neil E, Mujid S Kazimi, and Mahmoud Massoud (2021). *Nuclear systems Volume II: Elements of thermal hydraulic design*. CRC Press.

BIPM, IFCC et al. (2008). "Evaluation of measurement data—guide to the expression of uncertainty in measurement, JCGM 100: 2008 GUM 1995 with minor corrections". In: *Joint Committee for Guides in Metrology*.

Johnson, Ishak Mudrikah Banin (2022). *Quantifying Radiative Heat Transfer Scaling Distortions: System Code Development and a Scaling Methodology for Fluoride-Salt-Cooled High-Temperature Reactors*. University of California, Berkeley.

Zweibaum, N, Z Guo, et al. (2016). "Design of the Compact Integral Effects Test Facility and Validation of Best-Estimate Models for Fluoride Salt–Cooled High-Temperature Reactors". In: *Nuclear Technology* 196.3, pp. 641–660. ISSN: 0029-5450. DOI: 10.13182/NT16-15. URL: https://doi.org/10.13182/NT16-15.

Al-Geddawy, Tarek (2020). "A Digital Twin Creation Method for an Opensource Low-cost Changeable Learning Factory". In: *Procedia Manufacturing* 51, pp. 1799–1805.

Sleiti, Ahmad K, Jayanta S Kapat, and Ladislav Vesely (2022). "Digital twin in energy industry: Proposed robust digital twin for power plant and other complex capital-intensive large engineering systems". In: *Energy Reports* 8, pp. 3704–3726. ISSN: 2352-4847.

Kochunas, Brendan and Xun Huan (2021). "Digital twin concepts with uncertainty for nuclear power applications". In: *Energies* 14.14, p. 4235.

Van der Valk, Hendrik et al. (2020). "A Taxonomy of Digital Twins." In: *AMCIS*.

Rasheed, Adil, Omer San, and Trond Kvamsdal (2020). "Digital Twin: Values, Challenges and Enablers From a Modeling Perspective". In: *IEEE Access* 8, pp. 21980–22012. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2970143.

Boschert, Stefan and Roland Rosen (2016). "Digital twin—the simulation aspect". In: *Mechatron. Futur.* Springer, pp. 59–74.

Enders, Martin Robert and Nadja Hoßbach (2019). "Dimensions of digital twin applications-a literature review". In.

Grieves, Michael and John Vickers (2017). "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems". In: *Transdisciplinary perspectives on complex systems*. Springer, pp. 85–113.

Eckhart, Matthias and Andreas Ekelhart (May 22, 2018). "Towards Security-Aware Virtual Environments for Digital Twins". In: *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security*. ACM, pp. 61–72. DOI: 10.1145/3198458.3198464.

Jain, Palak et al. (2019). "A Digital Twin Approach for Fault Diagnosis in Distributed Photovoltaic Systems". In: *IEEE Transactions on Power Electronics* 35.1, pp. 940–956. DOI: 10.1109/tpel.2019.2911594.

Semeraro, Concetta et al. (2021). "Digital twin paradigm: A systematic literature review". In: *Computers in Industry* 130, p. 103469.

Poresky, Christopher et al. (Sept. 2022). "Advanced Reactor Control and Operations (ARCO): A University Research Facility for Developing Optimized Digital Control Rooms". In: *Nuclear Technology* 0.0, pp. 1–12. ISSN: 0029-5450. DOI: 10.1080/00295450.2022.2092366. (Visited on 10/05/2022).

Oroulet et al. (Oct. 2022). *opcua-asyncio*. Github. original-date: 2018-08-02T07:45:42Z. URL: https://github.com/FreeOpcUa/opcua-asyncio (visited on 10/04/2022).

Zidek, Kamil et al. (2020). "Digital twin of experimental smart manufacturing assembly system for industry 4.0 concept". In: *Sustainability* 12.9, p. 3658.

Oberg, J. (Dec. 1999). "Why the Mars probe went off course [accident investigation]". In: *IEEE Spectrum* 36.12, pp. 34–39. ISSN: 1939-9340. DOI: 10.1109/6.809121.

Boutin, Mike (June 2023). *Units of measurement (uom) v0.35.0*. Rust Crate io. URL: https://crates.io/crates/uom.

Saligrama, Aditya, Andrew Shen, and Jon Gjengset (2019). "A Practical Analysis of Rust's Concurrency Story". In: *arXiv preprint arXiv:1904.12210*.

Casella, Francesco, Alberto Leva, et al. (2003). "Modelica open library for power plant simulation: design and experimental validation". In: *Proceeding of the 2003 Modelica conference, Linkoping, Sweden*. Citeseer.

MacNamara, Shev and Gilbert Strang (2016). "Operator Splitting". en. In: ed. by Roland Glowinski, Stanley J. Osher, and Wotao Yin. Scientific Computation. Cham: Springer International Publishing, pp. 95–114. ISBN: 9783319415895. DOI: 10.1007/978-3-319-41589-5_3. (Visited on 10/04/2022).

Novak, April (2020). *Multiscale Thermal-Hydraulic Methods for Pebble Bed Reactors*. University of California, Berkeley.

Poresky, Christopher (2017). *Frequency response testing in the ciet facility*. Berkeley, CA.

Lukas, Raleigh, James Kendrick, and Per Peterson (July 2017). "Improved Heat Transfer and Volume Scaling through Novel Heater Design". English. In: *Transactions of the American Nuclear Society* 116. ISSN: 0003-018X. URL: https://www.osti.gov/biblio/23050429 (visited on 06/23/2023).

Graves, RS et al. (1991). "The thermal conductivity of AISI 304L stainless steel". In: *International journal of thermophysics* 12, pp. 409–415.

Gnielinski, Volker (2013). "On heat transfer in tubes". In: *International Journal of Heat and Mass Transfer* 63, pp. 134–140.

Ergun, Sabri and Ao Ao Orning (1949). "Fluid flow through randomly packed columns and fluidized beds". In: *Industrial & Engineering Chemistry* 41.6, pp. 1179–1184.

Wakao, N, S Kaguei, and T Funazkri (1979). "Effect of fluid dispersion coefficients on particle-to-fluid heat transfer coefficients in packed beds: Correlation of nusselt numbers". In: *Chemical Engineering Science* 34.3, pp. 325–336. ISSN: 0009-2509. DOI: https://doi.org/10.1016/0009-2509(79)85064-2. URL: http://www.sciencedirect.com/science/article/pii/0009250979850642.

Churchill, Stuart W (1977). "Friction-factor equation spans all fluid-flow regimes." In.

Zweibaum, N, J E Bickel, et al. (2015). "Design, fabrication and startup testing in the compact integral effects test facility in support of fluoride-saltcooled, high-temperature reactor technology". In: *Int. Top. Meet. Nucl. React. Therm. Hydraul. (NURETH-16), Chicago, IL, August*.

Hartnett, James P (1955). "Experimental determination of the thermal-entrance length for the flow of water and of oil in circular pipes". In: *Transactions of the American Society of Mechanical Engineers* 77.8, pp. 1211–1218.

Scarlat, Raluca Olga (2012). *Design of complex systems to achieve passive safety: Natural circulation cooling of liquid salt pebble bed reactors*. University of California, Berkeley.

Rossi, Diogo Folador et al. (2014). "A review of automatic time-stepping strategies on numerical time integration for structural dynamics analysis". In: *Engineering structures* 80, pp. 118–136.

Salem, Rami, Marc Errera, and Julien Marty (2019). "Adaptive diffusive time-step in conjugate heat transfer interface conditions for thermal-barrier-coated applications". In: *International Journal of Thermal Sciences* 145, p. 106048.

Aumiller, DL, ET Tomlinson, and RC Bauer (2001). "A coupled RELAP5-3D/CFD methodology with a proof-of-principle calculation". In: *Nuclear Engineering and Design* 205.1-2, pp. 83–90.

Fiorina, Carlo, Ivor Clifford, et al. (2015). "GeN-Foam: a novel OpenFOAM® based multi-physics solver for 2D/3D transient analysis of nuclear reactors". In: *Nuclear Engineering and Design* 294, pp. 24–37. ISSN: 0029-5493. URL: `internal-pdf://204.178.192.230/GeN-Foam%20a%20novel%20openfoam%20based%20multiphysics%20s.pdf`.

Duan, Ran, Hongxun Wu, and Renfei Zhou (2022). "Faster matrix multiplication via asymmetric hashing". In: *arXiv preprint arXiv:2210.10173*.

Fernández, Miguel A (2011). "Coupling schemes for incompressible fluid-structure interaction: implicit, semi-implicit and explicit". In: *SeMA Journal* 55.1, pp. 59–108.

Pautasso, Cesare and Erik Wilde (2009). "Why is the web loosely coupled? A multi-faceted metric for service design". In: *Proceedings of the 18th international conference on World wide web*, pp. 911–920.

Vera, Marcos and Alberto E Quintero (2018). "On the role of axial wall conduction in mini/micro counterflow heat exchangers". In: *International Journal of Heat and Mass Transfer* 116, pp. 840–857.

Robert, Yves et al. (2023). "Impact of thermal coupling on a pebble bed reactor equilibrium from hyper-fidelity depletion". In: *Proceedings of the 2023 30th International Conference on Nuclear Engineering ICONE30*. Kyoto, Japan.

Xianyi, Zhang, Wang Qian, and Zhang Yunquan (2012). "Model-driven level 3 BLAS performance optimization on Loongson 3A processor". In: *2012 IEEE 18th international conference on parallel and distributed systems*. IEEE, pp. 684–691.

Liu, Yang (2020). "Maximizing the CFL number of stable time–space domain explicit finite-difference modeling". In: *Journal of Computational Physics* 416, p. 109501.

OpenFOAM (Oct. 2023). *OpenFOAM: User Guide v2112 CourantNo*. website. Definiton of Courant Number. URL: `https://www.openfoam.com/documentation/guides/latest/doc/guide-fos-field-CourantNo.html`.

Rauter, Matthias et al. (2021). "Numerical simulation of impulse wave generation by idealized landslides with OpenFOAM". In: *Coastal Engineering* 165, p. 103815.

Hensen, Jan LM and Abdullatif E Nakhi (1994). "Fourier and Biot numbers and the accuracy of conduction modelling". In: *Proceedings of BEP'94 Conference*, pp. 247–256.

Thomas, BG, IV Samarasekera, and J Keith Brimacombe (1984). "Comparison of numerical modeling techniques for complex, two-dimensional, transient heat-conduction problems". In: *Metallurgical Transactions B* 15, pp. 307–318.

Wesseling, Piet (1996). "von Neumann stability conditions for the convection-diffusion equation". In: *IMA journal of Numerical Analysis* 16.4, pp. 583–598.

Tadmor, Eitan (1987). “Stability analysis of finite difference, pseudospectral and Fourier–Galerkin approximations for time-dependent problems”. In: *SIAM review* 29.4, pp. 525–555.

Charney, Jules G, Ragnar Fjörtoft, and J von Neumann (1950). “Numerical integration of the barotropic vorticity equation”. In: *Tellus* 2.4, pp. 237–254.

Errera, Marc-Paul and Sébastien Chemin (2013). “Optimal solutions of numerical interface conditions in fluid–structure thermal analysis”. In: *Journal of Computational Physics* 245, pp. 431–455.

Maffulli, R et al. (2018). “Fast conjugate heat transfer simulation of long transient flexible operations using adaptive time stepping”. In: *Journal of Turbomachinery* 140.9, p. 091005.

Mohan, Ram V and Kumar K Tamma (1994). “Finite element/finite volume approaches with adaptive time stepping strategies for transient thermal problems”. In: *Sadhana* 19, pp. 765–783.

Alsharef, Ahmad et al. (2021). “Cache memory: an analysis on performance issues”. In: *2021 8th international conference on computing for sustainable global development (INDIACom)*. IEEE, pp. 184–188.

Knuth, Donald E (1974). “Structured programming with go to statements”. In: *ACM Computing Surveys (CSUR)* 6.4, pp. 261–301.

Cardelli, Luca and Peter Wegner (1985). “On understanding types, data abstraction, and polymorphism”. In: *ACM Computing Surveys (CSUR)* 17.4, pp. 471–523.

Weidendorfer, Josef (2008). “Sequential performance analysis with callgrind and kcachegrind”. In: *Tools for High Performance Computing: Proceedings of the 2nd International Workshop on Parallel Tools for High Performance Computing, July 2008, HLRS, Stuttgart*. Springer, pp. 93–113.

Lunnikivi, Henri, Kai Jylkkä, and Timo Hämäläinen (2020). “Transpiling Python to Rust for Optimized Performance”. In: *International Conference on Embedded Computer Systems*. Springer, pp. 127–138.

Xu, Ben, Pei-Wen Li, and Cho Lik Chan (2012). “Extending the validity of lumped capacitance method for large Biot number in thermal storage application”. In: *Solar Energy* 86.6, pp. 1709–1724.

Meyer, Veronika R (2007). “Measurement uncertainty”. In: *Journal of Chromatography A* 1158.1-2, pp. 15–24.

Çelebioğlu, Hasan Emrah (2005). “Developing a computer program for evaluating uncertainty of some typical dimensional measuring and gauging devices”. MA thesis. Middle East Technical University.

Trojan, M (2014). “Transient Heat Conduction in Semiinfinite Solid with Surface Convection”. In: *Encyclopedia of*.

Harris, Charles R et al. (2020). “Array programming with NumPy”. In: *Nature* 585.7825, pp. 357–362.

Locka99 (Sept. 2022). *OPC UA server and client API implementation for Rust.* Github. original-date: 2017-01-07T09:48:09Z. URL: https://github.com/locka99/opcua (visited on 10/04/2022).

vorot (Dec. 2022). *Library of well known algorithms for numerical root finding.* Github. URL: https://github.com/vorot/roots (visited on 12/17/2023).

Virtanen, Pauli et al. (Feb. 3, 2020). "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature Methods* 17.3, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Bouckaert, Stéphanie et al. (2021). "Net zero by 2050: A roadmap for the global energy sector". In.

Holcomb, David E et al. (2013). "Fluoride salt-cooled high-temperature reactor technology development and demonstration roadmap". In: *ORNL/TM-2013/401, ORNL, Oak Ridge, TN.*

Haubenreich, Paul N and J R Engel (1970). "Experience with the molten-salt reactor experiment". In: *Nuclear Applications and technology* 8.2, pp. 118–136. ISSN: 0550-3043.

Andreades, Charalampos et al. (Sept. 2016). "Design Summary of the Mark-I Pebble-Bed, Fluoride Salt–Cooled, High-Temperature Reactor Commercial Power Plant". In: *Nuclear Technology* 195.3, pp. 223–238. ISSN: 0029-5450. DOI: 10.13182/NT16-2. URL: https://doi.org/10.13182/NT16-2.

Doniger, William et al. (2023). "Corrosion of 316L & 316H stainless steel in molten LiF-NaF-KF (FLiNaK)". In: *Journal of Nuclear Materials* 579, p. 154383.

Jiang, Dianqiang et al. (2022). "Fluoride-salt-cooled high-temperature reactors: Review of historical milestones, research status, challenges, and outlook". In: *Renewable and Sustainable Energy Reviews* 161, p. 112345.

Allen, Todd et al. (2013). *Fluoride-Salt-Cooled, High-Temperature Reactor (FHR) Methods and Experiments Program White Paper.* Tech. rep. URL: internal-pdf://87.157.115.188/12-002-FHR-Workshop-2-Report-Final.pdf.

Bragg-Sitton, Shannon M, Thomas J Godfroy, and Kenny Webster (2010). "Improving the fidelity of electrically heated nuclear systems testing using simulated neutronic feedback". In: *Nuclear Engineering and Design* 240.10, pp. 2745–2754. ISSN: 0029-5493. DOI: https://doi.org/10.1016/j.nucengdes.2010.04.036. URL: http://www.sciencedirect.com/science/article/pii/S0029549310003201.

Bragg-Sitton, S M and K L Webster (2007). "Application of Simulated Reactivity Feedback in Nonnuclear Testing of a Direct-Drive Gas-Cooled Reactor". In.

Bragg-Sitton, Shannon M and Matthew Forsbacka (2004). "Application of a virtual reactivity feedback control loop in non-nuclear testing of a fast spectrum reactor". In.

Kok, H V and THJJ Van der Hagen (1999). "Design of a simulated void-reactivity feedback in a boiling water reactor loop". In: *Nuclear technology* 128.1, pp. 1–11. ISSN: 0029-5450. URL: internal-pdf://0496842819/Design%20of%20a%20Simulated%20Void%20Reactivity%20Feedback.pdf.

Furuya, Masahiro, Takanori Fukahori, and Shinya Mizokami (2007). "Development of BWR regional stability experimental facility SIRIUS-F, which simulates thermal

hydraulics-neutronics coupling, and stability evaluation of ABWRs". In: *Nuclear technology* 158.2, pp. 191–207. ISSN: 0029-5450. URL: internal-pdf://239.46.61.132/DEVELO%7B~%7D1.PDF.

Chen, Hanying et al. (2017). "Experimental simulation study of transition transients between forced circulation and natural circulation with reactivity feedback". In: *International Electronic Journal of Nuclear Safety and Simulation* 8.1, pp. 42–49. URL: internal-pdf://193.60.43.224/EXPERI%7B~%7D2.PDF.

Shi, Shanbin et al. (2015). "Experimental study of natural circulation instability with void reactivity feedback during startup transients for a BWR-type SMR". In: *Progress in Nuclear Energy* 83, pp. 73–81. ISSN: 0149-1970. URL: internal-pdf://104.76.84.186/EXPT%20study%20BWR%20SMR%20void%20reactivity%20feedback%20Sh.pdf%20internal-pdf://3690031639/EXPT%20study%20BWR%20SMR%20void%20reactivity%20feedback%20S1.pdf.

Marcel, Christian P, M Rohde, and THJJ Van Der Hagen (2017). "An experimental parametric study on natural circulation BWRs stability". In: *Nuclear Engineering and Design* 318, pp. 135–146. ISSN: 0029-5493. URL: internal-pdf://0649814875/An%20experimental%20parametric%20study%20on%20natural%20ci.pdf.

Duderstadt, James J and Louis J Hamilton (1976). *Nuclear reactor analysis*. Vol. 84. Wiley New York. URL: internal-pdf://204.15.202.224/Duderstadt%7B%5C_%7DHamilton%7B%5C_%7D1976%20NE150%7B%5C_%7D250.pdf.

Lewins, Jeffery (1960). "A derivation of the time-dependent adjoint equations for neutron importance in the transport, continuous slowing-down and diffusion models". In: *Journal of Nuclear Energy. Part A. Reactor Science* 13.1-2, pp. 1–5.

Stewart, Ryan et al. (2022). "Generation of localized reactor point kinetics parameters using coupled neutronic and thermal fluid models for pebble-bed reactor transient analysis". In: *Annals of Nuclear Energy* 174, p. 109143.

Aboanber, Ahmed E, Abdallah A Nahla, and Zeid I Al-Muhiameed (2014). "A novel mathematical model for two-energy groups of the point kinetics reactor dynamics". In: *Progress in Nuclear Energy* 77, pp. 160–166.

Chinesta, Francisco et al. (2016). "Model order reduction". In: *Encyclopedia of computational mechanics*.

Schilders, Wilhelmus HA, Henk A Van der Vorst, and Joost Rommes (2008). *Model order reduction: theory, research aspects and applications*. Vol. 13. Springer.

Van De Graaf, Rudi, THJJ Van Der Hagen, and Robert F Mudde (1994). "Scaling laws and design aspects of a natural-circulation-cooled simulated boiling water reactor fuel assembly". In: *Nuclear technology* 105.2, pp. 190–200. ISSN: 0029-5450.

Frangos, Michalis et al. (2010). "Surrogate and reduced-order modeling: a comparison of approaches for large-scale statistical inverse problems". In: *Large-Scale Inverse Problems and Quantification of Uncertainty*, pp. 123–149.

Asher, Michael J et al. (2015). "A review of surrogate models and their application to groundwater modeling". In: *Water Resources Research* 51.8, pp. 5957–5973.

Alizadeh, Reza, Janet K Allen, and Farrokh Mistree (2020). "Managing computational complexity using surrogate models: a critical review". In: *Research in Engineering Design* 31, pp. 275–298.

Peterson, Per F (1994). "Scaling and analysis of mixing in large stratified volumes". In: *International journal of heat and mass transfer* 37, pp. 97–106.

Romano, Paul K and Benoit Forget (2013). "The OpenMC Monte Carlo particle transport code". In: *Annals of Nuclear Energy* 51, pp. 274–281. ISSN: 0306-4549. DOI: https://doi.org/10.1016/j.anucene.2012.06.040. URL: http://www.sciencedirect.com/science/article/pii/S0306454912003283%20internal-pdf://138.230.237.185/The%20OpenMC%20Monte%20Carlo%20particle%20transport%20code.pdf.

Romano, Paul K, Nicholas E Horelik, et al. (2015). "OpenMC: A state-of-the-art Monte Carlo code for research and development". In: *Annals of Nuclear Energy* 82, pp. 90–97. ISSN: 0306-4549. DOI: https://doi.org/10.1016/j.anucene.2014.07.048. URL: http://www.sciencedirect.com/science/article/pii/S030645491400379X.

Larsen, Edward W, Jim E Morel, and John M McGhee (1993). "Asymptotic derivation of the simplified PN equations". In: *Proceedings of the Joint International Conference on Mathematical Methods and Supercomputing in Nuclear Applications*. Vol. 1, p. 718.

Modest, Michael and Shenghui Lei (June 2012). "The Simplified Spherical Harmonics Method For Radiative Heat Transfer". In: *Journal of Physics Conference Series* 369, p. 2019. DOI: 10.1088/1742-6596/369/1/012019.

Chan, Te-Chang (1971). "Reactor transfer function measurements with the reactor oscillator". MA thesis. Iowa State University.

Ball, SJ and TW Kerlin (1965). *Stability analysis of the molten-salt reactor experiment*. Tech. rep. Oak Ridge National Lab., Tenn.

Legua, Matilde P, Isabel Morales, and Luis M Sánchez Ruiz (2008). "The heaviside step function and MATLAB". In: *Computational Science and Its Applications–ICCSA 2008: International Conference, Perugia, Italy, June 30–July 3, 2008, Proceedings, Part I 8*. Springer, pp. 1212–1221.

Kerlin, T (2012). *Frequency response testing in nuclear reactors*. Vol. 11. Elsevier. ISBN: 0323160514.

Schmid, Paul (1957). *Absolute Reactivity Measurement from Transient Behaviour of a Subcritical Nuclear Reactor*. Tech. rep. Reaktor AG, Wurenlingen, Switzerland.

Amendola, A et al. (1982). "Dynamic and static models for nuclear reactor operators-needs and application examples". In: *IFAC Proceedings Volumes* 15.6, pp. 103–110.

Hainoun, A and I Khamis (2000). "Determination of neutron generation time in miniature neutron source reactor by measurement of neutronics transfer function". In: *Nuclear engineering and design* 195.3, pp. 299–305.

Sanathanan, CK and Hideo Tsukui (1974). "Synthesis of transfer function from frequency response data". In: *International Journal of Systems Science* 5.1, pp. 41–54.

Oberhettinger, Fritz and Larry Badii (2012). *Tables of Laplace transforms*. Springer Science & Business Media.

Robinson, JC and DN Fry (1970). "Experimental neutron flux-to-pressure frequency response for the molten-salt reactor experiment: determination of void fraction in fuel salt". In: *Nuclear Science and Engineering* 42.3, pp. 397–405.

Rhodes, William D, Raymond V Furstenau, and Howard A Larson (2000). "Experimental Breeder Reactor–II Frequency Response Test Measurements Via Pseudorandom, Discrete-Level Binary and Ternary Signals". In: *Nuclear technology* 130.2, pp. 145–158.

Ljung, Lennart and Rajiv Singh (2012). "Version 8 of the MATLAB system identification toolbox". In: *IFAC Proceedings Volumes* 45.16, pp. 1826–1831.

Valério, Duarte, Manuel Duarte Ortigueira, and José Sá da Costa (2008). "Identifying a transfer function from a frequency response". In: *Journal of computational and nonlinear dynamics* 3.2.

Kollár, István, Rik Pintelon, and Johan Schoukens (1991). "Frequency domain system identification toolbox for MATLAB". In: *IFAC Proceedings Volumes* 24.3, pp. 1243–1247.

Anderson, George C, Brian W Finnie, and Gordon T Roberts (1967). "Pseudo-random and random test signals". In: *Hewlett-Packard Journal* 19.1, pp. 2–17.

Loewe, William E (1965). "Space-dependent effects in the response of a nuclear reactor to forced oscillations". In: *Nuclear Science and Engineering* 21.4, pp. 536–549.

Wang, Guoxu et al. (2017). "State-space model predictive control method for core power control in pressurized water reactor nuclear power stations". In: *Nuclear Engineering and Technology* 49.1, pp. 134–140.

OpenAI (Mar. 2023). *ChatGPT*. URL: chat.openai.com/.

Gong, Helin, Sibo Cheng, Zhang Chen, and Qing Li (2022). "Data-enabled physics-informed machine learning for reduced-order modeling digital twin: application to nuclear reactor physics". In: *Nuclear Science and Engineering* 196.6, pp. 668–693.

Gong, Helin, Sibo Cheng, Zhang Chen, Qing Li, et al. (2022). "An efficient digital twin based on machine learning SVD autoencoder and generalised latent assimilation for nuclear reactor physics". In: *Annals of Nuclear Energy* 179, p. 109431.

Zeng, Yuyun et al. (Mar. 2018). "Machine learning based system performance prediction model for reactor control". en. In: *Annals of Nuclear Energy* 113, pp. 270–278. ISSN: 0306-4549. DOI: 10.1016/j.anucene.2017.11.014. URL: https://www.sciencedirect.com/science/article/pii/S030645491730395X (visited on 05/24/2023).

Baur, Ulrike, Peter Benner, and Lihong Feng (2014). "Model order reduction for linear and nonlinear systems: a system-theoretic perspective". In: *Archives of Computational Methods in Engineering* 21.4, pp. 331–358.

Quarteroni, Alfio, Gianluigi Rozza, et al. (2014). *Reduced order methods for modeling and computational reduction*. Vol. 9. Springer.

Weiss, Julien (2019). "A tutorial on the proper orthogonal decomposition". In: *AIAA aviation 2019 forum*, p. 3333.

Gugercin, Serkan and Athanasios C Antoulas (2004). "A survey of model reduction by balanced truncation and some new results". In: *International Journal of Control* 77.8, pp. 748–766.

Gonzalez, Stephanie (2018). "Approximating the solutions of Lyapunov equations for balanced truncation". PhD thesis.

Antoulas, Athanasios C (2005). *Approximation of large-scale dynamical systems*. SIAM.

Elzohery, Rabab (2022). "On model-order reduction in neutronic systems via POD-galerkin projection". PhD thesis. Kansas State University.

Zarei, M (2021). "On a reduced order modeling of the nuclear reactor dynamics". In: *Applied Mathematics and Computation* 393, p. 125819.

German, Peter, Jean C Ragusa, and Carlo Fiorina (2019). "Application of multiphysics model order reduction to doppler/neutronic feedback". In: *EPJ Nuclear Sciences & Technologies* 5.ARTICLE, p. 17.

Simoncini, Valeria and Daniel B Szyld (2007). "Recent computational developments in Krylov subspace methods for linear systems". In: *Numerical Linear Algebra with Applications* 14.1, pp. 1–59.

Ipsen, Ilse CF and Carl D Meyer (1998). "The idea behind Krylov methods". In: *The American mathematical monthly* 105.10, pp. 889–899.

Breiten, Tobias and Tobias Damm (2010). "Krylov subspace methods for model order reduction of bilinear control systems". In: *Systems & Control Letters* 59.8, pp. 443–450.

Benner, Peter, Tobias Breiten, and Tobias Damm (2010). "Krylov Subspace Methods for Model Order Reduction of Bilinear Discrete-Time Control Systems". In: *PAMM* 10.1, pp. 601–602.

Salimbahrami, Behnam and Boris Lohmann (2002). "Krylov subspace methods in linear model order reduction: introduction and invariance properties". In: *Scientific Report*.

Bai, Zhaojun (2002). "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems". In: *Applied numerical mathematics* 43.1-2, pp. 9–44.

Feng, Lihong (2005). "Review of model order reduction methods for numerical simulation of nonlinear circuits". In: *Applied Mathematics and Computation* 167.1, pp. 576–591.

Chen, Yong (1999). "Model order reduction for nonlinear systems". PhD thesis. Massachusetts Institute of Technology.

Popov, Emilian et al. (2022). "Artificial intelligence-driven thermal design for additively manufactured reactor cores". In: *Nuclear Engineering and Design* 395, p. 111862.

Haubenreich, P N (1969). *MOLTEN SALT REACTOR EXPERIMENT.* Tech. rep.

Chen Qingyan, Wei Zhang (Apr. 2000). "Large Eddy Simulation of Natural and Mixed Convection Airflow Indoors with Two Simple Filtered Dynamic Subgrid Scale Models". In: *Numerical Heat Transfer, Part A: Applications* 37.5, pp. 447–463. ISSN: 1040-7782. DOI: 10.1080/104077800274154. URL: https://doi.org/10.1080/104077800274154.

Franklin, David G, Glenn E Lucas, and Arden L Bement (1983). *Creep of zirconium alloys in nuclear reactors*. 815. ASTM International.

Brantley, Patrick S and Edward W Larsen (2000). "The simplified P 3 approximation". In: *Nuclear science and engineering* 134.1, pp. 1–21.

Fratoni, Massimiliano (2008). *Development and applications of methodologies for the neutronic design of the Pebble Bed Advanced High Temperature Reactor (PB-AHTR)*. University of California, Berkeley. ISBN: 1109098359.

Duerigen, S et al. (2010). "A nodal SP3 approach for reactors with hexagonal fuel assemblies". In.

Bahabadi, Mohammad Hasan Jalili, Ali Pazirandeh, and Mitra Athari (2015). "New analytic function expansion nodal (AFEN) method for solving multigroup neutron simplified P3 (SP3) equations". In: *Annals of Nuclear Energy* 77, pp. 148–160.

Larsen, Edward, Jim Morel, and John McGhee (July 1996). "Asymptotic Derivation of the Multigroup P1 and Simplified PN Equations with Anisotropic Scattering". In: *Nuclear Science and Engineering* 123. DOI: 10.13182/NSE123-328.

Fiorina, Carlo, Mathieu Hursin, and Andreas Pautz (2017). "Extension of the GeN-Foam neutronic solver to SP3 analysis and application to the CROCUS experimental reactor". In: *Annals of Nuclear Energy* 101, pp. 419–428. ISSN: 0306-4549. URL: internal-pdf://191.77.145.168/Extension%20of%20GeN-Foam%20neutronic%20solver%20to%20SP3.pdf.

Lamarsh, John R and Anthony John Baratta (2001). *Introduction to nuclear engineering*. Vol. 3. Prentice Hall Upper Saddle River, NJ.

Tsang, DKL et al. (2005). "Graphite thermal expansion relationship for different temperature ranges". In: *Carbon* 43.14, pp. 2902–2906.

Caro, M et al. (2013). "Heavy liquid metal corrosion of structural materials in advanced nuclear systems". In: *Jom* 65, pp. 1057–1066.

Fu, Jian Xun, Xiang Dong Li, and Weng Sing Hwang (2011). "Study of the coefficient of thermal expansion for steel Q235". In: *Advanced Materials Research* 194, pp. 326–330.

Greene, Sherrell R et al. (2010). "Pre-conceptual design of a fluoride-salt-cooled small modular advanced high-temperature reactor (SmAHTR)". In: *Oak Ridge National Laboratory, ORNL/TM-2010/199*.

Lou, Lei, Dong Yao, et al. (2020). "A novel reactivity-equivalent physical transformation method for homogenization of double-heterogeneous systems". In: *Annals of Nuclear Energy* 142, p. 107396.

Abedi, Amin and Naser Vosoughi (2012). "Neutronic simulation of a pebble bed reactor considering its double heterogeneous nature". In: *Nuclear Engineering and Design* 253, pp. 277–284.

Aufiero, Manuele and Massimiliano Fratoni (2016). "Development of multi-physics tools for fluoride-cooled high-temperature reactors". In: PHYSOR. URL: internal-pdf://123.188.185.11/DEVELOPMENT%20OF%20MULTIPHYSICS%20TOOLS%20FOR%20FHR%20(Pro.pdf.

Satvat, Nader et al. (2021). "Neutronics, thermal-hydraulics, and multi-physics benchmark models for a generic pebble-bed fluoride-salt-cooled high temperature reactor (FHR)". In: *Nuclear Engineering and Design* 384, p. 111461.

Leppänen, Jaakko (2010). "Performance of Woodcock delta-tracking in lattice physics applications using the Serpent Monte Carlo reactor physics burnup calculation code". In: *Annals of Nuclear Energy* 37.5, pp. 715–722.

Bende, EE et al. (1999). "Analytical calculation of the average Dancoff factor for a fuel kernel in a pebble bed high-temperature reactor". In: *Nuclear Science and Engineering* 133.2, pp. 147–162.

Kloosterman, JL and Abderrafi M Ougouag (2005). *Computation of dancoff factors for fuel elements incorporating randomly packed triso particles*. Tech. rep. Idaho National Lab.(INL), Idaho Falls, ID (United States).

Rütten, HJ et al. (2005). "VSOP (99/05) computer code system". In.

Kim, Yonghee and Min Baek (2005). "Elimination of double-heterogeneity through a reactivity-equivalent physical transformation". In.

Lou, Lei, Xiaoming Chai, et al. (2020). "Research of ring RPT method on spherical and cylindrical Double-Heterogeneous systems". In: *Annals of Nuclear Energy* 147, p. 107741.

Noh, Jae Man et al. (2008). "Development of a computer code system for the analysis of prism and pebble type VHTR cores". In: *Annals of Nuclear Energy* 35.10, pp. 1919–1928.

Tu, Jiyuan et al. (2023). *Computational fluid dynamics: a practical approach*. Elsevier.

Moynihan, Cornelius T and Stanley Cantor (1968). "Viscosity and its temperature dependence in molten BeF2". In: *The Journal of Chemical Physics* 48.1, pp. 115–119. ISSN: 0021-9606.

Kasagi, Nobuhide and Mitsugu Nishimura (1997). "Direct numerical simulation of combined forced and natural turbulent convection in a vertical plane channel". In: *International Journal of Heat and Fluid Flow* 18.1, pp. 88–99. ISSN: 0142-727X.

Pope, Stephen B (2000). *Turbulent flows*. Cambridge university press.

Nguyen, Tri and Elia Merzari (2023). "Direct Numerical Simulation of High Prandtl Number Fluid Flow in the Downcomer of an Advanced Reactor". In: *Nuclear Science and Engineering*, pp. 1–26.

Gowen, RA and JW Smith (1967). "The effect of the Prandtl number on temperature profiles for heat transfer in turbulent pipe flow". In: *Chemical Engineering Science* 22.12, pp. 1701–1711.

Alam, Mohammad Faridul, David Thompson, and Dibbon Keith Walters (2017). "Critical assessment of hybrid RANS-LES modeling for attached and separated flows". In: *Critical Assessment of Hybrid RANS-LES Modeling for Attached and Separated Flows, BoD–Books on Demand*, p. 1.

Isaksson, Hanna (2019). *LES modelling of turbulent flow through an array of cylinders using OpenFOAM*.

Sagaut, Pierre (2005). *Large eddy simulation for incompressible flows: an introduction*. Springer Science & Business Media.

Meyers, Johan, Bernard J Geurts, and Pierre Sagaut (2007). "A computational error-assessment of central finite-volume discretizations in large-eddy simulation using a Smagorinsky model". In: *Journal of Computational Physics* 227.1, pp. 156–173.

Nicoud, Franck and Frédéric Ducros (1999). "Subgrid-scale stress modelling based on the square of the velocity gradient tensor". In: *Flow, turbulence and Combustion* 62.3, pp. 183–200.

Weickert, M et al. (2010). "Investigation of the LES WALE turbulence model within the lattice Boltzmann framework". In: *Computers & Mathematics with Applications* 59.7, pp. 2200–2214.

Li, Jing-Jing et al. (2018). "Large eddy simulation of unsteady flow in gas–liquid separator applied in thorium molten salt reactor". In: *Nuclear Science and Techniques* 29.5, p. 62.

Merzari, Elia et al. (2020). "Wall resolved large eddy simulation of reactor core flows with the spectral element method". In: *Nuclear Engineering and Design* 364, p. 110657.

Spalart, Philippe R (1997). "Comments on the Feasibility of LES for Wings and on the Hybrid RANS/LES Approach". In: *Proceedings of the First AFOSR International Conference on DNS/LES, 1997*, pp. 137–147.

Spalart, Philippe R et al. (2006). "A new version of detached-eddy simulation, resistant to ambiguous grid densities". In: *Theoretical and computational fluid dynamics* 20, pp. 181–195.

Shur, Mikhail L et al. (2008). "A hybrid RANS-LES approach with delayed-DES and wall-modelled LES capabilities". In: *International journal of heat and fluid flow* 29.6, pp. 1638–1649.

Shams, A et al. (2015). "Improved delayed detached eddy simulation of a randomly stacked nuclear pebble bed". In: *Computers & Fluids* 122, pp. 12–25.

Dave, AJ, K Sun, and L Hu (2020). "Numerical assessment of packed-bed heat transfer correlations for molten salt". In: *Annals of Nuclear Energy* 136, p. 107002.

Wu, CY et al. (2010). "Investigating the advantages and disadvantages of realistic approach and porous approach for closely packed pebbles in CFD simulation". In: *Nuclear Engineering and design* 240.5, pp. 1151–1159.

Novak, AJ et al. (2018). "Pronghorn: Porous media thermal-hydraulics for reactor applications". In.

Clifford, Ivor D (2013). *A hybrid coarse and fine mesh solution method for prismatic high temperature gas-cooled reactor thermal-fluid analysis*. The Pennsylvania State University.

Sobieski, Wojciech and Anna Trykozko (2014). "Darcy's and Forchheimer's laws in practice. Part 1. The experiment". In: *Tech. Sci. Warm. Maz. Olsztyn.* ISSN: 1505-4675.

Fiorina, Carlo, Nordine Kerkar, et al. (2016). "Development and verification of the neutron diffusion solver for the GeN-Foam multi-physics platform". In: *Annals of Nuclear Energy* 96, pp. 212–222. ISSN: 0306-4549. DOI: https://doi.org/10.1016/j.anucene.2016.05.023. URL: http://www.sciencedirect.com/science/article/pii/S0306454916303309.

Vafai, Kambiz (2015). *Handbook of porous media*. Crc Press. ISBN: 1439885575.

Hassan, Yassin A and Changwoo Kang (2012). "Pressure drop in a pebble bed reactor under high Reynolds number". In: *Nuclear technology* 180.2, pp. 159–173.

Ozahi, Emrah, Mehmet Yasar Gundogdu, and Melda Ö Carpinlioglu (2008). "A modification on Ergun's correlation for use in cylindrical packed beds with non-spherical particles". In: *Advanced Powder Technology* 19.4, pp. 369–381.

Liu, Maolong et al. (2019). "Development of a two-regime heat conduction model for TRISO-based nuclear fuels". In: *Journal of Nuclear Materials* 519, pp. 255–264.

Oh, Chang (2006). *Development of safety analysis codes and experimental validation for a very high temperature gas-cooled reactor Final report*. Tech. rep. Idaho National Lab.(INL), Idaho Falls, ID (United States).

Derdeyn, Will et al. (2018). "FLiBe Radiative Heat Transfer". In: *Transactions of the American Nuclear Society* 118. ISSN: 0003-018X.

Modest, Michael F (2013). *Radiative heat transfer*. Academic press. ISBN: 0123869900.

Amber, Ityona and TS O'Donovan (2017). "Heat transfer in a molten salt filled enclosure absorbing concentrated solar radiation". In: *International Journal of Heat and Mass Transfer* 113, pp. 444–455.

Abou Dbai, Mohamed, Raluca O Scarlat, and Mario F Trujillo (2020). "Radiative heat transfer in FLiBe molten salt participating medium in a vertical heated tube under forced and mixed convection laminar flows". In: *Nuclear Engineering and Design* 368, p. 110775.

Leppänen, Jaakko et al. (2014). "The Serpent Monte Carlo code: Status, development and applications in 2013". In: *SNA+ MC 2013-Joint International Conference on Supercomputing in Nuclear Applications+ Monte Carlo*. EDP Sciences, p. 6021. ISBN: 2759812693.

Novak, April et al. (2018). "Preliminary coupling of OpenMC and Nek5000 within the MOOSE framework". In: *Proc. PHYSOR*, pp. 22–26.

Aufiero, Manuele, Carlo Fiorina, et al. (2015). "Serpent–OpenFOAM coupling in transient mode: simulation of a Godiva prompt critical burst". In: *Proceedings of M&C+ SNA+ MC*, pp. 19–23.

Sorrell, Nina C and Ayman I Hawari (2019). "TREAT M2 experiment modeling for transient benchmark analysis". In: *Annals of Nuclear Energy* 128, pp. 398–405.

Krasnopolsky, Boris and Alexey Medvedev (2016). "Acceleration of large scale OpenFOAM simulations on distributed systems with multicore CPUs and GPUs". In: *Parallel Computing: On the Road to Exascale*. IOS Press, pp. 93–102.

Comsol (May 2023). *What hardware do you recommend for COMSOL Multiphysics?* Website. Shows that GPUs General-purpose computing on graphics processing units is not currently supported on Comsol. URL: `https://www.comsol.de/support/knowledgebase/866`.

Macfarlane, Robert et al. (2017). *The NJOY Nuclear Data Processing System, Version 2016*. Tech. rep.

Huysmans, Marijke and Alain Dassargues (2005). "Review of the use of Péclet numbers to determine the relative importance of advection and diffusion in low permeability environments". In: *Hydrogeology Journal* 13, pp. 895–904.

Riegel, Juergen, Werner Mayer, and Yorik van Havre (2016). *FreeCAD*.

Lee, Sang Yong, Chan Eok Park, and Shin Whan Kim (2014). "SALOME PLATFORM and TetGen for Polyhedral Mesh Generation". In.

Ahlborn, Boye, Mae L Seto, and Bernd R Noack (2002). "On drag, Strouhal number and vortex-street structure". In: *Fluid dynamics research* 30.6, p. 379.

Hao, Chen, Rongrui Yang, and Youying Cheng (2022). "Uncertainty analysis of keff due to the random packing of pebbles in the pebble bed HTR". In: *Nuclear Engineering and Design* 398, p. 111992.

Abedi, Amin and Naser Vosoughi (2011). "An exact MCNP modeling of pebble bed reactors". In: URL: `internal-pdf://0960594176/An-Exact-MCNP-Modeling-of-Pebble-Bed-Reactors.pdf`.

Vajta, Miklos et al. (2000). "Some remarks on Padé-approximations". In: *Proceedings of the 3rd TEMPUS-INTCOM Symposium*. Vol. 242, pp. 1–6.

Cisneros, Anselmo T et al. (2014). "Technical Description of the "Mark 1" Pebble-Bed Fluoride-Salt-Cooled High-Temperature Reactor (PB-FHR) Power Plant". In: *Tech. Rep.*

Wong, Samuel SM (1998). *Introductory nuclear physics*. John Wiley & Sons.

Li, XX et al. (2015). "Analysis of thorium and uranium based nuclear fuel options in Fluoride salt-cooled High-temperature Reactor". In: *Progress in Nuclear Energy* 78, pp. 285–290.

Urbatsch, Todd J et al. (1995). "Estimation and Interpretation of keff Confidence Intervals in MCNP". In: *Nuclear technology* 111.2, pp. 169–182.

Smith, Kord S (1986). "Assembly homogenization techniques for light water reactor analysis". In: *Progress in Nuclear Energy* 17.3, pp. 303–335.

Jiang, Wen et al. (2021). "TRISO particle fuel performance and failure analysis with BISON". In: *Journal of Nuclear Materials* 548, p. 152795.

Uddin, Md Jashim, Mir Md Moheuddin, and Md Kowsher (2019). "A New Study of Trapezoidal, Simpson's 1/3 and Simpson's 3/8 Rules of Numerical Integral Problems". In: *Applied Mathematics and Sciences: An International Journal (MathSJ)* 6.4, pp. 1–14.

Kavazauri, R et al. (2016). "Thermal properties of nonstoichiometry uranium dioxide". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 130. 1. IOP Publishing, p. 012025.

Pilehvar, AF et al. (2013). "Evaluation of compressible flow in spherical fueled reactors using the porous media model". In: *Annals of Nuclear Energy* 57, pp. 185–194.

Vidrio, Ricardo et al. (2022). "Density and Thermal Expansivity of Molten 2LiF-BeF2 (FLiBe): Measurements and Uncertainty Quantification". In: *Journal of Chemical & Engineering Data* 67.12, pp. 3517–3531.

Ma, Yuanqing et al. (2023). "Investigation of heavy gas dispersion characteristics in a static environment: Spatial distribution and volume flux prediction". In: *Building and Environment*, p. 110501.

Jasak, Hrvoje (2015). "Finite Volume Discretisation in OpenFOAM-Best Practice Guidelines". In: *Predavanja, CFD with OpenSource Software Course, Chalmers University*.

Kollie, TG et al. (1975). "Temperature measurement errors with type K (Chromel vs Alumel) thermocouples due to short-ranged ordering in Chromel". In: *Review of Scientific Instruments* 46.11, pp. 1447–1461.

Seborg, Dale E et al. (2016). *Process dynamics and control*. John Wiley & Sons.

Alpha, Wolfram (2023). *Wolfram—Alpha*. Accessed on October 12, 2023. URL: http://www.wolframalpha.com/.

Chen, Lei, Junhong Li, and Ruifeng Ding (2011). "Identification for the second-order systems based on the step response". In: *Mathematical and computer modelling* 53.5-6, pp. 1074–1083.

Bequette, B. Wayne (Mar. 1999). "The IMC-Based PID Procedure". In: IMC Based Procedure that removes that bad parts, by Rensselaer Polytechnic Institute. URL: https://rpi.edu/dept/chem-eng/WWW/faculty/bequette/courses/cpc/IMC_PID.pdf.

Cansalar, Civan Artun, Ertan Maviş, and Coşku Kasnakoğlu (2015). "Simulation time analysis of MATLAB/Simulink and LabVIEW for control applications". In: *2015 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, pp. 470–473.

Avramova, Maria N and Kostadin N Ivanov (2010). "Verification, validation and uncertainty quantification in multi-physics modeling for nuclear reactor design and safety analysis". In: *Progress in Nuclear Energy* 52.7, pp. 601–614.

Xie, Yichen and Alex Aiken (2005). "Context-and path-sensitive memory leak detection". In: *Proceedings of the 10th European software engineering conference held jointly with 13th ACM SIGSOFT international symposium on Foundations of software engineering*, pp. 115–125.

Mikac, Matija, Robert Logožar, and Miroslav Horvatić (2022). "Performance Comparison of Open Source and Commercial Computing Tools in Educational and Other Use—Scilab vs. MATLAB". In: *Tehnički glasnik* 16.4, pp. 509–518.

Butcher, John Charles and PB Johnston (1993). "Estimating local truncation errors for Runge-Kutta methods". In: *Journal of Computational and Applied Mathematics* 45.1-2, pp. 203–212.

Tsitouras, Ch and SN Papakostas (1999). "Cheap error estimation for Runge–Kutta methods". In: *SIAM Journal on Scientific Computing* 20.6, pp. 2067–2088.

Irvine, Tom (2011). "Table of Laplace Transforms". In: *Revision I, Vibrationdata*.

Campbell, Stephen L et al. (2010). *Modeling and Simulation in SCILAB*. Springer.

rupakrokade (Aug. 2018). *What is algebraic Loop? When it Occurs in Xcos? How to Overcome it?* fosse-forums. URL: https://forums.fossee.in/question/650/.

Isaksson, AJ and SF Graebe (2002). "Derivative filter is an integral part of PID design". In: *IEE Proceedings-Control Theory and Applications* 149.1, pp. 41–45.

Ernerfeldt, Emil (June 2023a). *egui v 0.23.0*. Rust Crate io. URL: `https://crates.io/crates/egui`.

— (June 2023b). *eframe template*. Github. URL: `https://github.com/emilk/eframe_template`.

Litvin, Andrei (June 2023). *rs-value-plotter*. Github. URL: `https://github.com/andy31415/rs-value-plotter`.

Gallant, Andrew (June 2023). *rust-csv v1.61.0*. Github. URL: `https://github.com/BurntSushi/rust-csv`.

Fruehauf, Paul S, I-Lung Chien, and Mark D Lauritsen (1994). "Simplified IMC-PID tuning rules". In: *ISA transactions* 33.1, pp. 43–59.

Xylem (June 2023). *Standard Xchange Fanex air-cooled heat exchanger — Xylem Norge*. nb-no. URL: `https://www.xylem.com/nb-noproducts--services/fanex--plate-fin-core-with-fan-air-cooled-heat-exchanger-airoil/` (visited on 06/27/2023).

Appendix A: Digital Twin Construction

Coiled Tube Air Heater (CTAH) Data for Modelling and Validation

The air-cooled heat exchanger in the primary loop modelled in Transform was meant to replicate the function of the CTAH in the FHR [De Wet and Per F Peterson, 2020]. In CIET RELAP models, this air-cooled heat exchanger was also called the CTAH [Nicolas Zweibaum, 2015]. The isothermal digital twin in previous work also follows this naming convention [Ong, 2023]. These air-cooled heat exchangers were Xylem Standard Exchange Model 15L F700 FanEx heat exchangers [Jeffrey E Bickel, Nicholas Zweibaum, and Per F Peterson, 2014]. We shall stick to this naming convention to avoid confusion with previous work.

Xylem Standard Exchange Model 15L F700 FanEx heat exchangers contain a large fan with aluminium fins and 0.5 inch outer diameter copper tubing to aid heat exchange [Xylem, 2023]. Its motor is a 115 Volt, single phase, 60 Hz motor [Xylem, 2023]. Modelling this geometry with high fidelity would be challenging. Therefore, one could use simplified calibrated models to model the CTAH.

The CTAH in Transform was modelled as an adiabatic inlet volume, followed by a heater consisting of 12 parallel copper tubes, and then an adiabatic outlet volume. The inlet and outlet volumes were 0.001 m^3 each [De Wet and Per F Peterson, 2020]. Heat transfer in the inner wall was assumed to be ideal as the F700 Model 15L uses hollow metal turbulator spheres to enhance heat exchange [Xylem, 2023; De Wet and Per F Peterson, 2020]. One should note however that this idealisation works for a flowrate of 0.18 kg/s.

The heat transfer coefficient at 0.18 kg/s oil flowrate in terms of fan frequency is [De Wet and Per F Peterson, 2020]:

$$h = -0.2021f^2 + 18.15f - 43.07 \tag{1}$$

Of course, we can also model the CTAH to some desired temperature boundary condition. For the SAM model, the CTAH was set to remove heat such that its outlet temperature was 80°C [Zou, R. Hu, and Charpentier, 2019]. This is usually done experimentally so that

the loop is able to reach steady state in preparation for frequency response testing [De Wet and Per F Peterson, 2020]. If one desires to use the frequency response data from CIET for the CTAH and the whole loop, one must note that in the PRBS frequency response tests for forced convection, the CTAH was kept to a contanst fan frequency [De Wet and Per F Peterson, 2020]. The correlation for that was [De Wet and Per F Peterson, 2020]:

$$h = 0.0352\dot{Q} - 8.7472 \tag{2}$$

I was not able to find units easily for both these correlations, but there is still some validation data for the CTAH with units. For example, De Wet gives the correlation of the required fan frequency in Hz to bring Therminol-VP1 temperatures from down to $80°C$. This was derived from experimental data at 0.18 kg/s where CTAH fan speed was plotted against heater power from 3000 W to 10000 W at $80°C$ CTAH outlet temperature. The simplified linear correlation is [De Wet and Per F Peterson, 2020]:

$$f(Hz) = 0.0035823\ \dot{Q}_{heater}(W) - 3.410 \pm 2.0Hz \tag{3}$$

The other dataset we can consider is the steady state temperature profile around CTAH at the primary loop flowrate of 0.18 kg/s and CTAH air inlet temperature of $22°C$. The fan frequency for this data set is set to ensure that the CTAH outlet temperature is $80°C$. This data is presented in Table 1:

CTAH Fan Frequency (Hz)	CTAH Therminol Inlet Temperature (BT-43) $°C$	CTAH Therminol Outlet Temperature (BT-41) $°C$	CTAH Air Inlet Temperature (AT-02) $°C$	CTAH Air Outlet Temperature (AT-01) $°C$
8.3	86.75	80	22	36.9
11.2	90.1	80	22	63.9
16.5	96.25	80.4	22	64.5
24.2	102	80	22	63.2
33.8	107.5	80	22	59.4

Table 1: CTAH Steady State Data at Various frequencies [De Wet and Per F Peterson, 2020]

This dataset provides us both temperature difference and enthalpy changes of the air and Therminol-VP1 flowing through CTAH at different fan frequencies. With appropriate thermophysical data and a thermal resistance model, we should be able to obtain the heat transfer coefficient h and determine the original units. The heater power, steady state inlet and outlet temperatures corresponding to the CTAH inlet and outlet temperatures is presented in Table 2:

Heater Power (W)	Heater Inlet Temperature (BT-11) $°C$	Heater Outlet Temperature (BT-12) $°C$	CTAH Inlet Temperature (BT-43) $°C$	CTAH Outlet Temperature (BT-41) $°C$
3000	78.75	86.93	86.75	80
4000	79	90.25	90.1	80
6000	79.4	96.5	96.25	80.4
8000	79.12	102.2	102	80
10000	78.9	107.75	107.5	80

Table 2: Forced Circulation Steady State Temperature Data [De Wet and Per F Peterson, 2020]

To model the thermal resistance within the CTAH, we also need the thermophysical properties of copper which make up the tubes. For these thermophysical properties, Copper for the CTAH tubes was taken to have a density of 8940 kg/m^3 in the SAM model [Zou, R. Hu, and Charpentier, 2019]. The c_p and k for copper are presented in the following Table 3:

Temperature (K)	k ($W\ m^{-1}\ K^{-1}$)	c_p ($J\ kg^{-1}\ K^{-1}$)
250	406	373.6018
300	401	384.7875
350	396	392.6174
400	393	398.2103
500	386	407.1588
1000	352	417.226

Table 3: Thermophysical Properties of Copper [Zou, R. Hu, and Charpentier, 2019; Nicolas Zweibaum, 2015]

This covers most of the data necessary to model CTAH. While it is not modelled in this current work, this modelling and validation data will become important in future work when heat exchangers and coolers will need to be properly modelled.

Appendix B: Simulated Neutronics Feedback Model Construction

GeN-Foam Stabilisation Bash Script

```
function thermalHydraulicsStabilisation (){
    copyCaseFiles rootCase thermalHydraulicsSteadyState

    # to stabilise, we first turn on energy and fluid mechanics
    # without the neutronics equation, run for 5000s
    controlDict_solveThermalHydraulicsOnly thermalHydraulicsSteadyState
    endTime="5000"
    controlDict_setEndTime thermalHydraulicsSteadyState $endTime
    eigenValueOff thermalHydraulicsSteadyState
    runApp_controlDict thermalHydraulicsSteadyState

    # thereafter we should have a proper temperature field
    # and now we run the eigenValue mode so that the keff can
    # be normalised for the transient simulation
    eigenValueOn thermalHydraulicsSteadyState
    controlDict_solveEnergy thermalHydraulicsSteadyState
    extendEndTime thermalHydraulicsSteadyState 10
    runApp_controlDict thermalHydraulicsSteadyState

    # now we can run it with neutronics on to get it to the
    # proper fields at constant temperature
    # only solve fluid mechanics and neutronics
    controlDict_solveNeutronicsFluidsOnly thermalHydraulicsSteadyState
    extendEndTime thermalHydraulicsSteadyState 10
    runApp_controlDict thermalHydraulicsSteadyState

    # now that we have a proper neutronics field,
    # we can turn all equations on, and solve with transient solver
```

```
    eigenValueOff thermalHydraulicsSteadyState
    controlDict_solveEnergy thermalHydraulicsSteadyState
    extendEndTime thermalHydraulicsSteadyState 8980
    runApp_controlDict thermalHydraulicsSteadyState

    # at this stage, one should be able to get a non zero
    # power reading
    # and we are ready to do frequency response!

}
```

GeN-Foam exmaple of fvSchemes for Tetrahedral Meshes

```
ddtSchemes
{
    default         Euler;
}

gradSchemes
{
    default         leastSquares;
}

divSchemes
{
    default                                 none;

    div(phi,alpha)                          Gauss upwind;
    div(phir,alpha)                         Gauss upwind;

    div(phi,alpha.liquid)                   Gauss upwind;
    div(phir,alpha.vapour,alpha.liquid)     Gauss upwind;
    div(phir,alpha.structure,alpha.liquid)  Gauss upwind;
    div(phi,alpha.vapour)                   Gauss upwind;
    div(phir,alpha.liquid,alpha.vapour)     Gauss upwind;
    div(phir,alpha.structure,alpha.vapour)  Gauss upwind;

    "div\(phi.*,U.*\)"                      Gauss upwind;
    "div(alphaRhoPhi,U)"                    Gauss upwind;
```

```
    "div(alphaRhoPhiNu,U)"                  Gauss linear;
    "div(alphaRhoPhi,K)"                    Gauss upwind;

    "div\(alphaRhoPhi.*,k.*\)"              Gauss upwind;
    "div\(alphaRhoPhi.*,epsilon.*\)"        Gauss upwind;

    "div\(alphaRhoPhi.*,(h|e).*\)"          Gauss upwind;
}

laplacianSchemes
{
    default         Gauss linear limited corrected 0.5;
}

interpolationSchemes
{
    default         linear;
}

snGradSchemes
{
    //default             corrected;
    default             limited corrected 0.5;
}
```

PRBS Sequence with Timestamps

Table 4: PRBS Sequence 128 Bit at 30s per Bit (part i)

Bit Number	PRBS Sequence	Simulation Time (s)
1	0	0
2	0	30
3	0	60
4	0	90
5	0	120
6	1	150
7	0	180
8	0	210
9	0	240
10	0	270
11	0	300
12	1	330
13	1	360
14	0	390
15	0	420
16	0	450
17	0	480
18	1	510
19	0	540
20	1	570
21	0	600
22	0	630
23	0	660
24	1	690
25	1	720
26	1	750

Table 5: PRBS Sequence 128 Bit at 30s per Bit (part ii)

Bit Number	PRBS Sequence	Simulation Time (s)
27	1	780
28	0	810
29	0	840
30	1	870
31	0	900
32	0	930
33	0	960
34	1	990
35	0	1020
36	1	1050
37	1	1080
38	0	1110
39	0	1140
40	1	1170
41	1	1200
42	1	1230
43	0	1260
44	1	1290
45	0	1320
46	1	1350
47	0	1380
48	0	1410
49	1	1440
50	1	1470
51	1	1500
52	1	1530
53	1	1560

Table 6: PRBS Sequence 128 Bit at 30s per Bit (part iii)

Bit Number	PRBS Sequence	Simulation Time (s)
54	0	1590
55	1	1620
56	0	1650
57	0	1680
58	0	1710
59	0	1740
60	1	1770
61	1	1800
62	1	1830
63	0	1860
64	0	1890
65	0	1920
66	1	1950
67	0	1980
68	0	2010
69	1	2040
70	0	2070
71	0	2100
72	1	2130
73	1	2160
74	0	2190
75	1	2220
76	1	2250
77	0	2280
78	1	2310
79	0	2340
80	1	2370

Table 7: PRBS Sequence 128 Bit at 30s per Bit (part iv)

Bit Number	PRBS Sequence	Simulation Time (s)
81	1	2400
82	0	2430
83	1	2460
84	1	2490
85	1	2520
86	1	2550
87	0	2580
88	1	2610
89	1	2640
90	0	2670
91	0	2700
92	0	2730
93	1	2760
94	1	2790
95	0	2820
96	1	2850
97	0	2880
98	0	2910
99	1	2940
100	0	2970
101	1	3000
102	1	3030
103	1	3060
104	0	3090
105	1	3120

Table 8: PRBS Sequence 128 Bit at 30s per Bit (part v)

Bit Number	PRBS Sequence	Simulation Time (s)
106	1	3150
107	1	3180
108	0	3210
109	0	3240
110	1	3270
111	1	3300
112	0	3330
113	0	3360
114	1	3390
115	0	3420
116	1	3450
117	0	3480
118	1	3510
119	0	3540
120	1	3570
121	1	3600
122	1	3630
123	1	3660
124	1	3690
125	1	3720
126	1	3750
127	0	3780
128	0	3810

The meaning of Phi (ϕ) in the context of GeN-Foam

Mass Flux or Volume Flux?

Under "reconstructU_1p.H", we find that phi may have the same units as velocity U:

```
U = fvc::reconstruct(fluid_.phi());
```

Furthermore, in "updateAlphaPhiAndInitControls_1p.H", we see two distinct fields being created, alphaPhi and alphaRhoPhi:

```
fluid_.alphaPhi() = alphaf*fluid_.phi();
fluid_.alphaRhoPhi() =
    fvc::interpolate(rho)*fluid_.alphaPhi();
```

This gives us a clue that density ρ is multiplied with ϕ to give us $\rho\phi$. So ϕ in the case for GeN-Foam may not contain the density. Now of course, it is in theory possible to weight the field by RhoPhi in GeN-Foam.

Does Flux mean Volumetric Flowrate or Volumetric Flowrate per Unit Area?

The reader should also note that flux in GeN-Foam context is not the same unit as velocity, but rather volumetric flowrate and this is sometimes the case in literature [Ma et al., 2023]. It is used to represent volumetric flowrate through faces on the mesh rather than volumetric flowrate per unit area.
We see previously that fluid_.phi() is called, within "fluid.C":

```
const surfaceScalarField& phi() const
{
    return phiPtr_();
}

surfaceScalarField& phi()
{
    return phiPtr_();
}
```

phiPtr is found in "fluid.C":

```
//- Superficial flux of the phase
autoPtr<surfaceScalarField> phiPtr_;
```

```
//- Volumetric flux of the phase
surfaceScalarField alphaPhi_;

//- Mass flux of the phase
surfaceScalarField alphaRhoPhi_;
```

We confirm that phi represents volumetric flux. Another clue is the unit convention in alphaPhi_:

```
alphaPhi_
(
    IOobject
    (
        IOobject::groupName("alphaPhi", this->name()),
        mesh.time().timeName(),
        mesh,
        IOobject::READ_IF_PRESENT,
        (
            (
                mesh.time().controlDict().lookupOrDefault<bool>
                (
                    "writeRestartFields",
                    true
                )
            ) ?
            IOobject::AUTO_WRITE :
            IOobject::NO_WRITE
        )
    ),
    mesh,
    dimensionedScalar("", dimVol/dimTime, 0)
),
```

The quantity alpha, which is the phase fraction, is dimensionless. Therefore phi has dimensions of volumetric flowrate. Now, the question is whether $\phi = \vec{u} \cdot \vec{A}$ or $\phi = \vec{u} \cdot \frac{\vec{A}}{|\vec{A}|}$. The latter can be interpreted as a projection, but projections generally are in the same unit as $\vec{u}$ rather than volumetric flowrate. It is most likely the case that phi is in units of volumetric flowrate.
The most concrete proof is where phi is created in "createPhi.H" under the openfoam source files in "src/finiteVolume":

```
Info<< "Reading/calculating face flux field phi\n" << endl;
```

```
surfaceScalarField phi
(
    IOobject
    (
        "phi",
        runTime.timeName(),
        mesh,
        IOobject::READ_IF_PRESENT,
        IOobject::AUTO_WRITE
    ),
    fvc::flux(U)
);
```

The function used here is “fvc::flux(U)”. These are in the source files “fvcFlux.H” and “fvcFlux.C” under “src/finiteVolume/fvc”:

```
Foam::tmp<Foam::surfaceScalarField> Foam::fvc::flux
(
    const volVectorField& vvf
)
{
    return scheme<vector>
    (
        vvf.mesh(),
        "flux(" + vvf.name() + ')'
    )().dotInterpolate(vvf.mesh().Sf(), vvf);
}
```

We see a dotInterpolate function which I would assume is the dot product. “vvf” is just shorthand for the volVectorField, in this case “U” or the velocity vector field over the whole volume. The volVectorField entry has a “mesh()” function which I suppose returns the mesh, and “Sf()” which I suppose returns surface area. What then is this mesh? Based on “fvcFlux.H”, the most likely candidate is the “volFieldsFwd” source file which is included in “fvcFlux.H”.
Reading “fields/volFields/volFieldsFwd.H”, we see that it includes “fieldTypes.H”. Seems to be a dead end. The other lead is “surfaceInterpolate.H”. This gives us the dotInterpolate function:

```
 Foam::fvc::dotInterpolate
 (
     const surfaceVectorField& Sf,
```

```
    const GeometricField<Type, fvPatchField, volMesh>& vf
)
{
    if (surfaceInterpolation::debug)
    {
        InfoInFunction
            << "interpolating GeometricField<Type, fvPatchField, volMesh> "
            << vf.name() << " using run-time selected scheme"
            << endl;
    }

    return scheme<Type>
    (
        vf.mesh(),
        "dotInterpolate(" + Sf.name() + ',' + vf.name() + ')'
    )().dotInterpolate(Sf, vf);
}
```

I suppose this means that phi is really $\vec{u} \cdot \vec{A}$. This means that I need to be taking the weighted average of velocity $\vec{u}$ using ϕ or volumetric flowrate at the surface as the weight.