AN ABSTRACT OF THE THESIS OF

Cole W. Leingang for the degree of Master of Science in Nuclear Engineering presented on September 27, 2021.

Title: External Coupling of Computational Fluid Dynamics Modeling of the Density Evolution Inside of the Helium-3 Enhanced Negative Reactivity Insertion (HENRI) System with the Monte-Carlo Neutronics code Serpent 2.

Abstract Approved: _____

Guillaume, Mignot

The testing of nuclear fuel under reactivity-initiated accident (RIA) conditions is paramount for the better understanding of the fuel's behavior during these transient accident events. The Transient Reactor Test (TREAT) facility is a nuclear reactor that will be capable of recreating the thermal-hydraulic and neutronic boundary conditions representative of RIA events for light water reactors (LWRs). However, one of the engineering challenges to perform such fuel tests is to increase the energy deposition on the fuel sample by reducing the current TREAT's pulse width of 89 milliseconds down to 40 milliseconds. Idaho National Laboratory (INL) proposed to clip the pulse by inserting helium-3, a strong neutron absorber, into an annular control rod using a gas injection system known as the Helium-3 Enhanced Negative Reactivity (HENRI) facility.

This study outlines the process of the coupling of the computational fluid dynamics (CFD) model of HENRI built in STAR CCM+, with the reactor physics code, Serpent 2. The coupling of these two codes is pursued in order to provide a more representative analysis of the TREAT reactor reactivity transient when the HENRI cartridge system is deployed. Specifically, to outline the effectiveness of the HENRI cartridges capability of inserting 5% negative reactivity in 5 msec in order to "clip" TREAT. Using the coupling of STAR-CCM+ and Serpent 2 it can be shown that the activation of 4 HENRI cartridges with an initial pressure of 3.45 MPa leads to an insertion of 13% negative reactivity within 4 milliseconds.

External Coupling of Computational Fluid Dynamics Modeling of the Density Evolution Inside of the Helium-3 Enhanced Negative Reactivity Insertion (HENRI) System with the Monte-Carlo Neutronics code Serpent 2

By

Cole W. Leingang

A THESIS

Submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Master of Science

Presented: September 27, 2021

Commencement: INSERT DATE

Master of Science thesis of Cole W. Leingang presented on September 27, 2021.

APPROVED:

Major Professor, representing Nuclear Engineering

Head of the Department of Nuclear Science and Engineering

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Cole W. Leingang, Author

ACKNOWLEDGEMENTS

I would like to express my deepest regards and appreciation to my faculty advisor, Dr. Guillaume Mignot. Dr. Mignot guided me through 2 years of effort which culminated in the completion of my thesis. It was a true honor to have worked under Dr. Mignot and I have grown tremendously as a researcher under his tutelage.

I would like to also thank Dr. Wade R. Marcum, whose research group I was a part of for these past 2 years. Dr. Marcum brought a tremendous amount of experience and guidance that allowed me to push through the hurdles associated with completing my thesis. Without Dr. Marcum I would not be where I am today and for that I am eternally grateful.

I would like to thank Dr. Trevor K. Howard. Dr. Howard provided key insights and experience on the basis of coupling codes and the verification process. His inputs were greatly appreciated and helped both the development of my work and my own understanding of code coupling and verification.

I would also like to thank my committee members, the Radiation Center staff, and the other members of my research group who all played a role in creating the wonderful experience I was a part of at Oregon State University.

Finally, I would like to deeply thank my parents, Scott Leingang and Teresa Leingang. They have always provided constant support through my life and helped me achieve my farthest goals.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

Table Page

External Coupling of Computational Fluid Dynamics Modeling of the Density Evolution Inside of the Helium-3 Enhanced Negative Reactivity Insertion (HENRI) System with the Monte-Carlo Neutronics code Serpent 2.

1. INTRODUCTION

Recently, in November 2017, the TREAT facility at Idaho National Laboratory (INL) reached criticality once again after having been placed on standby since 1994. This was done in accordance with a Department of Energy request to restore operations of the TREAT facility. This request primarily focused on the desire to support the Accident Tolerant Fuel (ATF) program [1] which gained prevalence after the Japanese Fukushima Daichi plant accident in 2011. The focus of this program is to further expand the knowledge of the response of new and currently deployed nuclear fuel concepts under proposed nuclear accident conditions.

One focus of the ATF program is the desire to better understand Reactivity Initiated Accidents (RIAs). RIAs are scenarios in which a nuclear reactor can experience a sudden and rapid increase of fission rate leading to an increase in reactor power. This can ultimately lead to the failure of the fuel rods and there cladding which often results in the expulsion of fuel particles directly into the coolant of the reactor. This expulsion can lead to further damages to the reactor depending on its severity. A general overview of RIAs and their impact on nuclear fuel can be found here [2]. It was found in this report that the current understanding of RIAs is more limited than other types of nuclear accidents. This stems from the difficulty of RIA testing and the lack of extensive computational models. Due to the destructive nature of RIAs and the nuclear industries commitment to a defense in depth philosophy, the study and further understanding of RIAs and how fuel behaves under such situations is essential in ensuring the safe operation of a nuclear power plant.

The TREAT facility has the capability of performing large power bursts, however, due to the overall design of the reactor these power bursts are longer than what is desired to study expected

RIA events within light water reactors, which is a main focus of the ATF program. A long-duration pulse has the potential to exceed the maximum allowable energy deposition indicative of a RIA event within a light water reactor, which is 2500 MJ. Coincidentally a shorter-power burst would not provide the minimum desired energy deposition. This has led to the idea of “clipping”, or shortening, the full-power pulse in order to reach the desired energy deposition.

The TREAT facility has reached a minimum Full-Width at Half-Maximum (FWHM) of 89 milliseconds [5] in its current configuration using only the operation of the control rods. In order to reach the desired energy deposition, this FWHM needs to be reduced to around 40 milliseconds.

To achieve this goal, it has been proposed to inject a neutron poisoning gas into the TREAT reactor while in operation. The reason being that using control rods or additional poison rods to reach the desired FWHM can lead to large mechanical stresses on the system and possible mechanical failures. The use of a neutron poisoning gas would eliminate the concern of mechanical stresses on the system and not require any major mechanical modifications or additions to the TREAT facility, bar the system used to inject the gas.

When first conceiving the concept in 1998, Crawford et al. [3] proposed to use helium-3 gas as a poison gas. The proposed system was to make use of pre-existing penetrations of the core and would be capable of inserting a large amount of negative reactivity into the TREAT core rapidly. The desired amount of negative reactivity was stated to be 5% negative reactivity in 5 msec. The system proposed was never built but the concept was born. The use of injecting helium-3 gas within the TREAT system was revisited in 2019 [6].

Helium-3 is a rare and expensive gas, with the estimated cost of ~1000 $/L [6]. Due to this, other viable neutron poison gases were considered. The main alternative was that of boron-10 trifluoride ($^{10}BF_{3)}$, which when analyzed performed similar to helium-3. However, $^{10}BF_3$ is toxic if inhaled and can form acids when exposed to airborne moisture. If this gas was used as an alternative, it would require additional containment structures and run a much higher risk of accidents. There also exists a similar system using helium-3 which has been deployed at the French CABRI reactor [4] for years. This allows for working knowledge of helium-3 and its characteristics to be leveraged. As such the use of helium-3 within the proposed system was chosen.

In addition to being able to leverage previous knowledge, helium-3 has the intrinsic properties of not only being a stable isotope, meaning it can be stored without worry of depletion, but more importantly it has a large neutron absorption cross section of ~5000 barns. This high neutron absorption cross section makes it an ideal candidate to "clip" the power of the reactor as it has the potential to absorb a large number of neutrons that would otherwise be feeding the fission-chain reaction used to produce the power-pulse within TREAT. However, because helium-3 has such a high neutron absorption cross section, it cannot be placed inside the TREAT reactor core in any large quantity at start-up as it would act as a fission poison and not allow TREAT to reach the full-power pulse needed to simulate a RIA. As such, the proposed system would inject the gas into the reactor while it is in operation. This system would consist of a pressurized out of core driver tank connected to an evacuated tube placed inside the reactor. For comparison, the system in the CABRI facility has the helium-3 initially in the reactor and is then rapidly removed [4].

The decision to use a fast-pressurization system with helium-3 led to the design and construction of the out-of-pile Helium-3 Enhanced Negative Reactivity Insertion (HENRI) facility at Oregon State University. The purpose of this facility is to test the viability of the proposed goal of 5% negative reactivity in 5 msec. The HENRI system consists of a pressurized driver tank and a vacuumed test section. The test section is 1.83 m long and 0.04m in diameter. Ultimately, it would substitute a fuel element in the TREAT core design. Both volumes are initially separated by a rupture disk mimicking a fast-opening valve and located in the middle of a 0.6 m intermediate section. The following figure shows a 3-D rendering of the facility as it has been constructed at Oregon State University. Note that in TREAT the driver tank would be placed on top of the reactor and the test section would be pointed downward.

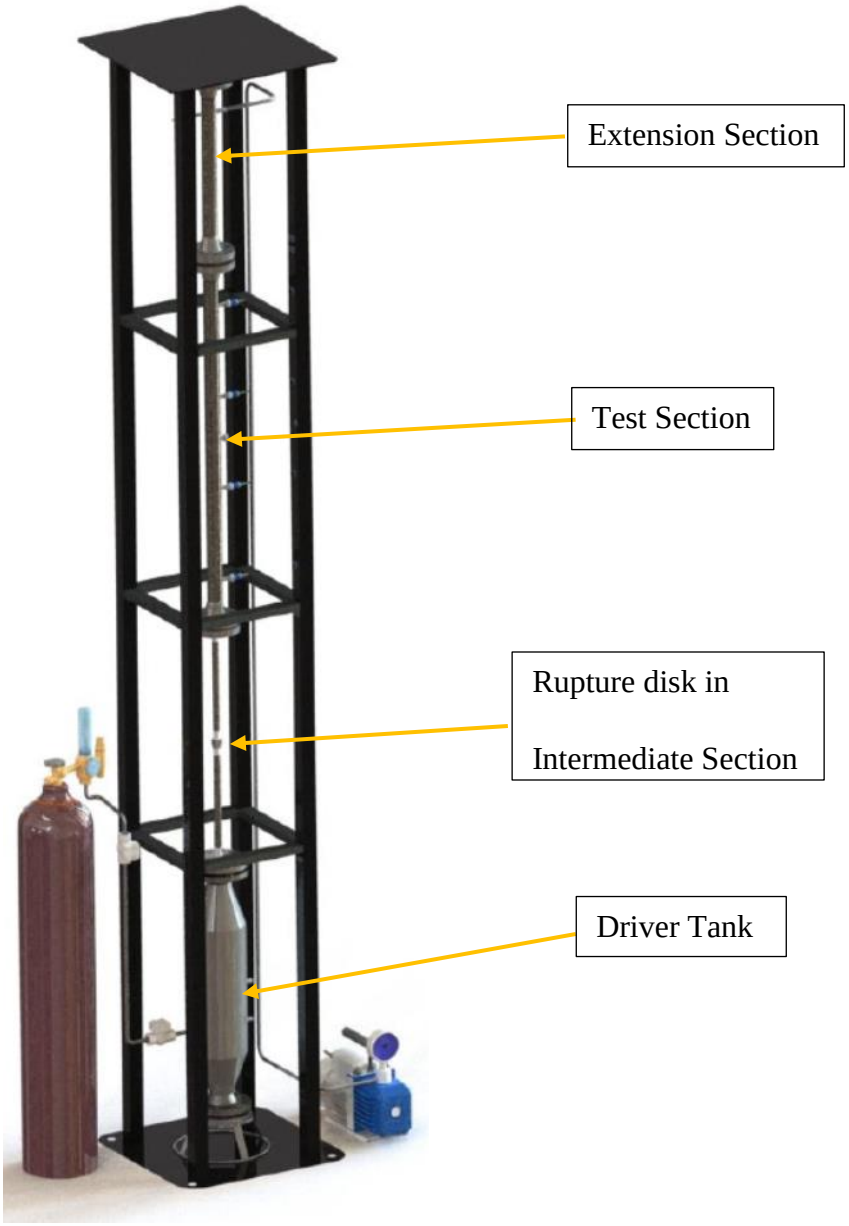


Figure 1.1.1. HENRI 3-D render

In operation the driver tank of HENRI is pressurized using helium-4 gas and the test section is evacuated. Once the desired pressure is reached in the driver tank, the rupture disk ruptures. This causes a quick blowdown of helium gas from the driver tank to the test section. The test section is lined with thermocouples and pressure transducers in order to measure the overall density evolution of the helium gas as it blows down into the test-section. A series of schematics and drawings of the out of pile HENRI system can be found in Appendix A.

The decision to use helium-3 as the neutron poison gas for HENRI creates an interesting phenomenon. Helium-3 is a low-Z material and thus undergoes neutron capture reactions. In these reactions an atom of helium-3 when struck with a neutron will absorb the incident neutron and produce by-products. These reactions create high-energy charged particles, primarily protons and tritium. The protons created from these reactions have the potential of depositing a large amount of energy locally within the helium-3 gas. That local energy deposition will cause the temperature of the gas to increase.

This leads to a highly coupled phenomena, in which the more incident neutrons strike the helium-3 gas, the more energy will be deposited locally in the gas. The more energy deposited locally, the higher the temperature increase in the gas. As the temperature of the gas increases so does the pressure. These higher temperatures and pressures can affect a variety of phenomena, such as the shockwave propagation of the helium-3 gas and the overall pressurization of the HENRI cartridge. This local energy deposition effect on pressurization and higher gas temperatures may have a noticeable effect on the density time-evolution of the helium-3 gas. The density of helium-3 gas within HENRI dictates the amount of negative reactivity being inserted into the TREAT reactor core and thus this phenomenon is important to consider.

As the flux of neutrons influences the helium-3 gas, the helium-3 gas influences the flux of neutrons. When the helium gas is injected into the system and begins to act more poisonous toward the neutron economy of the TREAT reactor core, the power of the reactor will drop. The lower the power of the reactor, the less neutrons that will be created. The smaller number of neutrons in the system the less local energy deposition will occur. A similar effect in which a local increase in pressure sees a local decrease of reactor power, named the TOP effect, has been observed in the CABRI facility [4].

This highly coupled process cannot be accurately investigated by the out-of-pile HENRI facility alone. This would require placing HENRI inside a reactor without fully understanding what effect this coupled phenomena might have on the system. Instead, it has been proposed to create computational models that can accurately model these two processes, the injection of helium-3 gas and the TREAT reactor core physics. These codes must then be capable of passing information back and forth to capture the effect of the coupled phenomena.

A computational fluid dynamics (CFD) model capable of confidently characterizing helium gas behavior inside HENRI has already been developed, verified, and validated by Balderrama [7]. This leads to the desire of building a code capable of coupling the CFD model of HENRI with a reactor physics model of TREAT in order to model the phenomena described above.

Ultimately, the work described in this study is used to support the design and implementation of the HENRI facility within TREAT.

1.1. Purpose

The purpose behind this work is to introduce a coupled code that combines the CFD model of HENRI with a reactor physics code in order to study the effect of the negative reactivity insertion due to helium-3 in TREAT and the effect of the neutronic reaction on the helium-3 fluid flow. The coupling of these two codes allows for a detailed and accurate representation of the coupled physics of the HENRI cartridges when implemented in TREAT.

1.2. Objectives

For a more accurate representation of the insertion of negative reactivity in TREAT due to helium-3, further development to the CFD model outlined by S. Balderrama [7] and the coupling of the upgraded CFD model to a reactor physics code are needed. Hence, the following objectives must be accomplished:

- develop the CFD model so that it accurately simulates helium-3 flow in the HENRI system.
- develop a base model of the TREAT facility within the Serpent 2 code that includes the HENRI cartridges placed in their expected positions.
- develop a wrapping code that can automate the execution of both STAR-CCM+ and Serpent 2.
- verify that data being passed between STAR-CCM+ and Serpent 2 is accurate.

1.3. Document Overview

This document was completed by following a systematic approach in which Chapter 2 provides a discussion of the process and application of code coupling as well as its merits in furthering multi-physics studies.

Chapter 3 focuses on the methodology followed for the development of the coupled CFD and neutronic code. This chapter goes into details about the coupling approach, the type of coupling, spatial mesh overlay, coupled time-step algorithm, coupling numerics, coupling verification, and development of the reactor physics code model.

Chapter 4 outlines the results obtained from various runs used to verify the coupling and then simulate various proposed scenarios. The results of which are presented along with an interpretation of said results.

Chapter 5 presents the conclusion section. Here the results obtained from the study are mentioned as well as an overview of the limitations encountered, and suggestions on future work are provided.

2. THEORETICAL BACKGROUND

The sections covered in this chapter are meant to support the assumptions and decisions made towards the development of the externally coupled code detailed in this thesis. The first section outlines the concept and basic implementation of externally and internally coupled codes and shows past coupling efforts of various CFD codes with various reactor physics codes. This review shows the work that currently exists and provides context for the work outlined in this document. The second section consists of describing the properties of existing and available reactor physics software. This section outlines why Serpent 2 was chosen and the properties of the code that makes it attractive when considering coupling

2.1. Overview of Code Coupling

The coupling of simulation codes is of growing interest within the nuclear engineering discipline. There exists a wide range of studies looking into the possibility of creating multi-physics simulations which can not only capture the thermo-hydraulic effects observed within nuclear reactors but also simultaneously capture the neutronic effects and how these two distinct phenomena, specifically gas fluid flow and the neutron capture reaction, interact with each other. To this end, there exists a body of work detailing these available couplings, which in part will be examined here. An extremely comprehensive and detailed examination of the coupling of neutronics and thermal-hydraulic system codes can be found in the Neutronics/Thermal-

hydraulics Coupling in LWR Technology: State-of-the-art Report [9] and Zhang [10]. A shorter summary is provided here, in which the basic tenants of coupling are discussed.

As discussed in both [9] and [10], there are four major considerations involved in coupling a code. These being the execution of the codes, whether the coupling is to be internal or external, the geometric coupling of the models, and the temporal coupling of the models.

The execution of the codes can be done in two ways, either in serial or parallel. If the coupling involves two codes, called Code1 and Code2, serial execution involves Code1 being executed until completion of its first step, then falling “asleep”, then having Code2 start executing, and falling “asleep”, after which Code1 is executed again. This process of serial execution will continue until the defined problem is finished. Parallel execution involves both codes being executed simultaneously throughout the simulation process. A visual representation found in [10] is shown here:

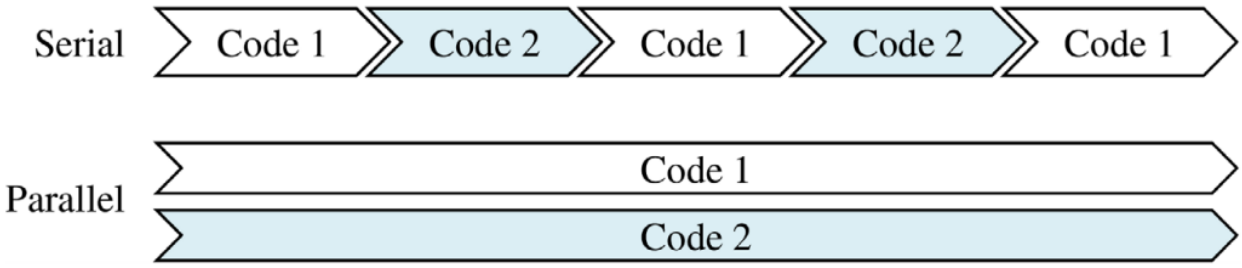


Figure 2.1.1: Serial vs parallel execution [10]

An important decision to make when coupling codes is the type of data flow coupling, of which there are two distinct varieties, internal and external. Internal coupling involves the integration of the neutronics code as another module of the thermal-hydraulics code or vice versa. With internal coupling, information is directly exchanged through memory of the codes. This kind of coupling generally requires one or both of the codes to be designed for this or the user having access to the source code in order to make the necessary changes to implement internal coupling.

External coupling of a code involves the communication between codes using input and output files where data can then be read and exchanged. This is usually achieved using an external wrapping code which will execute the coupled codes in serial and then handle the manipulation of

data such that each code receives the inputs that are desired. This method is much simpler to implement in most cases, except when the codes are designed to be internally coupled.

External coupling in general is the more popular type of coupling when considering the coupling of neutronics codes to thermal-hydraulic/CFD codes. Numerous examples exist of externally coupled neutronics and thermal-hydraulics codes, while internally coupling is much rarer.

Another decision to make when coupling is the way that the mesh information is developed and passed. This is important since, in general, each code will decompose the problem into different size meshes which it will solve over. These mesh sizes are usually determined on a code-to-code basis and are dependent on the equations being solved and the desired resolution of the results. In general, the thermal-hydraulic/CFD codes will have fine meshes. Neutronic codes on the other hand generally do not have pre-defined meshes, but the information can be manually decomposed into representative meshes.

This leads to the decision on how the information of these differently meshed codes will be transferred. As stated in [10], there are three general approaches, user-defined, user-toolkit, or third-party software. The user-defined method involves developing the mesh passing methodology on a code-to-code basis. The user-toolkit is when a tool is developed to automatically handle the passing of mesh information, this method is generally more efficient but is usually constrained to specific problems. The use of third-party software is powerful and can be extremely useful, however they usually require codes to have pre-defined meshes. Since most neutronics codes do not have pre-defined mesh definitions it cannot be used.

The final major consideration involves the temporal side of the coupling. The previous considerations primarily deal with the spatial/geometric considerations, which covers how the information is passed between the two codes. The temporal side deals with the time when the information is passed.

There are in general three ways of performing temporal coupling, explicit, semi-implicit, and implicit. The explicit method is the easiest to implement and involves the passing of data only during the start and end of each timestep. When using this method, the convergence of the solution would be dictated by each of the codes separately, this can lead to unstable/unsteady/uncovered solutions if the codes are not properly converged. The semi-implicit method involves the codes

iterating over each timestep until they both reach convergence over the solution. This method is generally more robust but increasingly difficult to implement. The implicit method is by far the most robust as it involves the passing of information between each solving step, rather than just over each timestep. Unless a code is originally designed with this type of coupling in mind, it requires significant modification of the source code and is extremely time-intensive to implement. A visual representation of temporal coupling derived from [10] is shown here:

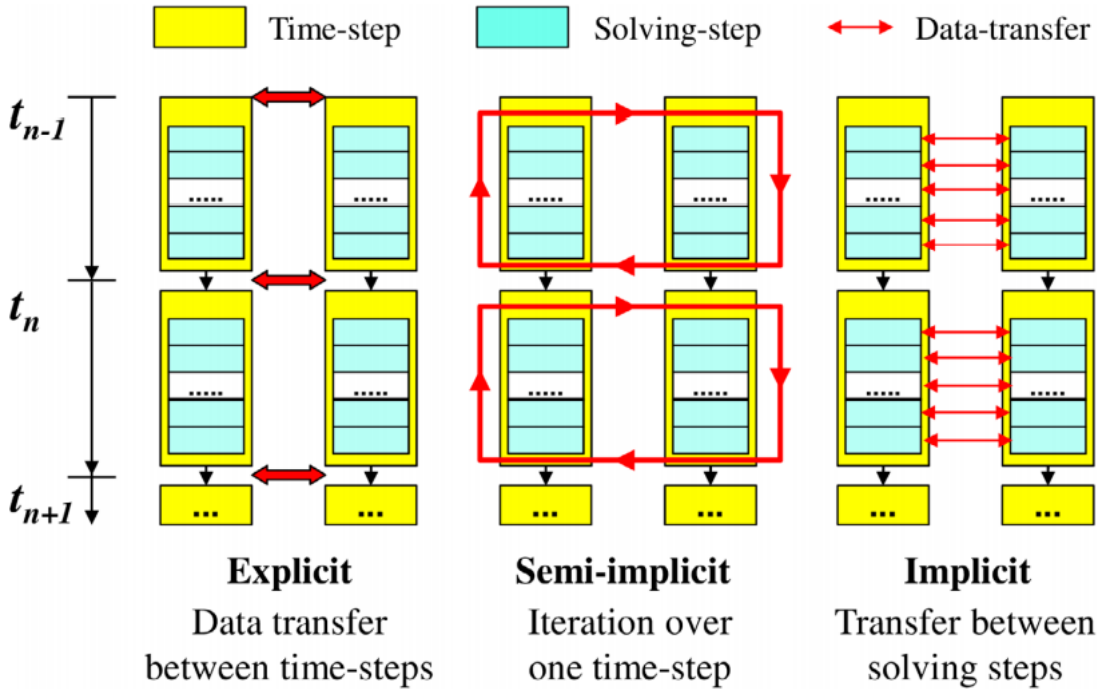


Figure 2.1.2: Temporal coupling [10]

2.2.Overview of Neutronic Codes

When selecting a neutronics code to be coupled with the already developed STAR-CCM+ model, there were three major considerations. The ability and ease of the code to be coupled, the code's ability to handle transient situations, and the availability of the code.

In general, there are two types of neutronics code. They are differentiated by how they go about solving Boltzmann's neutron transport equation, shown here.

$$\frac{\delta n}{\delta t} = \hat{\Omega} \cdot \vec{\nabla}\psi(\vec{r}, E, \hat{\Omega}) + \sum_t (\vec{r}, E)\psi(\vec{r}, E, \hat{\Omega}) = \iint_0^\infty \sum_s (\vec{r}, E \to E, \hat{\Omega} \cdot \hat{\Omega}')\psi(\vec{r}, E', \hat{\Omega}')dE'd\hat{\Omega}' + S(\vec{r}, E, \hat{\Omega}) \quad \text{Eq.(1)}$$

This equation is essentially a large balance equation tracking the neutrons gained and lost at an energy level during the transport of neutrons through some medium [23]. The main purpose for solving this equation is to solve for the scalar flux of neutrons or the eigenvalue/criticality of the solution.

The eigenvalue, also referred to as the multiplication factor, of a nuclear system is a term that defines how the neutron population is growing within a system. The multiplication factor is defined below:

$$k = \frac{number\ of\ neutrons\ in\ current\ generation}{number\ of\ neutrons\ in\ previous\ generation} \quad \text{Eq.(2)}$$

This factor can be used to quickly determine if a nuclear system is in a critical, subcritical, or supercritical state. If k =1, the number of neutrons per generation is not changing and the system is critical. If k >1, the number of neutrons in the system is increasing per generation and the system is supercritical. If k < 1, the number of neutrons in the system is decreasing per generation and the system is subcritical. In a normally operating nuclear power reactor, the k of the system is kept as close to 1 as possible.

The two methods generally used when solving the neutron transport equation are the deterministic and Monte-Carlo methods.

The deterministic method solves the transport equation by discretizing the equation to form a system of algebraic equations which can then be solved. A detailed review of deterministic methods can be found in [24]. A major drawback of discretizing this complicated equation is that solving the transport equation in this manner can introduce many errors and unknowns without using a sufficiently large grid size.

The other common method of solving the transport equation is through the use of Monte Carlo methods. Instead of solving the neutron transport equation by discretizing it, the Monte Carlo

method uses a form of random sampling that tracks each individual neutron as it passes through a defined geometry. This method tracks and categorizes the “history” of each neutron as it is “born” within the system.

It accomplishes this by randomly sampling from a specified source probability distribution. Once the neutron is “born” it will travel in a straight path until it interacts with a material, defined by a randomly sampled interaction probability. The neutron will then interact based on the cross-section data for that material. This will continue until the neutron leaves the defined solution area or is absorbed. Once this occurs that “history” is saved and then the next neutron is “born”, and the process repeated for the number of requested “histories”. Since the Monte-Carlo method does not require some of the approximations used in the deterministic method, a potentially more accurate solution can be obtained.

A draw back of the Monte-Carlo method is that depending on the complexity of the geometry being studied, a large number of “histories” may need to be simulated in order to obtain reasonable convergence of the transport equation. This can lead to large simulation times even when using supercomputers.

Traditionally, codes utilizing deterministic methods have been favored over Monte-Carlo methods. This is due to the high-computational costs associated with Monte-Carlo codes. However, with the continued development of computational capacity, Monte-Carlo codes have been used more frequently. In this review, both deterministic and Monte Carlo codes were considered.

Due to the transient nature of the HENRI test facility, a neutronics code that has been tested and verified for transient scenarios is desired. To determine the ideal code to use for the modeling of the HENRI facility a literature review was conducted to determine the previous use of various neutronics codes. The following table summarizes the codes that were studied. Table 2.2.1 presents the type of solver (Monte Carlo/Deterministic) used when solving the transport equation (Eq.(1)) and if the code has been used for steady state or transient simulations. Interest was focused on codes that have been previously coupled and contain subroutines that facilitate coupling.

Table 2.2.1: Classification of considered neutronic codes

Code	Type	Use
MCNP5	Monte Carlo	Steady State
Serpent 2	Monte Carlo	Reactivity Transients/Steady State
TRIPOLI	Monte Carlo	Steady State/Transients
RMC	Monte Carlo	Transients
PARCS	Deterministic	Steady State/ Load Rejection Transients
RATTLESNAKE	Deterministic	Transients
SCALE (KENO-IV)	Monte Carlo	Steady State (Uncertainty Analysis)

2.3. Overview of non-CFD/neutronic coupled codes

The process of coupling a deterministic neutronics code (PARCS) to a thermal-hydraulic system code (RELAP5) [16] was done during the power uprate process undertaken by the Ringhals-3 reactor in Finland. A model simulating the full reactor was developed.

The development of the model took three steps the first step utilized the codes CASMO and SIMULATE in order to understand the neutronic history of the reactor which would then be fed into PARCS. These simulations were used to provide the 3-D spatial distribution of the history of the moderator density, boron concentration, and the burnup. This was then used to correct the material data for each fuel/reflector segment in the PARCS model. To obtain macroscopic cross-sections, discontinuity factors, microscopic cross-sections and yields for the poison PMAXS files were used which were constructed from CASMO-4 calculations.

To develop a model in PARCS the first step was to model an actual core loading in which one hundred and fifty-seven fuel assemblies constitute the core. The second step in the setup is to associate a PMAXS file to each of the segments however the actual heterogeneity of the core was considered with the core being represented by a large number of regions. This standalone calculation was run to determine the accuracy of the PARCS model and was found to be in very good agreement with the data gathered.

The RELAP model was then developed with a rather fine nodalization, and a complex control system meant to model plant responses.

The coupling between RELAP5 and PARCS was done in an external manner with a parallel virtual machine and used an internal integration scheme in which the solution of both the system and core

thermal-hydraulics is performed by RELAP5, and the spatial kinetics solution is performed by PARCS. The mapping between the neutronics and the thermal-hydraulics is handled by mapping the neutronic/hydrodynamic structures and the neutronic/heat structures. The first mapping allows specifying in the neutronic code which fuel temperature, moderator temperature/density from the thermal-hydraulic code needs to be associated with a specific neutronic node. The second mapping allows specifying in the thermal-hydraulic code which fission power needs to be associated with a specific thermal-hydraulic node [16].

PARCS has a built-in ability to generate the mapping file necessary however it is not ideal for complicated problems thus a separate tool for generating the mapping tables was created. Each mapping file requires a weighting factor between the neutronics and the thermal-hydraulics in which the weighting factors of the RELAP5 components and the cells belonging to the PARCS nodes sum to unity. The weighting factors thus represent how much a RELAP5 component contributes to a PARCS node and how much the PARCS node contributes to the heat generated in the RELAP5 component. It has been shown that the coupled codes in general can reproduce all the general trends in the load rejection transient. Matching of the recorded plant parameters and the code calculation was found to be good in most cases. Disagreement was found in areas where operational experience and operational data were lacking.

Another type of code, the Monte-Carlo neutronics code Serpent 2, was used to model and simulate the Special Power Excursion Reactor Tests (SPERT), which was a series of reactor experiments done at Idaho National Laboratory in the late 50's [25]. The reason Serpent 2 was selected for this was its high level of flexibility and its advanced time-dependent capabilities. A detailed explanation of Serpent 2 time-dependent simulation capabilities can be found here [26], while a basic summary of its capabilities is presented here:

In order to simulate a transient, the Serpent 2 code requires an initial neutron source file, from which it can then track the neutrons through time and space. Serpent 2 has the capability of generating this source file itself and it can be obtained by running the code in its source generation mode. In this mode Serpent 2 will save the location, direction of travel, weight, and energy of the user-defined number of live neutrons to the source file. It will also calculate and save delayed neutron data to a separate file. A delayed neutron being a neutron that is emitted after a fission event by one of the fission products. These delayed neutrons are important when controlling the

criticality of a reactor and thus the information about them must be captured for an accurate transient simulation.

Once the source files have been generated a transient simulation can then be run using Serpent 2. Serpent 2 will first normalize the weight of the neutrons so that the neutron population in the transient simulation matches that of the source simulation. In a transient simulation the execution of Serpent 2 is subdivided into time “bins”. Each bin corresponds to a user-defined length of time in which the transport equation (Eq.(2)) is solved. For every time bin the code will simulate the neutron population that was live for the previous time bin and the number of delayed neutrons that should be born in that time bin based on information from the previous time bin. It will continue this process until all time bins are simulated.

The SPERT experiments were a series of power-excursion experiments in which a control rod within the reactor was ejected rapidly. This causes the reactor power to increase quickly. These SPERT tests were used to verify early reactor kinetics tools and have been used in that manner ever since. When simulating the SPERT experiments using the time-dependent methodology found within Serpent 2 it was found to adequately model the experimental results.

Another attractive feature of Serpent 2 is that it is specifically designed with the coupling of external thermal-hydraulic codes in mind. For the external coupling of a Monte Carlo code and a CFD code, the Monte-Carlo geometry needs to be subdivided into a multitude of cells which corresponds to the thermal-hydraulic information if the Monte-Carlo tool does not provide a standardized coupling interface. This can than introduce significant error but in the case of Serpent 2 a standardized coupling interface is provided thus reducing this potential error.

2.4.Overview of Existing CFD/Neutronic Coupled codes

In an effort to understand the current existing body of work involving the coupling of CFD and neutronic codes, a review was performed. This review focused on the coupling of the neutronic codes discussed in Section 2.2with a variety of CFD codes. The purpose being to acquire a base level of knowledge in how these coupling were performed and to leverage previous work if possible. A variety of codes were studied, however a focus was applied to two specific codes (MCNP-5 x STAR-CD) [13] and (TRIPOLI x CFX) [14]. The reason being that the methodology

provided in both of these codes allows for the framework in which the coupled code discussed in this report was developed upon.

One of the external coupling codes that was studied exists within the family of Siemens codes, of which STAR-CCM+ is a member. The code uses STAR-CD and the Monte-Carlo neutronics code MCNP5 [13]. A model of a 3D 3X3 array of PWR fuel pins in particular was studied where MCNP5 was used to simulate the transport of neutrons while thermal-hydraulic feedback data was obtained with STAR-CD. An interface program to couple the MCNP and STAR-CD codes was written. This program, which is named McStar, is a FORTRAN90 program that can execute MCNP and STAR-CD alternately until the eigenvalue and flux are converged.

McStar is used to perform some necessary input and output manipulation. Some manipulations to the MCNP input are performed in McStar to reflect the temperature effect in the Monte Carlo calculation. The cross-section file is rewritten with the new generated cross section data. The MCNP input file is then modified by replacing the calculated temperatures and cross section data.

Prior to the iterative coupled calculation, a standalone STAR-CD calculation is performed with an initial power profile to obtain an initial temperature distribution for the MCNP cross section library. MCNP is also run with a standard library to obtain an initial source file. The iterative calculation begins with the MCNP calculation which calculates the normalized power and saves it to an external file. This file is then read by STAR-CD to update the power profile in the CFD calculation. STAR-CD then uses the MCNP power distribution to calculate new temperature, density, and volume data which is passed back to MCNP/NJOY.

This code is of particular interest as it provides a general methodology that can be followed for this work when using Siemens codes. This is due to the Siemens codes using the same methodology of automatically importing and exporting data during their execution using JavaScript MACROs.

Another externally coupled code of interest in this study is the coupling of the Monte-Carlo code TRIPOLI with the CFD code CFX [14]. The reason this code is of particular interest is that it involves the use of an external coupling code to study a reactivity insertion transient within a TRIGA reactor and thus provides a general methodology for external coupling when considering transient situations.

The TRIPOLI code model consists of the fuel elements and the control rods as well as the supporting grid, graphite reflector, irradiation channels, graphite of the thermalizing and thermal column, carrousel with explicitly modelled irradiation tubes, neutron source, triangular irradiation channel, radial and tangential beam ports. This provides a rather accurate model of the TRIGA reactor

The ANSYS CFX code was used to develop two models one for the entire pool (6.5 m high cylinder of 2m diameter) called the “full pool” model where the core consisted of 58 fuel rods, 4 control rods, and 5 irradiation channels and the second smaller and refined model called the “core” model where the fuel elements and control rods containing fuel parts are fully modelled (rod, fuel, and cladding). In the smaller model the velocity distribution and temperature are imposed according to the converged steady state of the full pool model.

The coupling and convergence of the TRIPOLI and ANSYS CFX model methodology is described in depth with the code NJOY being used to generate nuclear cross-section at various temperatures.

The coupling algorithm used is explained in the following five steps.

Step 1: The initial power density distribution is obtained with critically at room temperature which is then normalized to obtain the desired thermal power.

Step 2: The CFD code is initialized to obtain the new temperature distribution, the power distribution is updated with the previous iteration value

Step 3: The previously calculated temperature distribution is used by NJOY via TRIPOLI to calculate the new microscopic cross-sections. Additionally, the requested temperature and post processing of the nuclear data at the format needed by TRIPOLI.

Step 4: The power density distribution is evaluated by simulating a number of neutron histories with TRIPOLI in criticality mode. The fuel and coolant properties in the Monte-Carlo calculations are updated.

Step 5: Relaxation is applied to the flux distribution.

The algorithm is then stopped when chosen convergence criteria is met.

The transfer of data between TRIPOLI and CFX is handled carefully since the number of compositions in TRIPOLI is lower than the number of discrete volumes in the CFX model. This means that TRIPOLI has a much coarser mesh than the CFX model and interpolation is done to transmit the correct power density to the CFX mesh elements. The actual transfer of data is done through a FORTRAN wrapping text where the fluid quantities are averaged and then a shell script is used in order to input the composition file of TRIPOLI. The calculated power density is read from the TRIPOLI output then modified and interpolated by FORTRAN.

A rod extraction, transient operation of the reactor, was studied. Initially, the reactor is critical at 61 kW. At the time t=0 s, the transient rod is withdrawn partially in about 4 s (from position 407 to 336) leading to reactivity increase of 220 pcm (inferred from control rod worth curves). Physically, reactivity is inserted in the system; power starts to increase. Energy released heats up fuel elements, their temperatures increase, TRIGA fuel has large prompt negative temperature coefficient of reactivity: the temperature rise makes the reactivity decrease until the reactor reaches a new critical state at power around 130 kW. The coupled model shows good agreement with the experiment data collection with less than 3% discrepancies.

In an effort to model transient flows in TREAT the Monte-Carlo neutronics code Reactor Monte Carlo (RMC) and the thermal-hydraulic code ANSYS Fluent were externally coupled [18]. Fluent was chosen as the geometry of the reactor core and reflector are relatively simple.

The RMC code uses the Predictor-corrector quasi-static (PCQS) method. This method factorizes the neutron angular flux into an amplitude function which can then be manipulated to form the point kinetics function and the shape function. The PCQS is used to solve the amplitude function as it changes rapidly and must be solved using micro time-steps, the PCQS was originally developed for deterministic calculations but has since been studied for use in Monte Carlo calculations with it being fully implemented in RMC.

RMC and Fluent are linked through a file sharing system and an overlapping python script, making the coupling scheme used for these codes an external one. RMC communicates with FLUENT through output and input files. RMC outputs the total fission power and neutron flux for each coded mesh at the end of every time step. The power density for FLUENT is then obtained, once RMC has gone through one time step the calculation is paused and the information is passed to

FLUENT which then calculates the temperature and uses the energy source from RMC to calculate a temperature distribution.

The python script then takes this calculated temperature distribution and passes it back to RMC which then uses the new information to calculate new cross sections based on Doppler broadening and the newly calculated power distributions. The python script then ensures convergence before the process is started over. This method only considers changes in temperature.

The Monte-Carlo neutronics code TRIPOLI was used to calculate the 3-D core depletion of a pool-typed (heavy water) reactor [27]. The analysis showed the reliability of TRIPOLI-4 to calculate a complex configuration. The depletion calculation was carried out on a model of the ORPHEE research reactor where the results are validated by comparison with the deterministic transport code APOLLO2 which ran a similar depletion calculation on a 2-D core configuration.

Tripoli's depletion capability relies on two C++ interfaces, the first which is a set of generic Monte Carlo neutron transport equation which allow for the simulation to run while the second provides a way to perform material depletion calculation using a fourth order Runge-Kutta numerical approximation. In most Monte-Carlo codes the calculated isotope weights are provided with statistical uncertainty however in most burnup calculating the coupling script omits the statistical uncertainty. To ensure the statistical uncertainty is maintained throughout the burnup calculations independent simulations are run through each of the different burnup steps.

The calculations run on the irradiated core did not take into consideration the thermal-hydraulic feedback effects (no coupling of neutronics and thermal-hydraulics) in the scope of the calculations this was deemed acceptable as the thermal-hydraulic feedback in a depletion calculation is minimal.

The Monte-Carlo neutronics code KENO-IV, also referred to as SCALE, was used to benchmark and quantify uncertainty in the analysis of light water reactors [28]. KENO-VI has convenient geometry input, including the treatment of lattice arrays of materials. The principal result from the neutron transport calculation by the KENO-VI code is an estimate of the multiplication eigenvalue k, and is calculated for each active generation

Table 2.4.1 shows a comparison between the various neutronics, non-CFD, and CFD coupled codes that were studied in this work. If the type of coupling (internal vs external) is known, it is

listed. As it is important to understand for this study, the capability of the coupled codes to simulate transient conditions is also color coded in the table. Where coupled codes that simulate transient conditions are marked with red and coupled codes that simulate steady-state conditions solely are marked in light green.

Table 2.4.1: Comparison of previously coupled neutronics and thermal-hydraulic codes

		Non-CFD				CFD			
		SUB-CHANFLOW	RELAP5	BISON	MELCOR	CFX	Fluent	Star-CD	StarCCM+
Deterministic	RATTLESNAKE	-	-	Coupled [19]	-	-	-	-	-
	PARCS	-	Coupled [16]	-	-	-	-	-	-
Monte-Carlo	MCNP	Internal[11]	-	-	-	External [12]	External [15]	External [13]	-
	TRIPOLI	External[11]	-	-	-	External [14]	-	-	-
	RMC	-	-	-	-	External [17]	External [18]	-	-
	Serpent	-	-	-	Coupled [20]	-	-	-	-
	Serpent 2	External/Internal[11]	-	-	-	-	-	-	Current Coupling

With all the neutronics codes documented here considered, the Serpent 2 code was chosen for this work. The reason being that Serpent 2 is a code that is readily available, capable of simulating transients, and offers a standardized coupling interface. The standardized coupling interface, in particular, allows for ease of coupling and reduces potential errors when passing data back and forth between the two codes.

It should be noted that Serpent 2 does have the capability of being internally coupled as seen in its internal coupling with the SUBCHANFLOW thermal-hydraulics code [3]. However, STAR-CCM+ is a commercially available code and access to its source code is not feasible within the scope of this study therefore the external coupling capabilities of Serpent 2 are studied.

3. MODEL AND METHODOLOGY

The use of coupled CFD/thermal-hydraulic codes with reactor physics models has become more and more popular. The ultimate goal of the coupling of codes is to accurately model and predict the physical behavior of a nuclear power plant, and in particular its behavior during transient situations. The use of such a coupled code can be beneficial from a research and design standpoint as it can be used to help inform engineering decisions without having to build new experimental facilities, thus reducing costs. The development of such models is not meant to replace experiments outright, instead it is meant to support experiments and provide a tool for more comprehensive visualization and understanding. In the case of the HENRI facility, a coupled code allows for a detailed understanding of the effect that the injection of helium-3 gas would have on the neutron economy of TREAT and vice versa. This understanding can help with the eventual final design and installation of the HENRI facility inside TREAT .

The first step taken in this study was to take the STAR-CCM+ model of the HENRI facility created by Balderrama [7] and upgrade it to simulate helium-3. This original model was verified and validated for helium-4 using steps defined in the ASME V&V 20-2009 [29]. After the model was upgraded to simulate helium-3 the coupling process was undertaken.

The development of coupled codes can be a long and complicated process depending on the application and scope expected of such a code. Some coupled codes can be used to simulate entire nuclear reactor systems, including but not limited to, coolant flow, fuel burnup, and core power. While the code described in this report is used for a much less complicated scenario it is still important to follow a general methodology when developing a coupled code. To ensure that the coupled code provides a detailed and accurate solution, it is common to follow six basic components of coupling methodology, listed here:

1. Coupling Approach – serial or parallel processing
2. Type of Coupling – internal or external
3. Spatial Mesh Overlay
4. Coupled time-step algorithm
5. Coupling numerics – explicit, semi-implicit, and implicit.
6. Coupling verification

3.1. Helium-3 Verification

The original CFD code developed and discussed in detail by Balderrama [7] used helium-4 gas in its execution. The reason for this is primarily because the neutronic properties of helium-3 were not a factor in the physical characterization of the HENRI cartridge in an out-of-pile facility. Also, helium-3 gas is a rare and expensive gas, and it would not have been economically feasible to conduct the physical testing outlined by Balderrama with helium-3. Therefore, helium-4 was substituted for both the physical testing and CFD modeling efforts. This decision to use helium-4 in both aspects allowed for the CFD model to be validated against the out-of-pile test facility built at OSU, however, it failed to fully reproduce the physical behavior that one could expect with the use of helium-3 and its associated nuclear reaction in the TREAT facility.

The main reason for this is the slightly smaller size of helium 3, consisting of two protons and a neutron with an atomic mass of 3.016 u, while helium-4 consists of two protons and two neutrons with an atomic mass of 4.0002 u. This 30% lower mass can lead to a faster shockwave propagation. STAR-CCM+ allows for new materials to be defined and implemented, as long as a few of the thermodynamic properties are provided, thus an effort was made to calculate the properties requested by the STAR-CCM+ code as well as others in order to quantify the difference between helium-4 and helium-3 to determine if making changes to the model was necessary.

Petersen [8] presents a formulation for calculating the thermodynamic properties for helium gas from pressures ranging from 1 to 100 bar and temperatures ranging from 298 K to 1800 K. The formulation described within assumes that helium is an ideal monoatomic gas which due to its size and atomic composition, is an acceptable assumption for the purpose of this study. The way that the equations are formulated in Petersen allows for the calculation of the thermodynamic properties for any helium isotope. The specific heat is of particular interest as it will dictate how quickly the helium-3 gas will heat up when exposed to the neutron flux and the subsequent heat generated by the neutron absorption reaction, indicative of helium-3.

When calculating the specific heat of helium, the universal gas constant, R, and the molar mass, M, of the isotope must be input. With these known values the specific heat at a constant pressure, C_p, and the specific heat at a constant volume, C_v, can be calculated using the following equations.

$$C_p = 2.5 * \frac{R}{M} \qquad \text{Eq.(3)}$$

$$C_v = 1.5 * \frac{R}{M} \qquad \text{Eq.(4)}$$

In addition to specific heat, other useful thermodynamic properties including the viscosity, μ, and thermal conductivity, k, of helium were compared. The viscosity as it would play a factor in the time the helium-3 takes to travel across HENRI through friction, and the thermal conductivity for how quickly the heat generated by the nuclear capture reaction would dissipate through the gas. As stated by Petersen the general equation for the viscosity of helium, which is taken from an intermolecular potential function can be expressed as follows:

$$\mu = 2.6693 * 10^{-6} * \frac{\left(\sqrt{M * T}\right)}{\sigma^2 Q^{2.2} + T^x} \qquad \text{Eq.(5)}$$

Where, M is the molecular weight of helium, T is the temperature of helium, sigma is the potential parameter, and $Q^{2.2} + T^x$ is the collision integral. The equation itself is derived from a complex system of equations which considers interatomic attraction and the collision integral correlated from experimental data. Petersen proposes a simplified function of the form:

$$\mu = a * T^b \qquad \text{Eq.(6)}$$

Petersen explains that for most gases, this simple correlation is not very good, however due to helium having extremely small interatomic attractive forces, this equation is far more superior than most. Petersen uses the more common helium-4 isotope to solve for a general form of the dynamic viscosity of helium dependent only on the temperature, however a simple substitution of M in Eq.(6) allows for a general form of helium-3 dynamic viscosity to be obtained. The equation for dynamic viscosity, dependent only on temperature, is shown here for both helium-4 (Eq.(7)) and helium-3 (Eq.(8)).

$$\mu = 3.674 * 10^{-7} * T^{0.7} \qquad \text{Eq.(7)}$$

$$\mu = 3.182 * 10^{-7} * T^{0.7} \qquad \text{Eq.(8)}$$

With the dynamic viscosity solved for, Petersen presents a formula for the thermal conductivity, k, of helium dependent on the dynamic viscosity, μ, and specific heat at a constant volume, C_v . This equation is as follows:

$$k = 2.5 * \mu * C_v \qquad \text{Eq.(9)}$$

Note that the values needed for updating the STAR-CCM+ model were simply the molar mass and the specific heat, as the STAR-CCM+ code then uses Sutherland's Law to calculate dynamic viscosity and the thermal conductivity. However, obtaining the dynamic viscosity and thermal conductivity values using these relations allowed to gain some confidence that updating the simulation from using helium-4 gas to helium-3 gas properties was a worthwhile endeavor as long as the thermodynamic properties were different enough to warrant investigation.

After implementing the helium-3 properties following the formulation presented above, the equations listed, and their results were verified by calculating the values for helium-4 at 300K and then comparing them to the values provided by the Engineering Equation Solver (EES) code v10.36-3D. Once the equations and their results had been verified the corrected values were calculated for helium-3 and then implemented into the STAR-CCM+ model.

Once the thermodynamic properties obtained from these calculations were implemented in STAR-CCM+, the second step was to verify the model by conducting a shock tube analysis. It consisted of a simple shock tube geometry with a 0.8m long high-pressure section and a 1.85m long low-pressure section separated initially by a fictitious boundary. Such a problem can be derived as a simple 1-D approach. The shock wave velocity and density front propagation can be solved analytically. These analytical results were then compared with the shock tube simulation performed by STAR-CCM+ to verify that the new helium-3 model could accurately predict the solution. A similar shock tube analysis was initially successfully performed for the helium-4 model verification work conducted by Balderrama[7]. Figure 3.1.1 shows the results of this shock tube analysis.

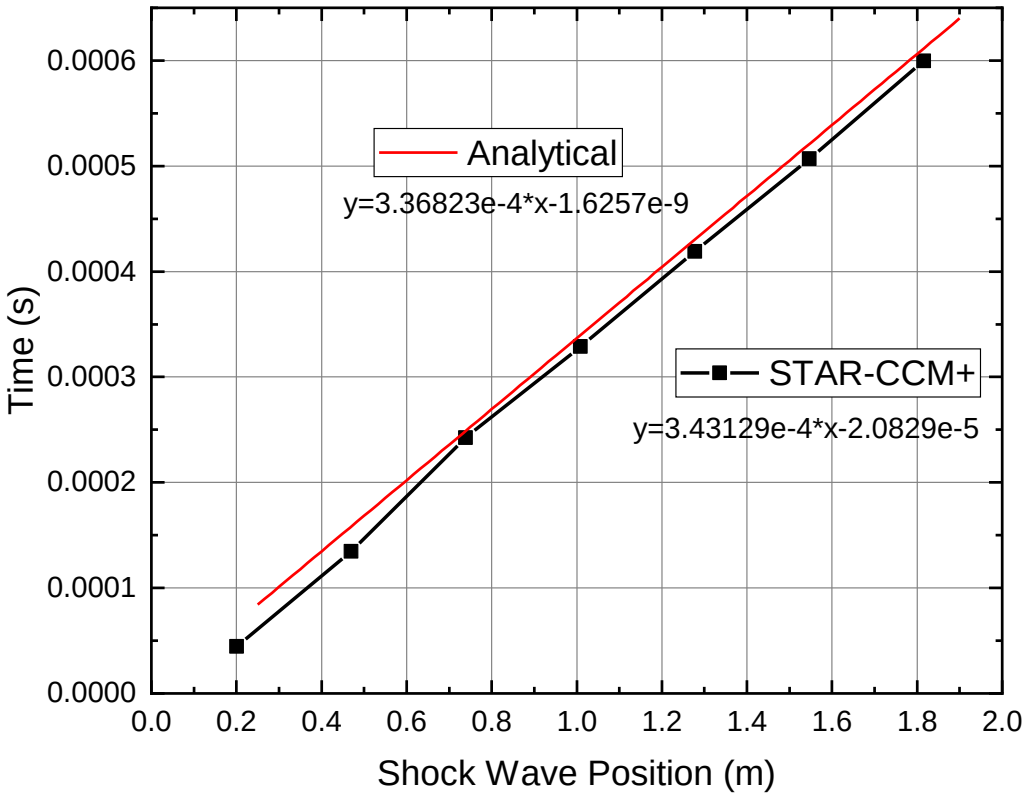


Figure 3.1.1. Helium-3 shockwave comparison of analytical solution and STAR-CCM+

Figure 3.1.1, compares the shockwave propagation that is calculated using analytical shock wave equations and one that is calculated by a STAR-CCM+ simulation of a shock tube. The slope of the linear fit line for the analytical helium-3 calculation and STAR-CCM+ helium-3 shock tube simulation are very similar, with a calculated 1.9% deviation between the two slopes. This leads to the conclusion that STAR CCM+ is capable of accurately modeling helium-3 shock wave propagation for use in further simulations.

3.2.Coupling Approach

When executing two codes for the purpose of coupling, an important first step is to decide what kind of processing structure to use. There are generally two types of processing, serial and parallel. As mentioned in Section 2.1, these methods correspond to how the two codes will be executed relative to one another. When considering the coupling of STAR-CCM+ to that of Serpent 2, the serial execution of the codes was chosen. This stems from the capabilities of both codes.

On the one hand, STAR-CCM+ is a commercially available code with no built-in subroutine to allow for parallel processing. This means that modification of the source code, which would be necessary to implement parallel processing, is not available. Instead, the execution and interaction of the code outside of the built-in Graphics User Interface (GUI) must be done using a JavaScript macro. The purpose of a STAR-CCM+ macro is to automate repetitive tasks or manipulate options while the CFD simulation is running. One such macro was developed that allows STAR-CCM+ to be executed in serial with another code, using a signaling file.

STAR-CCM+ has built-in subroutines that facilitate the making of these macro files, in which actions within the STAR-CCM+ GUI, such as execution signals or modification of built-in options, are automatically translated to the JavaScript language. This allows for no extensive knowledge of the JavaScript coding language or STAR-CCM+ objects to be required to develop a working macro file. The macro used to automate the execution of STAR-CCM+ can be found in Appendix B.

On the other hand, Serpent 2 is a code that was specifically designed to be executed in serial with another code. It does this by executing in a specific coupled calculation mode. When Serpent 2 is executed in this coupled mode it can be controlled with one of two communication modes:

1. File-based communication
2. POSIX-signal communication

The File-based communication mode was chosen for this coupling. When Serpent 2 is executing in this file-based mode it will create two files (*com.in*, and *com.out*). It is through these files that Serpent 2 can be told to either execute or fall asleep. The *com.in* file specifically controls the activeness of Serpent 2. Using an external wrapping code, a SIGUSR1 or SIGUSR2 signal can be sent to this file. A SIGUSR1 signal sent to *com.in* would inform Serpent 2 to iterate over the current time-interval, while a SIGUSR2 would inform Serpent 2 to execute the next interval. Once the specified time-interval has been simulated Serpent 2 will fall asleep and wait for the signal to be updated once again, once this happens Serpent 2 will then proceed based on the received signal.

3.3. Type of Coupling

Once the type of processing has been chosen for a coupling, the next decision is the type of coupling itself. The two distinct variates being that of an internal or external coupling. For STAR-CCM+ and Serpent 2, only external coupling is available. This is because of the inherent

capabilities of both codes. Since external coupling has been chosen it facilitates the creation of what is commonly known as a wrapping code. The purpose of this wrapping code is to facilitate the passing of input and output files between the two codes and control the serial execution of both codes. The details of the development of the wrapping code can be found in the Coupling Algorithm section (3.7).

The following sections outline the general methodology, including input and output options specific to both STAR-CCM+ and Serpent 2 that are used when developing an external coupling methodology.

3.3.1 STAR-CCM+ External Coupling

As discussed previously, STAR-CCM+ is a commercially available code and its source code is not readily accessible. This leaves its execution to only be manipulated using a JavaScript macro, which facilitates external coupling as the macro can automatically manipulate options within STAR-CCM+ using input and output files.

Specifically, the data that is desired to be extracted from STAR-CCM+ is the axial distribution of the density and temperature of the helium-3 gas within the HENRI cartridge. In order to extract this data, while STAR-CCM+ is being executed, the built-in STAR-CCM+ table functions were utilized.

STAR-CCM+ has the capability of automatically importing and exporting data during an active simulation using tables. These tables consist of tabular data that is defined using field functions. Field functions are how STAR-CCM+ can store data, such as temperature and density throughout a simulation, they can be in the form of scalars or vectors. When exporting data from a STAR-CCM+ simulation there are three things to consider:

1. the field functions being extracted
2. the derived parts from where that data is being extracted
3. when the data should be extracted.

The desired field functions are the axial distribution of the density and temperature of the helium-3 gas within the HENRI cartridge. These are the main figures of merit of the STAR-CCM+ simulation in this coupling. The axial distribution of density and temperature will be used by

Serpent 2 in order to calculate the overall volumetric heating that occurs within the helium-3 while it is being injected into TREAT.

The derived parts option defines a specific part that can track and sample the field function data. The line-probe derived part was chosen for this study to extract the density and temperature data of the helium-3 gas. A line-probe is a STAR-CCM+ part that defines a line of points located at a user-defined position and with a user-defined resolution. The resolution corresponds to the number of data points used to make the line-probe.

For extracting the density, temperature, and relative position of the helium-3 gas. A line-probe was created in the base coordinate system with points between 1.66305 m (166.305 cm) and 2.87274 m (287.274 cm), the location of which is at the TS00 and just after the TS04 data point detailed in Figure 4.3.1. This corresponds to the location of the test-section that is expected to be inserted into TREAT. A resolution of five hundred was selected. This corresponds to 0.24 cm between each point.

The update function can be used to define a time-step, iteration, or delta-time frequency over which the requested data is extracted. The time-step option was chosen to extract the information over. This extracts the axial distribution of density and temperature of the helium-3 gas after a user-defined number of time-steps.

Once all these options have been defined STAR-CCM+ will automatically export the requested data in the form of a .csv file. This file can then be read and manipulated by the external wrapping code with the relevant data being passed to Serpent 2.

When importing data during a simulation, STAR-CCM+ uses much of the same methodology as when it is exporting data. Using the built-in table functions STAR-CCM+ can automatically import data from a user defined .csv file that contains the data expected by STAR-CCM+. The expected data depends on the STAR-CCM+ subroutine that the data is being imported too.

In the case of this coupling that is the Volumetric Heating Source option, which expects units in W/m^3. This option is chosen as it allows for the definition of the heat released from the helium-3 capture reaction. It is the only heating option provided by STAR-CCM+ that allows the definition of heating to be imported along the x, y, and z axes.

3.3.2 Serpent 2 External Coupling

As stated previously Serpent 2 has been designed with external coupling in mind. It accomplishes this using the universal multi-physics interface. This is a built-in subroutine of Serpent 2 that has been designed to easily bring in temperature and density fields from external codes. There are multiple input options that can be used to define the type of data that will be read. The option selected for this coupling is the regular mesh-based interface. The interface input options are shown here:

Line	#1
Line	#2
Line	#3
Line	#4
Line	#5
Line	#6

```
TYPE MAT OUT
OUTFILE [omit line if OUT is set to 0]
NZ ZMIN ZMAX NR [omit line if OUT is not set to 1]
MESHTYPE
[MESH DATA]
DENS_1 T_1
DENS_2 T_2
...
```

Figure 3.3.1: Excerpt of Serpent 2 regular mesh-based interface inputs [31]

Line #1 of this input defines the interface type, in this case TYPE = 2, then the material that the density and temperature data is given for, and finally what kind of output data is requested. This output data can be output along the defined pins (2) , the same as the mesh (1), or no output (0). Line #2 is the definition of the location of the output file if one is requested. Line #3 defines the number of z points the data will be output along, the minimum z coordinate, maximum z coordinate, and finally the number of radial bins that the power distribution will be output across, if an output is defined.

Line #4 is where the user defines the type of mesh data, this can be three types of meshes. A regular cartesian mesh, a hexagonal mesh, or an irregular cartesian mesh. For this coupling, a regular cartesian mesh was chosen. This type of mesh is output by the STAR-CCM+ using the .csv file discussed in Section 3.3.1.

Line #5 is where the mesh data is defined ([MESH DATA] in Figure 3.3.1). For the regular cartesian mesh, the input options are shown here:

```
NX XMIN XMAX NY YMIN YMAX NZ ZMIN ZMAX
```

Figure 3.3.2: Serpent 2 regular cartesian mesh input [31]

Where *NX*, *NY*, and *NZ*, define the number of mesh cells in each direction. The *MIN* and *MAX* variables define the limits of the mesh in each respective direction. For a regular cartesian mesh, the cells are divided axially along equal length segments.

Line #6, following the mesh data definition, represents the density and temperature data for *NX*NY*NZ* number of points. The density can be either positive or negative. Positive densities correspond to atomic densities (atoms/cm^3) while negative densities correspond to mass densities (g/cm^3). Mass densities are used for this coupling. The order of the density and temperature data is read such that the x-index is increased first, then the y-index, and finally the z-index.

With the multi-physics interface defined, Serpent 2 can be executed in the coupled communication mode discussed in the previous Coupling Approach section.

As stated previously the main purpose of Serpent 2 in this coupling is to take in density and temperature data of the helium-3 from STAR-CCM+ and then calculate the volumetric heating that would result from the incident neutron flux found within TREAT. Once this has been calculated by Serpent 2 the next step would be to output the volumetric heating values calculated. Serpent 2 has built-in inputs called detector cards that can facilitate this process.

Detector cards in Serpent 2 are inputs that can be used to track user-defined reaction rates which are integrated over space and energy [30]. The detector response function determines the type of the calculation. Detectors are useful in outputting data as they will automatically create a cartesian mesh based on user inputs and output the requested response function data along that mesh. In the simplest case a response function of 1 is used to calculate the neutron flux integrated over space and energy. If a reaction cross section is used, the result is the corresponding reaction rate.

As the simulation of the HENRI cartridge within TREAT will be that of a coupled transient, the different ways that Serpent 2 handles coupled transients must be understood. The main difference between normal transient execution and coupled transient execution comes with how Serpent 2 outputs detector data. The detector output in transient simulations works differently depending on whether the transient simulation is coupled or not.

1. In non-coupled transient cases, the code simulates a full batch of neutrons through all of the defined time-intervals before simulating the next batch of neutrons.
2. In coupled transient cases, the code simulates all of the neutron batches through the first time-interval and then moves to the next time-interval.

Due to the difference in the simulation process, the detector results are also collected differently:

1. In non-coupled transient cases, the detector results are collected from all of the time-intervals. The output will then contain the results for all of the time-intervals
2. In coupled transient cases all output results are cleared between time-intervals. This means that during the simulation of a certain time-interval only the result from that time-interval is output to the output file.

In coupled transient simulations Serpent 2 produces one detector output file for each time interval. *[input]_det0.m* will contain detector scores from the first time-interval, *[input]_det1.m* from the second time-interval and so on.

When both the multi-physics interface inputs and detector outputs have been decided on and created, Serpent 2 can be run in coupled mode and be placed into an externally coupled system. The developers of Serpent 2 have provided a figure showing the expected external coupling methodology that can be used. It is reproduced here:

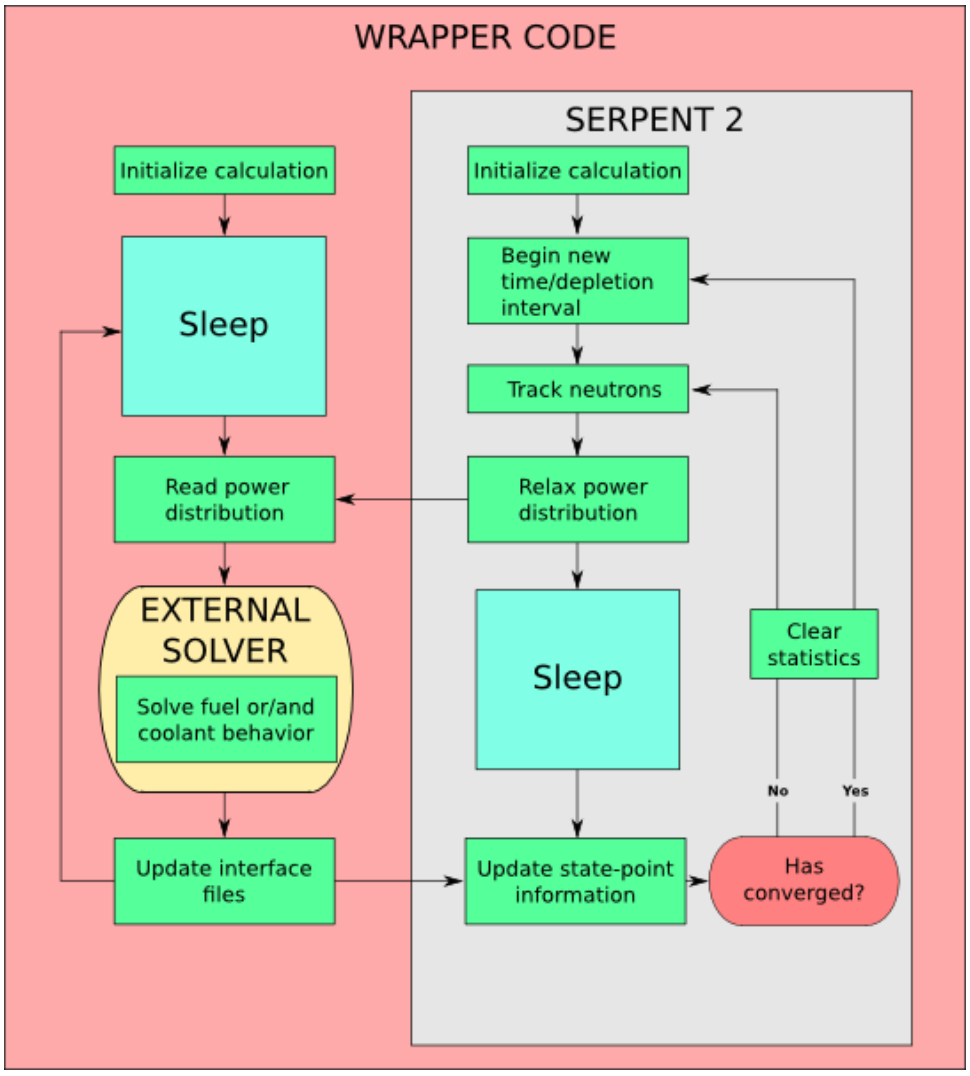


Figure 3.3.3. Serpent 2 basic external coupling methodology [31]

3.4.Spatial Mesh Overlay

Choosing the proper spatial mesh overlay for transients is an important and challenging task. It is a balance between accuracy and computational cost. In the terms of coupling, it is also necessary to ensure that a large amount of accuracy is not lost between the simulations and that a large amount of numerical error is not introduced by the coupling . This becomes especially challenging when considering the coupling of a CFD code, such as STAR-CCM+, and a Monte Carlo neutron transport code, such as Serpent 2. The main reason for this is the vastly different requirements for the meshing of the simulation.

In the terms of a CFD code, meshing is usually extremely fine, to ensure an accurate solution of the required physics equations. This is especially true when simulating shockwave propagation, where the short timeframe of the phenomena leads to a fine mesh resulting in an increased computational cost as described by Balderrama [7]. To ensure that the equations are being solved accurately within a CFD code performing a grid convergence index (GCI) study is recommended in the ASME V&V 20-2009 [29]. The purpose of a GCI is to quantify the error introduced in the selected mesh size, help in determining an adequate size of a mesh, and ultimately to demonstrate that the solution is independent of the mesh size. In terms of the STAR-CCM+ simulation of the HENRI cartridge, a GCI study was performed by Balderrama [7]. As no change was made to the mesh size presented by Balderrama, the results of the GCI and the subsequent mesh were accepted for use in this coupling.

A brief overview of the mesh of the STAR-CCM+ simulation is discussed here. The base mesh size is a 3 mm polyhedral mesh. The mesh is further reduced to 30% of the base size in the intermediate section and 60% of the base size in the test section. The final mesh refinement of the STAR-CCM+ simulation is shown here.

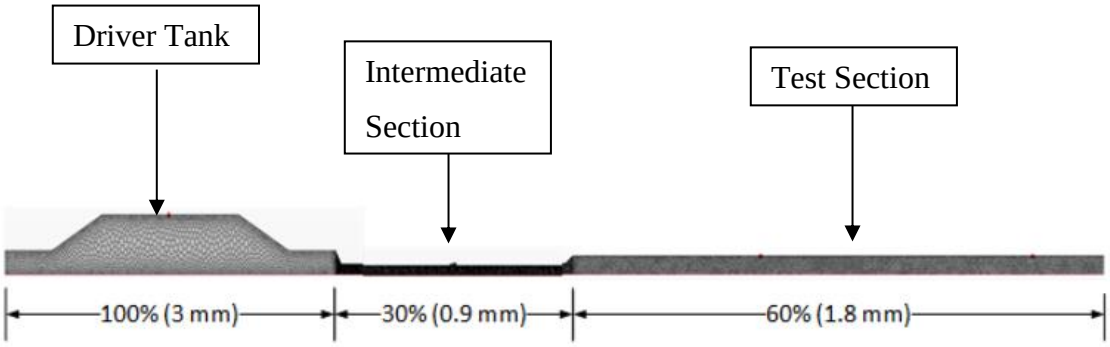


Figure 3.4.1. Final mesh refinement of HENRI facility

As can be seen in Figure 3.4.1 the mesh for the STAR-CCM+ simulation is extremely fine. This is necessary but computationally intensive and, when considering coupling, can lead to excessively long execution times. When considering the spatial overlay of a Monte-Carlo neutron transport code, such as Serpent 2, a different challenge arises. Serpent 2 has no defined mesh and does not solve simulations using any user defined mesh. When importing data into Serpent 2 from STAR-CCM+ the problem of spatial mesh overlay becomes simplified. This is due to the previously

mentioned universal multi-physics interface, in which Serpent 2 will create an initial mesh for importing physics data that corresponds to the mesh data given in the input.

The Serpent 2 model is 3-D with the geometry defined in the x, y, and z direction. The STAR-CCM+ simulation is that of HENRI using a 2-D mesh. Due to this difference the data from STAR-CCM+ must be manipulated. In this case the x-direction values in STAR-CCM+ would correspond to the z-directional values in Serpent 2. The y-direction values would correspond to either the x or y values within Serpent 2.

It would be more convenient to just deal with z-directional values within STAR-CCM+ as it would simplify the passing of data. In order to justify only using z-directional values a standalone STAR-CCM+ simulation was run. In this simulation the radial density, halfway through the test section of a HENRI cartridge was tracked. This was done to ascertain whether the density of helium-3 was changing drastically along the radial length of the HENRI cartridge. Figure 3.4.2 shows the results of the simulation.

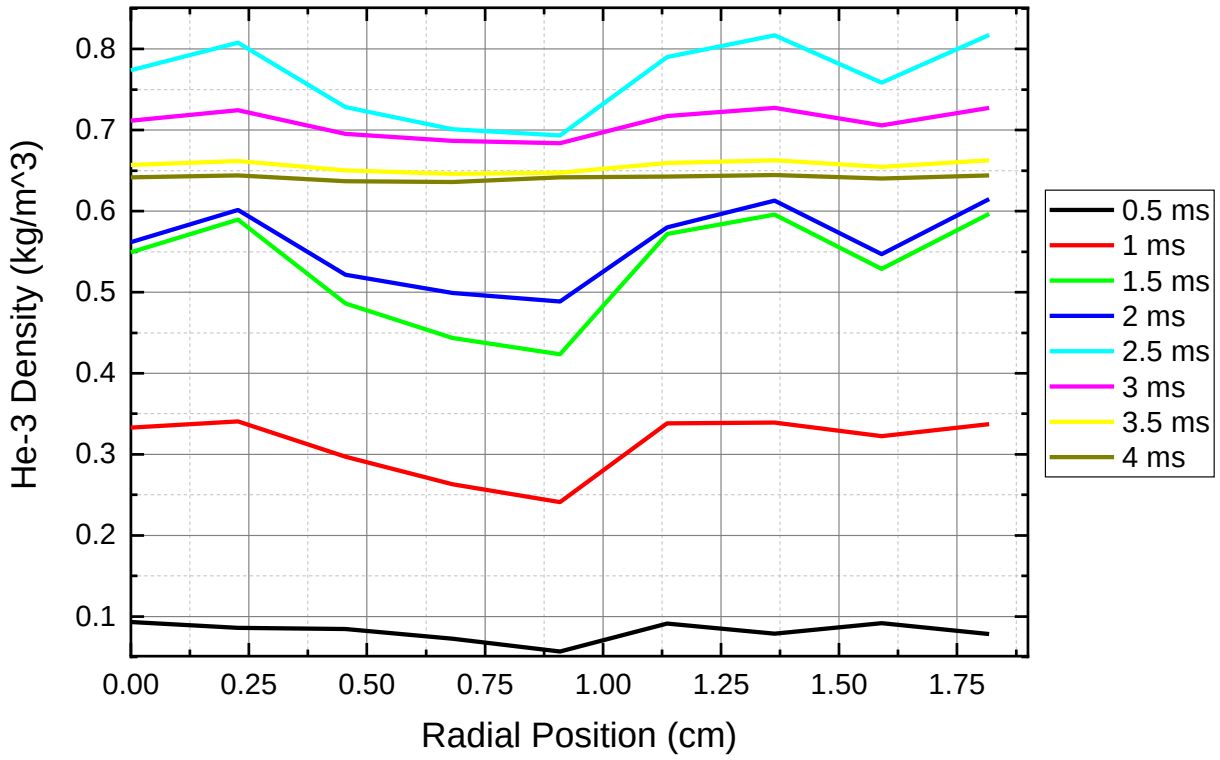


Figure 3.4.2. Time evolution of radial density profile of HENRI

While some variation is seen early in the transient with a max difference of 39% at 0.5 milliseconds, which is a result of the initial shock wave traveling through HENRI, as time progresses the difference in radial density reduces drastically with a minimal difference of 1% seen at 4 milliseconds. The average radial difference through the first four milliseconds of simulation is 18% The results of this simulation give credence to only using z-directional data when passing the relevant data to Serpent 2.

The spatial mesh overlay becomes more challenging when considering the importing of data into STAR-CCM+ from Serpent 2. In this scenario the data from Serpent 2 must be first converted into some sort of mesh that will then be read by STAR-CCM+. Serpent 2 is capable of outputting data along user-defined cartesian meshes using a detector input as described previously. An illustration of this is shown in Figure 3.4.3 The top of the figure shows the base STAR-CCM+ mesh and the Serpent 2 detector input, and the bottom shows the Serpent 2 mesh overlayed on top of the STAR-CCM+ mesh.

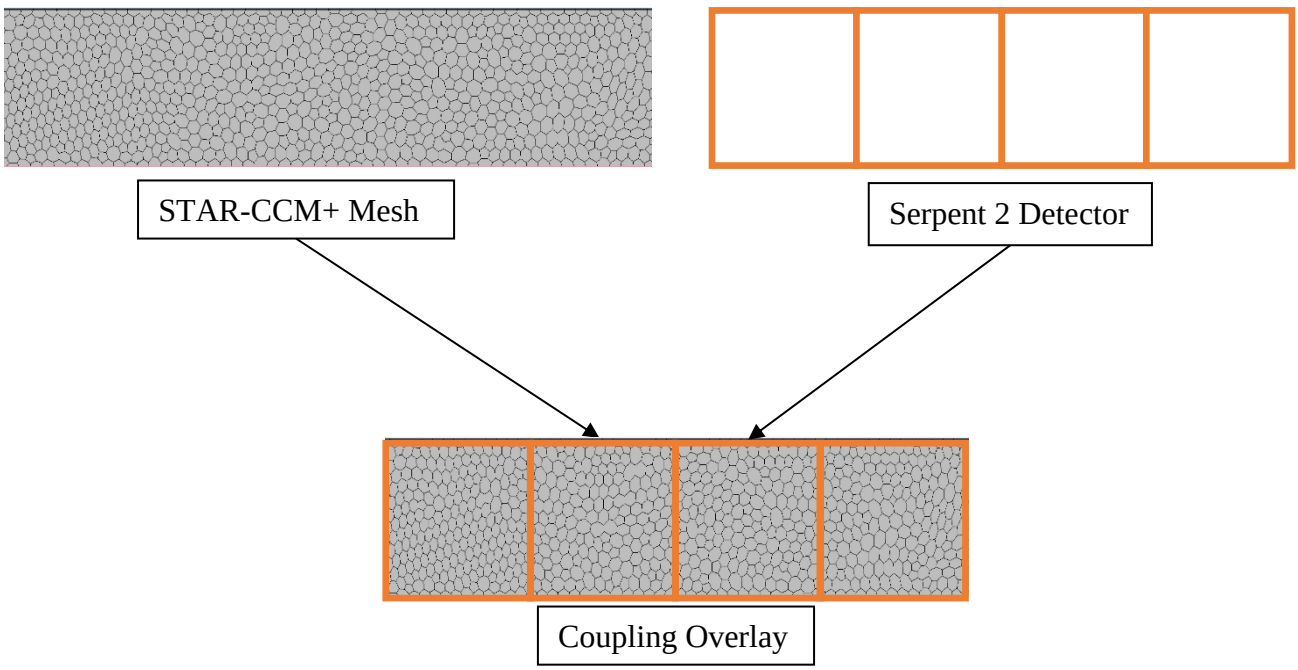


Figure 3.4.3. Serpent 2 and STAR-CCM+ overlay

This leads to the situation where any mesh created by Serpent 2 to output data is incapable of directly matching the polyhedral mesh used in STAR-CCM+. As there is no direct solution to this issue available within either of the codes at the moment, a certain amount of error is expected and

will need to be quantified. The quantification of this error can be found in the Coupling Verification section.

3.5.Coupled Time-Step Algorithm

A coupled time-step algorithm refers to the process in which the time-step for each code is decided upon. This is important as the demands of each code can lead to different time-step lengths being necessary to preserve the corresponding physics and accuracy of the solution. In the case of STAR-CCM+, Balderrama [7] performed a time-step sensitivity that determined that a variable timestep between 1E-7 seconds and 5E-8 seconds was appropriate to provide an accurate simulation.

In the case of a reactor physics code, Serpent 2, the selection of a time-step in transient simulations is dominated by the spatial resolution obtainable. This is highly variable depending on the complexity of the problem as well as the number of neutron batches being simulated. In general, the smaller the time-step the more neutron batches that must be simulated. The reason being that the smaller the time-step the less overall neutron interactions will occur reducing spatial resolution. The simulation of more neutron batches increases the computational cost of the simulation significantly. To balance between a refined spatial resolution and avoidance of excessive computational cost, a time-step of 2.E-6 seconds was chosen for the Serpent 2 calculation. This time-step was chosen based on the Courant number of the coupled mesh. The Courant number is a dimensionless value that represents how long a particle within a simulation stays within one cell of a mesh. The Courant number should always be below one, with it ideally being lower. If the Courant number is above one a particle can completely pass by a mesh cell in one time-step causing instability in the solution. The Courant number is calculated using the following equation:

$$C = \mu \frac{\Delta t}{\Delta \mathrm{x}} \qquad \text{Eq.(10)}$$

Where μ is the velocity of the particle, Δt is the chosen time-step of the simulation, and Δx is the mesh size. From Figure 3.1.1 the velocity (μ) of the helium-3 gas in the STAR-CCM+ simulation is calculated to be 3000 m/s, with a coupled time step (Δt) of 2E-6 seconds, and from Table 3.8.1 the size of the coupled mesh (Δx) is calculated at 1.21 cm. Using these values, the expected Courant number for the coupled mesh is calculated as 0.49. This puts the Courant number well below 1.0 as is desired.

From this Courant number it can be said that this time step, while larger than that of the STAR-CCM+ simulation is sufficient enough to ensure that the shock wave propagation of the helium-3 gas within a HENRI cartridge seen in the STAR-CCM+ simulation is still captured in the coupled mesh.

The two time-steps chosen for this study result in a coupled time-step algorithm of 40/1. This means that for every 40 timesteps simulated in STAR-CCM+, 1 timestep of Serpent 2 is performed. A variety of other ratios were investigated, ranging from 1 to 1 to 100 to 1, but as mentioned above 40 to 1 was found to be the best balance between computational cost and error propagation. The impact this has on the simulation is discussed in the Coupling Verification section.

3.6.Coupling Numerics

Coupling Numerics, also referred to as temporal coupling, determines how and when the data from each code is passed to the other. As presented in Section 2.1, this can be achieved in three ways, explicit, semi-implicit, and implicit.

When choosing the temporal coupling for the code outlined in this study the explicit method is chosen. The reasoning is two-fold, the inherent limitations of the codes chosen, and the type of calculation being pursued. STAR-CCM+ is a commercially available code with no access to the source code. This limits the capabilities of temporal coupling as no direct access to the timestep solving process is available. When considering previous works done with coupling STAR-CCM+, only examples of explicit temporal coupling can be found [10].

When operating Serpent 2 in the coupling mode described previously in Section 3.3.2, the code only outputs data after each time-step has been completed. This limits the available temporal coupling of Serpent 2 to explicit or semi-implicit. As STAR-CCM+ is limited to explicit temporal coupling, that is chosen for the coupling code.

When implementing explicit coupling the fact that is not the most robust coupling available must be addressed. In general terms, the coupling presented is referred to as a weak coupling. Weak coupling should only be implemented in scenarios where the driving physics of the system is not changing drastically between each time-step. For the case of injecting helium-3 into a nuclear reactor, this is the case. The injection of the gas will not have a pronounced effect on the driving

physics. The driving physics of TREAT being the fuel, coolant, and control rod thermophysical properties.

The main effect of the injection of helium-3 will be on the criticality of the reactor. Since, the main driving physics are not changing drastically between time-steps the use of a weak coupling algorithm is justified.

3.7. Coupling Algorithm

To implement the external coupling of STAR-CCM+ and Serpent 2 a wrapping code was developed. The wrapping code, which was named CONSTELATION, is a Python script which executes the STAR-CCM+ and Serpent 2 simulations sequentially. The CONSTELATION script performs a few necessary functions. At the start of the simulation, it creates the initial multi-physic interface files required by Serpent 2. After writing these files it sends an execution signal, specifically a SIGUSR1 signal, to Serpent 2.

Once one time-step is completed by Serpent 2, a set of output files is created. The output created by Serpent 2 is a MATLAB file containing the volumetric heating of the helium-3 gas. CONSTELATION then manipulates this output file into an input file for STAR-CCM+. The input file for STAR-CCM+ is a csv file containing the volumetric heating calculated by Serpent 2. Once this input file is created, STAR-CCM+ is executed. The output created by STAR-CCM+ is a csv file containing the temperature and density distribution of the helium-3 gas. CONSTELATION manipulates this file into a Serpent 2 input file. The input file for Serpent 2 is an updated multi-physics interface file with the new density and temperature distribution calculated by STAR-CCM+. This process is repeated until the end of the allotted simulation time is reached. The simplified algorithm of CONSTELATION is depicted in Figure 3.7.1.

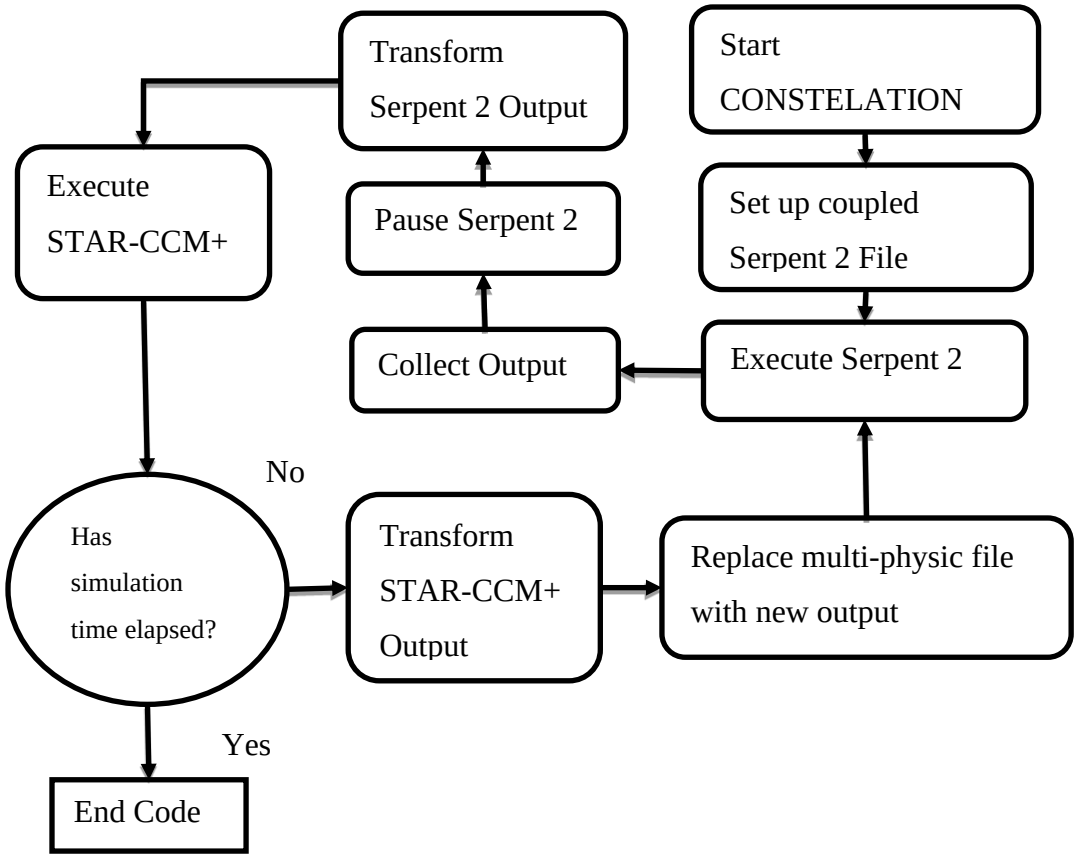


Figure 3.7.1. CONSTELATION coupling algorithm

Prior to the coupled calculation, a standalone STAR-CCM+ calculation is performed for one timestep to obtain an initial density and temperature distribution for HENRI. In addition to the CONSTELATION wrapping code, two JavaScript MACROs are used. These MACROs control the input and output reading of the csv files for each STAR-CCM+ simulation and create text files at the end of each coupled time-step. The text files are used by CONSTELATION to control the serial execution of the codes. Appendix B contains the CONSTELATION script in full.

3.8. Serpent 2 Base Model

To perform the coupling described in this study a Serpent 2 input deck modeling TREAT is used. The implementation of this model is done in seven steps.

1. Obtaining TREAT model
2. Implementation of HENRI cartridges
3. Coupling of helium-3 Properties

4. Creation of Source Files
5. Calculation of Volumetric Heating
6. Transient Simulation Inputs
7. Calculation of Fuel Temperature

3.1.1. Obtaining TREAT model

The TREAT model used for this study was provided by INL, it models the whole reactor facility. An overview of the model including discussion of modelling uncertainty can be found here [32]. The model uses the M8CAL half-slot core configuration which was the core configuration used by the TREAT facility before it was shut down in 1994. It is the current core configuration of the reactor. The M8CAL model consists of the following:

1. 318 fuel assemblies
2. 20 control rod assemblies
3. 13 zircaloy-clad dummy assemblies
4. 10 empty coolant channel regions

To mimic the core configuration expected at the start of a TREAT pulse transient experiment, the compensation rods were fully withdrawn, the control rods were positioned at 1.09728 meters (43.2 inches), and the transient rods were positioned at 0.48768 meters (19.2 inches). Figure 3.8.1 and Figure 3.8.2 show a top-down and side-view respectively of the M8CAL half-slot core configuration when run with Serpent 2.

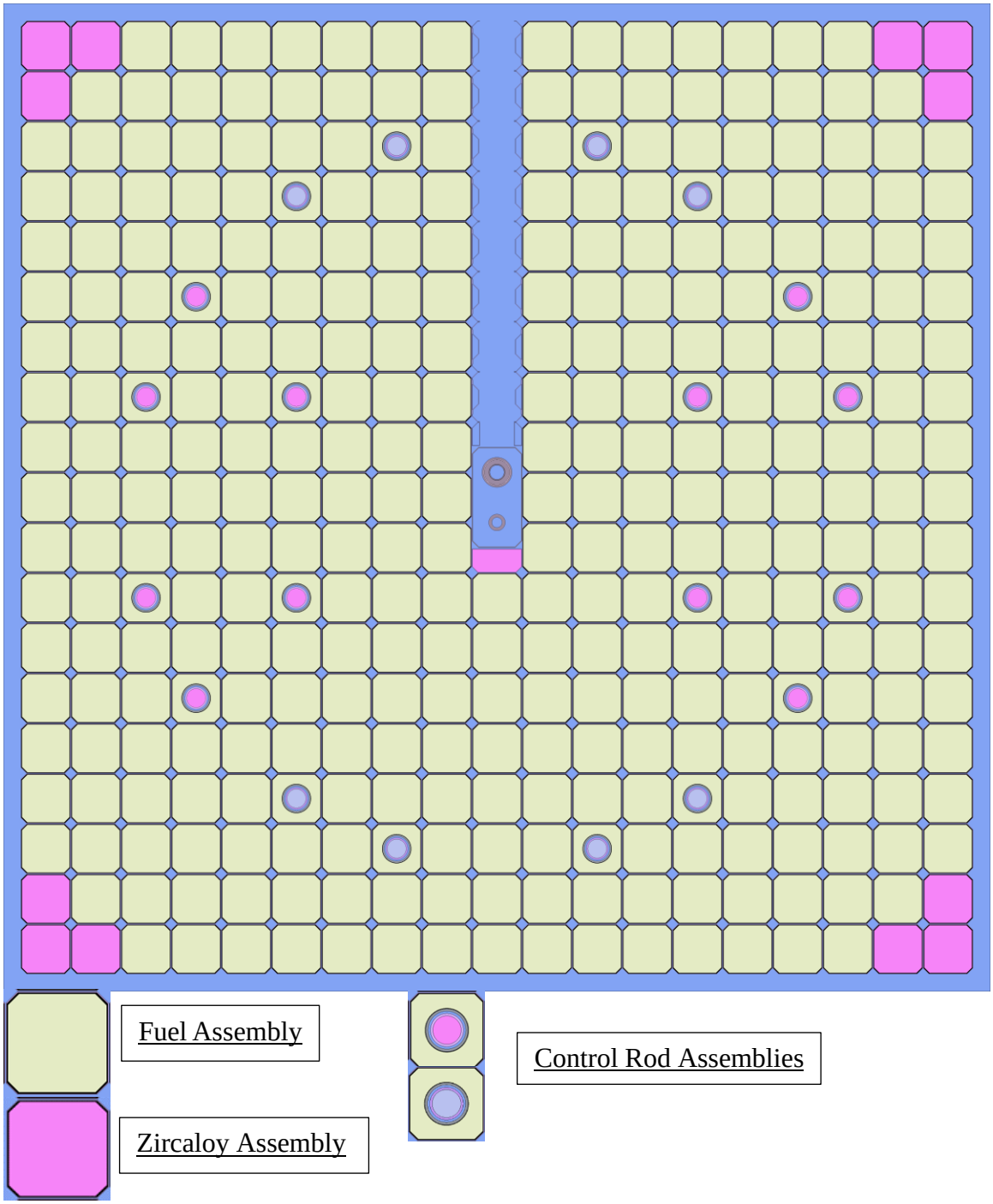


Figure 3.8.1 M8CAL half-slot core configuaration Serpent 2 model

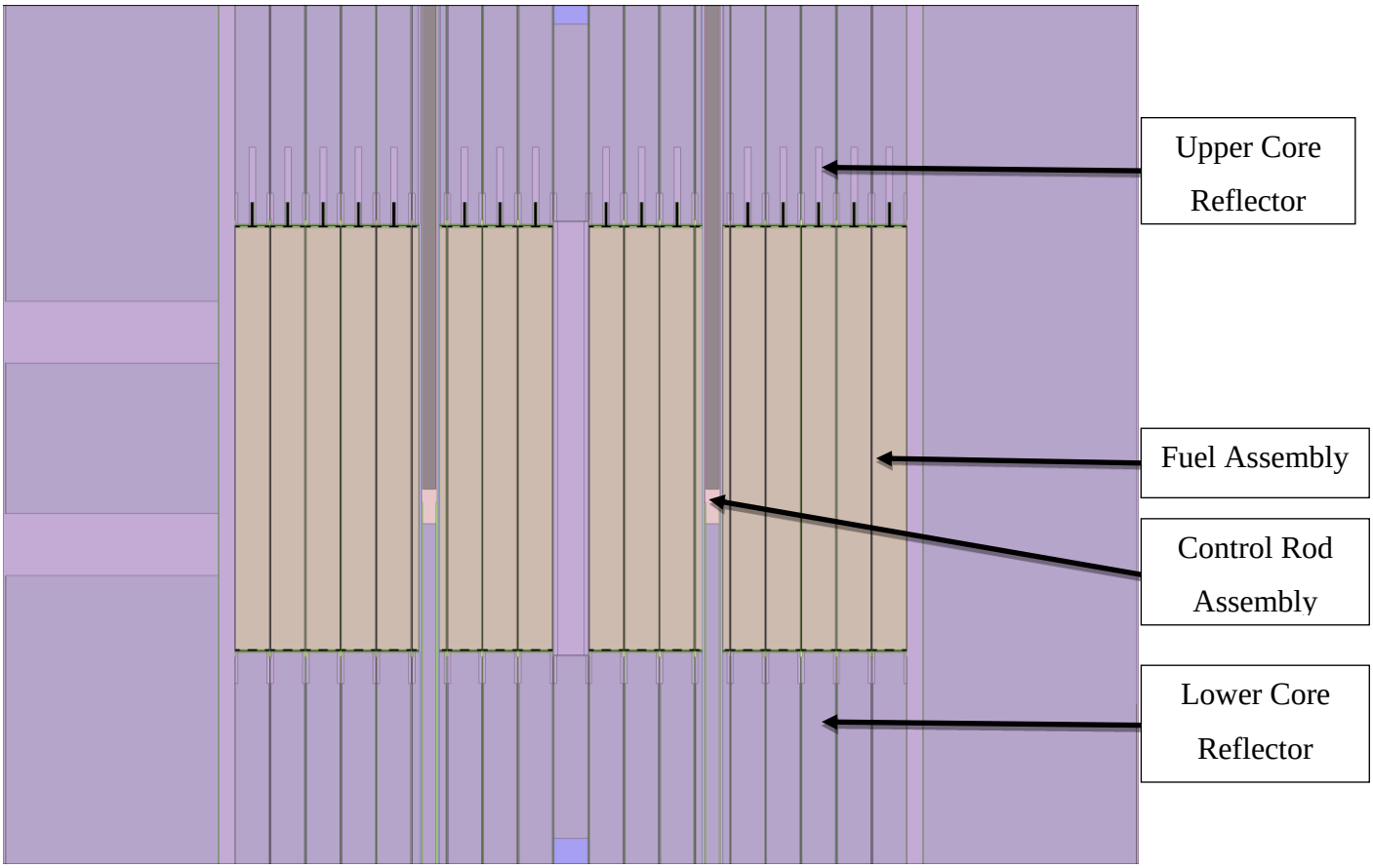


Figure 3.8.2 M8CAL half-slot core configuration side-view

When simulated with Serpent 2 in the critically simulation mode the k_{eff} obtained for this model is as follows:

$$k_{eff} = 1.00964 \ +/- \ 0.00209 \ [1.00555 \ \ 1.01373]$$

*Calculation performed using ENDF/B-7.1 nuclear data

3.8.1 Implementation of HENRI Cartridges

Specific inputs for the modeling of the HENRI cartridges in TREAT were created. Each HENRI cartridge is expected to be placed within an active fuel assembly. To model this, a circular region in which the helium-3 is simulated was created. An inner radius of 2.045 cm was surrounded by zirconium cladding with an outer radius of 2.500 cm. A small air gap between the outside of the zirconium cladding and the fuel assembly was also modeled. This cylinder was then placed at the

center of a standard fuel assembly. Figure 3.8.3 shows the implementation of HENRI within a standard fuel assembly.

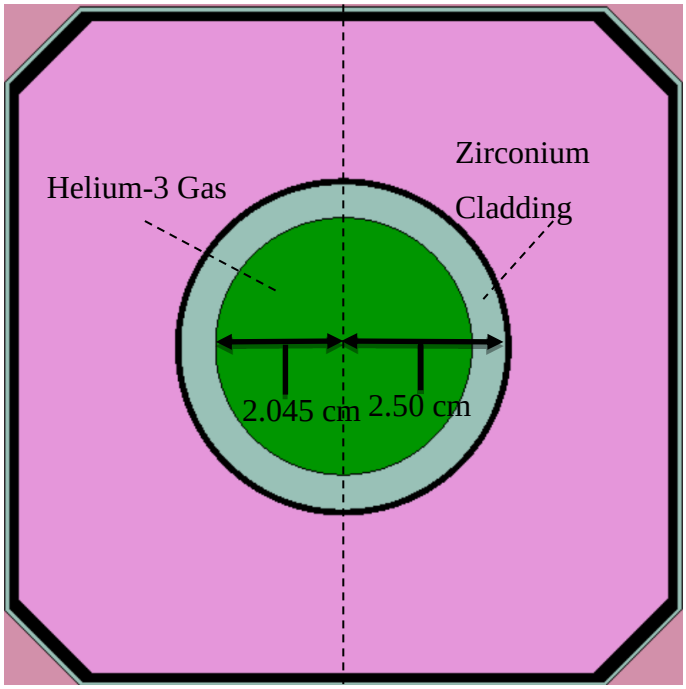


Figure 3.8.3. HENRI fuel assembly

It is expected that four (4) HENRI cartridges will be active within the TREAT core at one time. The placement of each HENRI facility is not finalized but an approximate location is known. Figure 3.8.4 shows a top-down view of the M8CAL half-slot core configuration with the implementation of 4 HENRI cartridges. The HENRI cartridges are the fuel assemblies with the green and blue regions at their center. Figure 3.8.4 also shows a side view taken of the M8CAL reactor at the red-dotted line annotated on the top-down view of the HENRI-TREAT model.

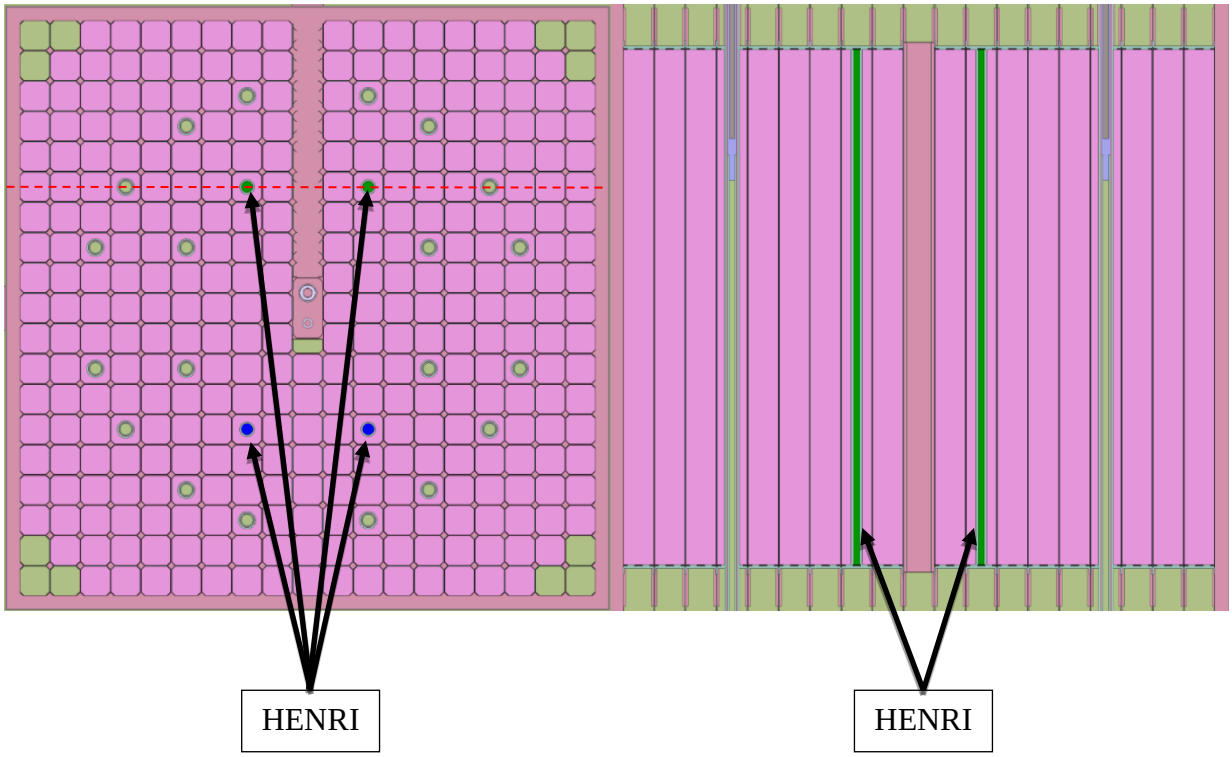


Figure 3.8.4. Location of HENRI cartridges in TREAT

The test section of HENRI is expected to be close to a vacuum before the injection of helium-3. To execute Serpent 2 in transient mode, the HENRI cartridges cannot be initially modeled as a vacuum. This is because Serpent 2 when being executed in transient mode only modifies the density and temperature of a material already defined in the geometry. It cannot add or replace material definitions within the geometry. Therefore, an initial density of helium-3 must be defined.

To simulate this and satisfy the conditions of Serpent 2's transient mode a small overall density of helium-3 is simulated within each HENRI cartridge. The value chosen for this initial amount of helium-3 is 1E-6 g/cm^3, which corresponds to less than 6894.76 Pa (1 psi) of pressure within the HENRI cartridge. This amount is chosen for two reasons. The first being that Serpent 2 does not allow a material to be defined as having zero mass density. The second being that 1E-6 g/cm^3 is the lowest possible density that can be output by the HENRI STAR-CCM+ model while it is executing.

To ensure that this small amount of helium-3 has a minimal to no effect on the criticality of the model a criticality simulation was executed. The k_{eff} obtained is as follows:

$$k_{eff} = 1.00401 \ +/- \ 0.00031 \ [1.00341 \ 1.00461]$$

The small initial amount of helium-3 has minimal effect on the criticality of the Serpent 2 model and is therefore used. The M8CAL half-slot model with the implementation of four HENRI facilities is referred to as the HENRI/TREAT model from now on.

3.8.2 Coupling of Helium-3 Properties

To simulate the density time evolution of helium-3 using the coupled transient mode of Serpent 2 a decision of symmetry must be made. When using the multi-physics interface input described in the Serpent 2 External Coupling section a distribution of material density and temperature is defined. In the coupling algorithm the distribution defined by the multi-physics interface is inherently linked to a single STAR-CCM+ simulation.

Therefore, it is possible to use one STAR-CCM+ simulation to dictate the time-density evolution of helium-3 for all four HENRI cartridges. This would be ideal as it would significantly reduce the computational cost of the coupled code. To justify this simplification, the neutron flux across all four HENRI facilities would need to be symmetrical as it would result in symmetrical volumetric heating.

When executing the HENRI TREAT model an a-symmetry in expected neutron flux was observed. Figure 3.8.5 shows a detector mesh plot of total neutron interactions when executing the HENRI TREAT model in criticality mode on the left side of the figure. On the right side of the figure is the plot of total neutron interactions recorded for the two HENRIs located in the lower flux section, and the two HENRIs located in the higher flux section.

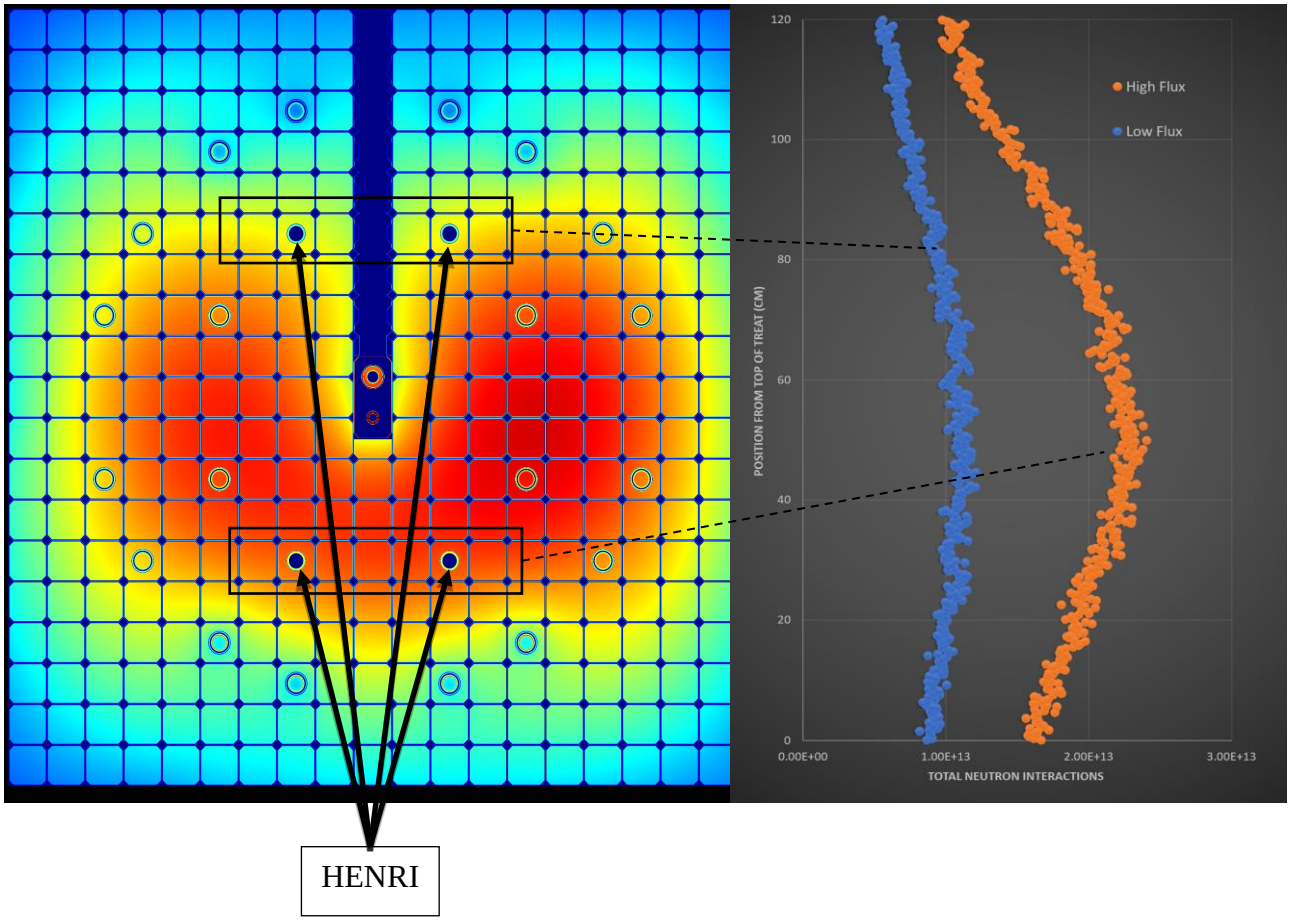


Figure 3.8.5. Neutron interactions in HENRI TREAT model

As can be seen from this model, there is essentially a half-core symmetry seen by the four HENRI cartridges. The HENRIs located at the top of Figure 3.8.5 see a smaller flux than the ones located at the bottom. This has the possibility of creating asymmetrical results for the density time evolution. As a smaller neutron flux would result in a smaller amount of volumetric heating. This would result in a lower temperature increase within the helium-3 gas as it is being injected into TREAT.

Therefore, to account for this, two multi-physic interface input files are used. This corresponds to two separate STAR-CCM+ simulations being executed simultaneously to account for the a-symmetry of the neutron flux seen in the HENRI TREAT model. One STAR-CCM+ simulation will simulate the density time evolution of helium-3 for the low flux part of the reactor,

denominated 1st Group. The other STAR-CMM+ simulation will simulate the density time evolution of helium-3 for higher flux side of the reactor, denominated 2nd Group. This grouping of HENRIs is visualized in the following mesh plot.

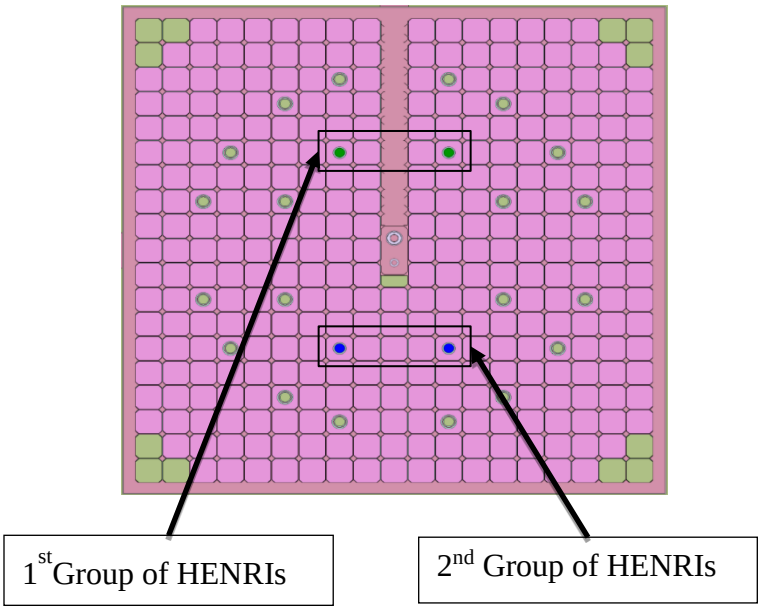


Figure 3.8.6. Grouping of HENRIs

3.8.3 Creation of Serpent 2 Source Files

In order to execute Serpent 2 in transient mode, a series of source files must be created by first executing the code in source generation mode. When executing the code in source generation mode a series of files are created that randomly save the position and energy of the simulated neutrons. This is performed to get an accurate representation of the initial neutron distribution within the simulated geometry. This information is then used by Serpent 2 to initialize a transient simulation.

Little modification to the base model described in the Implementation of HENRI Cartridges section is required. A single line must be added:

```
set savesrc FILEPATH [PN PP NX NY NZ]
```

Once the base model has been modified to run in source generation mode it can be executed. Special care must be taken to ensure that the source files created accurately describe the physics occurring.

In Serpent 2 the probability of storing a "live" neutron i.e., a neutron that is actively moving and interacting within the problem, is inversely dependent on the cross section used to sample its path. This can lead to an interaction probability greater than one. This is non-physical and will result in a warning. The developers of Serpent 2 recommend when this occurs to adjust the PN value, which dictates the probability of storing a neutron in the source file. They also recommend halving the maximum mean free path used by Serpent 2, this effectively halves the distance that the majority of the neutrons will travel through the problem, decreasing the chance that an interaction probability greater than one occurs. Both solutions are utilized for the generation of the HENRI TREAT model source files. The PN value for the generation of source files is set to 0.09 and the maximum mean free path is halved. These values were found through an iterative process, in which various PN and mean free path values were simulated. When the base model is executed with these changes no warnings are generated.

In addition to ensuring that an interaction probability greater than one is avoided when generating the source files, the power at which the reactor is operating must be simulated. In this first coupling, TREAT is assumed to be in a steady state at the assumed peak pulse power. A simulation of a TREAT pulse experiment is the eventual end-goal of the coupling but falls outside the current scope of this study.

Serpent 2 allows the total fission power of a simulation to be dictated using the following input:

```
set power P [ MAT ]
```

Where *P* is the fission power in Watts and *MAT* is the material. If *MAT* is omitted, then *P* is taken as the total fission power of the system. TREAT is capable of pulses ranging from 8,000 to 16,000 MW. A higher pulse power would result in a higher overall flux and thus more volumetric heating within the helium-3 gas. The more volumetric heating within the helium-3 gas the higher the temperature of the gas. Therefore, a 16,000 MW pulse is conservatively assumed for this coupling.

A final caveat of source generation using Serpent 2 is that the system must be as close to criticality as possible. A system not close to criticality would result in errors in the live neutron source. This is a result of the k-eigenvalue iterator utilized by Serpent 2 only being accurate for critical solutions. The HENRI-TREAT model was executed in source generation mode with a normalization power of 16,000 MW with the following results:

$$k_{eff} = 1.01055 \pm 0.00027 \ [1.01003 \ \ 1.01107]$$

As the system is close to critical it can be used in the generation of source files.

3.8.4 Volumetric Heating Calculation

One goal of the Serpent 2 code within the coupling is to calculate the total volumetric heating occurring within the HENRI cartridge. When the helium-3 gas is injected into TREAT it will encounter a neutron flux. This flux is expected to be on the order of 10^{16} neutrons/cm^2-s. When this happens the helium-3 gas will capture many of these neutrons and undergo fission. The two major capture reactions that helium-3 is capable of are the (n, γ) and (n, p), where the helium-3 atom will absorb a neutron and eject a gamma ray and proton, respectively. Based on the cross section of these reactions, the (n, p) reaction is expected to be the reaction that occurs most often as it has a much higher (5300 barns) cross section than the (n, γ) reaction (0.05 milibarns).

The 0.764 MeV energy of the (n, p) reaction is expected to deposit its energy locally in the helium-3 gas. The reason being that the release of the proton will contain the majority of the 0.764 MeV, and the proton being a heavy charged particle will interact with the surrounding local gas. This energy deposition is expected to result in a local temperature increase of the helium-3 gas.

To calculate the expected volumetric heating, a Serpent 2 detector was used. The detector response function was set to the -4 option. This option is specific to Serpent 2 and calculates the total heat production within a detector volume. It does this by using KERMA data for both neutrons and photons to calculate the total heat production occurring within the detector volume. It then multiples this by the material density to get the overall heat production within the defined detector volume. When running Serpent 2 in steady state mode the output of this detector response is in Watts. Serpent 2 divides all results by detector volume, but unless specified using the *dv* option, this is always taken to be 1.

As mentioned previously, when executing Serpent 2 in transient mode all detector results are output in a time-integrated format, therefore the detector response output would be in Joules. To calculate energy deposition within a HENRI cartridge the following detector card inputs outlined in Table 3.8.1 are used

Table 3.8.1 - Volumetric heating detector inputs

Card	Input	Description
Det		
NAME	Serpent 2STAR	Sets name of detector.
Dr		
MT	-4	Defines the response function number.
MAT	Void	Setting the MAT card to void tells Serpent 2 to use the material at the collision point.
Dm		
	He3	Integrates the detector over the defined material. In this case the helium-3 gas. This ensures only deposition within helium-3 is counted.
Dv		
VOL	3.97	The volume of each detector mesh in cm^3.
Dx		
XMIN	18.5	Sets the minimum x-component of the detector.
XMAX	20.405	Sets the maximum x-component of the detector
NX	1	Sets the number of x-bins.
Dy		
YMIN	38.5	Sets the minimum y-component of the detector.
YMAX	40.405	Sets the maximum y-component of the detector.
NY	1	Sets the number of y-bins.
dz		
ZMIN	-60.48375	Sets the minimum z-component of the detector.
ZMAX	60.48375	Sets the maximum z-component of the detector.
NZ	100	Sets the number of z-bins.

The detector volume was calculated by taking the volume of ¼ of the HENRI cartridge in cm^3 and then dividing by the total number of detectors meshes. This was done since the STAR-CCM+ simulation simulates a symmetrical quarter slice of the HENRI cartridge.

The following equations show how the detector volume is calculated, where the radius of HENRI is taken to be 2.045 cm and the length of the test section is the total active region of TREAT, which is 121 cm:

$$V_{HENRI} = \pi * 2.045^2 * 121 = 1589.72\ cm^3 \qquad \text{Eq.(11)}$$

$$V_{\frac{1}{4}Henri} = \frac{V_{HENRI}}{4} = 397.43\ cm^3 \qquad \text{Eq.(12)}$$

$$V_{Detector} = \frac{V_{\frac{1}{4}Henri}}{100 * 1 * 1} = 3.97\ cm^3 \qquad \text{Eq.(13)}$$

The *dx* input defines the x-mesh where the reactions are scored. The *dy* input defines the y-mesh where the reactions are scored. The *dz* input defines the z-mesh where the reactions are scored. The meshes created are of equal size. The numbers shown in the *dx, dy*, and *dz*, cards represent the location of ¼ of the HENRI cartridge from the 1st Group located on the right of TREAT shown in Figure 3.8.4.

Since the STAR-CCM+ simulation uses a 2-D mesh, effort was made in order to output 2-D data from the 3-D data calculated by Serpent 2. This was done by manipulating the detector card so that it outputs a single slice of data in the x direction, due to the change of reference frame between the two codes, the x-direction in Serpent 2 is the z-direction in STAR-CCM+.

For determining the total number of mesh bins, it was desired to keep the size small enough that the change in volumetric heating caused by the gas propagation would be captured. It is also desired to be large enough that statistical convergence is reached by Serpent 2. As such a 100 mesh bins were chosen. This divides the detector results calculated by Serpent 2 axially along the HENRI cartridge and 100 bins in the total axial length of TREAT, which is 121 cm, results in each mesh bin having a length of 1.21 cm. As the *dx* option is set to 1, the *dy* option divides the results radially. The resolution of this radial division is a subject of this study as it can at least partially account for any self-shielding of the helium-3 gas.

The self-shielding of helium-3 is an important phenomenon to consider when developing the detector inputs. As the capture cross section of helium-3 is large (5300 barn) it will readily absorb neutrons. This results in many capture reactions happening on the periphery of HENRI with not as many occurring as you move to the radial center. The amount of self-shielding seen by the helium-3 gas and the effect the radial resolution of the detector has on capturing this effect is discussed in the Results section.

3.8.5 Serpent 2 Transient Inputs

To run Serpent 2 in transient mode small modifications must be made to the source generation inputs. These inputs include the following

1. Time binning of the solution
2. Total simulated population of neutrons
3. A previously generated set of source files

The time binning of the simulation is dictated by the following input:

tme NAME NUM NB T_{min} T_{max}

Where NAME is a user defined name of the time binning. The NUM dictates that the binning type (1 = arbitrary, 2 = uniform, 3 = log-uniform), 2 is used for this transient input. NB is the number of total bins to be simulated. Tmin and Tmax dictate the initial time of the simulation and the final time of the simulation, respectively. When using the uniform time binning the total number of timesteps simulated by Serpent 2 is dictated by taking Tmax and diving by NB.

The total simulated populations of neutrons that are simulated is dictated by the following input:

set nps *PP* [*BTCH TBI*]

Where PP is the total number of particles simulated. In transient simulations this refers to the total number of particles simulated per time interval. BTCH is the number of batches that the total population is split into, dividing PP by BTCH gives the number of particles simulated per batch. TBI determines the time binning structure for transient simulations. Table 3.8.2 details the inputs used for the TME and NPS cards used for the Serpent 2 transient simulation.

Table 3.8.2 - Serpent 2 transient inputs

Card	Input	Description
Tme		
NAME	Tsim	Sets name of time binning to be called by other cards.
NUM	2	Sets time binning structure to uniform. This is done to ensure that the time-steps are equal in length throughout the simulation.
NB	2000	Sets the number of time batches being simulated. This determines the overall time-step of the solution. The time-step is calculated by taking Tmax and diving by NB.
Tmin	0	Sets initial time of simulation to 0 secs.
Tmax	1E-2	Sets max time of simulation to 10 milliseconds. This is the max time at which HENRI is expected to be most impactful.
Nps		
PP	200000000	Sets the total number of particles simulated per timestep. A large number of neutrons is needed to be simulated for complicated geometries such as that of the HENRI TREAT model.
BTCH	500	Sets the total number of batches simulated per timestep. A large number of batches is needed in order to ensure statistical variance is minimized.
TBI	Tsim	Sets the time binning to that of the time binning dictated by the **tme** card.

With a total source population of 200,000,000 neutrons over 500 batches. This corresponds to 400,000 source neutrons per batch.

3.8.6 Calculation of Fuel Temperature

As stated earlier, the power of the HENRI-TREAT model is set to 16,000 MW as this is the expected power of the reactor at the peak of a pulse. This leads to a non-inconsequential amount of energy being deposited into the reactor over the course of the transient, specifically in the graphite fuel that makes up a majority of TREAT.

The effect of this energy deposition can be accounted for by using the following equation:

$$\Delta Q = mC_p\Delta T \qquad \text{Eq.(14)}$$

Where ΔQ is the change in energy of the fuel, m is the mass of the fuel, C_p is the specific heat of the fuel, and ΔT is the change in temperature of the fuel. When considering this equation in a HENRI-TREAT transient, it can be re-arranged to solve for the temperature of the fuel after each timestep. The steps used to calculate the temperature of the fuel after each timestep is shown here.

First, the energy of the fuel at the beginning of the timestep is calculated

$$Q_B = T_B mC_p \qquad \text{Eq.(15)}$$

Where T_B is the temperature of the fuel at the beginning of the timestep.

Second, the amount of energy deposited in the fuel over each timestep is needed. This can be automatically calculated by Serpent 2 using a detector input similar to the one described in Section 3.8.4. The detector inputs are described in Table 3.8.3

Table 3.8.3 Fuel energy deposition inputs

Card	Input	Description
Det		
NAME	FuelDeposition	Sets name of detector
Dr		
MT	-8	Defines the response function number. -8 calculates the energy deposition within the fuel.
MAT	Void	Setting the MAT card to void tells Serpent 2 to use the material at the collision point.
Dm		
	Fuel1	Integrates the detector over the defined material. In this case the fuel. This ensures only deposition within the fuel is counted.
Dx		
XMIN	-96.52	Sets the minimum x-component of the detector
XMAX	96.52	Sets the maximum x-component of the detector
NX	1	Sets the number of x-bins.
Dy		
YMIN	-96.52	Sets the minimum y-component of the detector
YMAX	96.52	Sets the maximum y-component of the detector
NY	1	Sets the number of y-bins.
dz		
ZMIN	-60.48375	Sets the minimum z-component of the detector
ZMAX	60.48375	Sets the maximum z-component of the detector
NZ	10	Sets the number of z-bins.

As the energy deposition in a nuclear reactor is expected to change axially, with higher energy deposition at the center and lower at the edges, the NZ value is set to 10 for this detector This detector outputs the energy deposited in the fuel in Joules. Since the results of the detector are divided axially, the mass of each node needs to be calculated in order to accurately account for the temperature change in the fuel.

The mass of fuel in each node is calculated using the following equations:

$$m_n = V_n * \rho \qquad \text{Eq.(16)}$$

Where V_n is the volume of each node and ρ is the density of the fuel. As the length of the transient is short, the density of the fuel is assumed to be constant. The volume of each node can be calculated by the following equation:

$$V_n = A_{FA} * A_{FC} * A_{FH} * H \qquad \text{Eq.(17)}$$

Where A_{FA} is the total area of the fully intact fuel assembles, A_{FC} is the area of fuel in the control rod assembles, A_{FH} is the area of fuel in the HENRI fuel assemblies, and H is the height of the node. When considering the TREAT model and an axial distribution of ten, the mass of fuel in each node is 642.41 kg. The density and specific heat of TREAT fuel is 1720 kg/m^3 and 998 J/kg-K [33].

With the mass of each node determined the temperature of the fuel at the end of the timestep can be calculated using the following equation:

$$T_E = (Q_b + dQ)/(m * C_p) \qquad \text{Eq.(18)}$$

Where T_E is the temperature of the fuel at the end of the timestep and dQ is the amount of energy calculated by the detector described in Table 3.8.3. This newly calculated temperature is then updated by CONSTELATION and passed to Serpent 2.

3.9.Coupling Verification

An important factor when considering the coupling of codes is verifying that the solution is accurate. There currently exists no published standard on verification and validation of coupled computational fluid dynamics and neutronic codes but previous work as outlined in Section 2.4 provides an approach to verify and validate coupled simulations. In general, most coupled simulations are validated by comparing the coupled code to previously validated coupled codes or with experimental data. However, as no current experimental data of HENRI within TREAT exists

for comparison and acquiring a separately validated coupled code is outside the scope of this study, a separate approach is used.

Verification of CONSTELATION by ensuring that no large amount of error is introduced while running the coupled simulation is performed. This is done by applying the principles of the conservation of mass and energy. It is imperative to show that the coupled simulation does not create or destroy a large amount of mass in the simulation, this can be quantified by tracking the mass error percentage of the simulation. The mass error is quantified as,

$$\varepsilon_m = \frac{m(t) - m_i}{m_i} \qquad \text{Eq.(19)}$$

where m_i is the initial mass of the simulation and $m(t)$ is the current mass of the simulation.

When considering the conservation of energy of the solution, the total amount of energy released by the helium-3 capture reaction calculated by the Serpent 2 simulation must be conserved by STAR-CCM+. This can be quantified by tracking the energy error of the simulation. The energy error is quantified as,

$$\varepsilon_E = Q_{STAR-CCM+} - Q_{SERPENT2} \qquad \text{Eq.(20)}$$

where $Q_{STAR-CCM+}$is the energy of the STAR-CCM+ simulation and $Q_{SERPENT2}$ is the energy added by the Serpent 2 calculation.

4. RESULTS

The objective of this study is to develop a coupled CFD and reactor physics model of the HENRI-TREAT system and to verify the said coupling. As no experimental data exists, a proper validation of the coupled code is outside the scope of this study. Nonetheless, to show that the coupling of these two codes is working, three studies were performed. The first study consists of verifying the communication between the coupled codes, STAR-CCM+ and Serpent 2. This study focuses on ensuring that the density and temperature calculated by STAR-CCM+ is being accurately implemented by Serpent 2 and that the volumetric heating calculated by Serpent 2 is being accurately implemented by STAR-CCM+. For the purpose of this study, a simplified version of the STAR-CCM+ HENRI model is used. The main simplification being that no heat transfer is modeled within the HENRI cartridge. This simplification ensures that the energy balance within the STAR-CCM+ model would only account for the energy being added to the simulation from what was read in by the Serpent 2 outputs.

The second study is that on the radial resolution of the HENRI TREAT model. Since self-shielding is expected to be a dominant phenomenon, an understanding of the depth at which this occurs is necessary. The depth at which the self-shielding is being simulated is entirely controlled by inputs into the HENRI TREAT model. Therefore, multiple simulations using CONSTELATION were performed in order to quantify the ideal radial resolution needed to fully capture the effect of self-shielding within the HENRI cartridge.

The third study outlined in this section shows the results of the CONSTELATION code when executed with the ideal radial resolution and the full HENRI TREAT configuration . The purpose of this is to demonstrate the relationships between the HENRI cartridges and TREAT during a pulse experiment and provide insights into the coupled multi-physics of the simulation as well as determine the feasibility of the stated 5% negative reactivity in 5 milliseconds goal of HENRI.

4.1.Coupling Verification

In order to verify the coupling of the HENRI STAR-CCM+ model to the HENRI TREAT model a verification simulation was performed. In this simulation the HENRI model was pressurized to 3.45 MPa (500 psi) and the test-section assumed to be at 873 K (600 ^{O}C) initially. Additionally,

heat transfer was not considered. This was done by setting the wall conditions within the STAR-CCM+ model to be adiabatic.

The initial temperature is a conservative assumption and corresponds to the max expected temperature of TREAT during a pulse experiment. 873 K is expected to occur much later in the transient, after HENRI has been activated, but it is considered here in order to bound the problem as a higher initial temperature in the test-section is expected to delay the pressurization of the HENRI cartridge. The number of radial points that Serpent 2 volumetric heating was passed with was set to 1. This was done to simplify the problem and simplify the process of extracting results from both simulations.

On the Serpent 2 side of the calculation, 200,000,000 neutrons were simulated per timestep in 500 batches. STAR-CCM+ was executed with 96 cores.

The mass error of the solution throughout the simulation is extremely small, with a max mass error of 1.6×10^{-8} % and an average mass error of 4.7×10^{-10} % .This leads to the conclusion that the STAR-CCM+ simulation is not creating or destroying mass in between time-steps.

In order to verify that Serpent 2 and STAR-CCM+ are correctly passing information to one another, the total wattage calculated per time-step was tracked throughout the simulation. As this is the figure of merit for the Serpent 2 simulation output, it is critical that any change in implementation of this value by STAR-CCM+ is understood and quantified. Figure 4.1.1 compares the total wattage calculated by Serpent 2 compared to the total wattage calculated by STAR-CCM+.

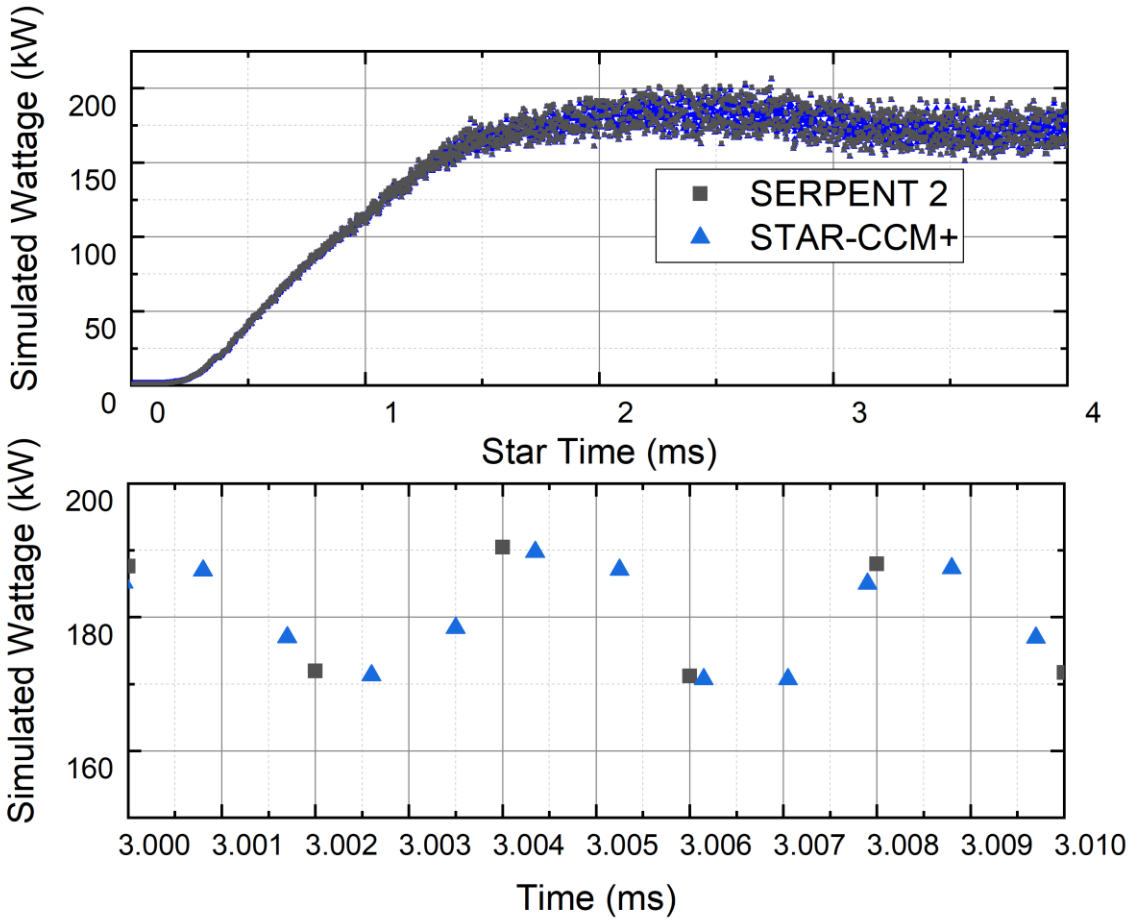


Figure 4.1.1. Serpent 2 vs STAR-CCM+ wattage for one HENRI cartridge

As can be seen from these plots, initially the amount of wattage being calculated by both codes stays converged. However, as the transient progresses the wattage being calculated begins to oscillate between each time-step as depicted in the bottom plot of Figure 4.1.1. The overall oscillation does not reach more than 20 kW amplitude, about 10% of the overall calculated wattage. To understand where these oscillations originate from individual values of wattage from Serpent 2 and STAR-CCM+ per time-step were compared and plotted in Figure 4.1.2. The goal being to quantify where the oscillations are occurring, either from one of the codes or from the coupling itself. The information passing time-step is defined as the time-step when the Serpent 2 output detailed in Section 3.7 is read and implemented by STAR-CCM+.

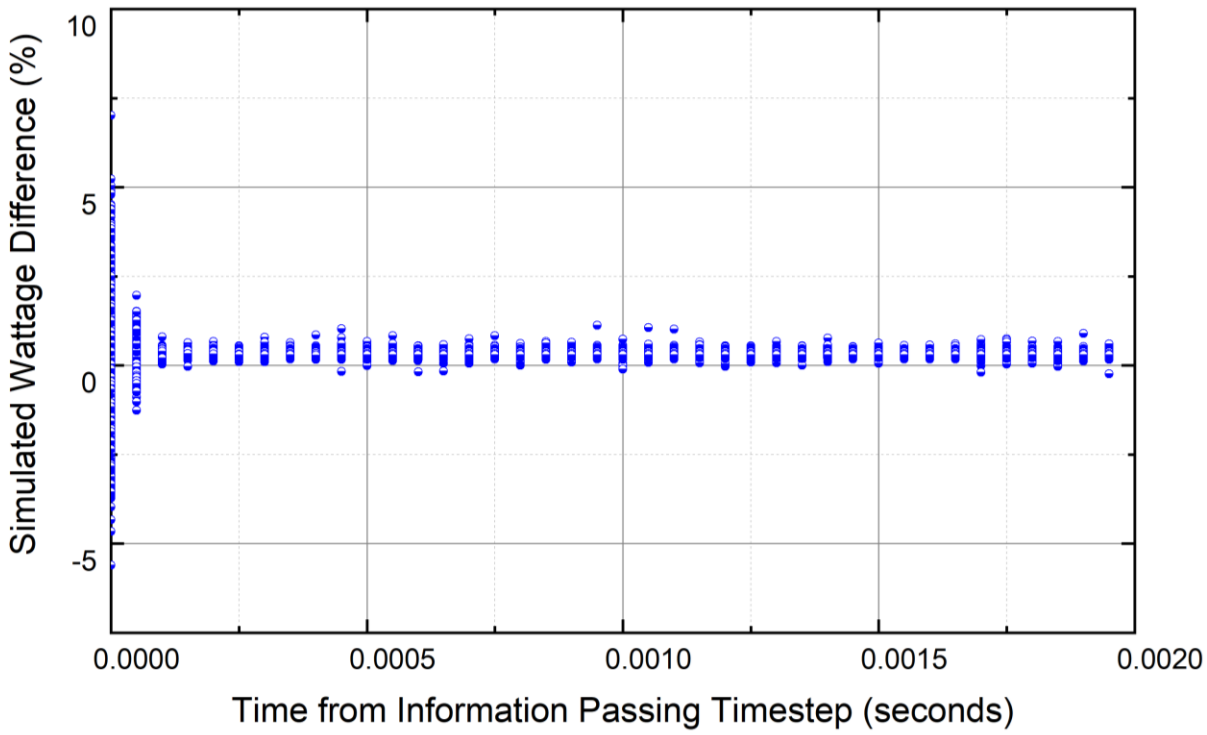


Figure 4.1.2. Serpent 2 vs STAR-CCM+ percent difference

From Figure 4.1.2 the total wattage simulated by both codes agrees considerably well. There is slight variation early on, but it converges to value below 1% error after two time-steps. The initial variation can be explained from the oscillations seen in Figure 4.1.1, when large oscillations occur between the previously implemented wattage value and the newly read in value STAR-CCM+ takes on average two time-steps to converge to the newly read in value.

Once STAR-CCM+ converges there is still a small amount of error between the value read in from Serpent 2 and what STAR-CCM+ is simulating. This can be explained by the spatial mesh overlay. As the spatial mesh overlay is not a one-to-one representation between both codes as seen in Figure 3.4.3 the total energy being simulated can be under or overestimated as STAR-CCM+ tries to apply the cartesian definition of Serpent 2 to the polyhedral mesh of its own simulation. Therefore, the error seen in this plot after STAR-CCM+ has converged is the “mesh error” of the coupling. The average mesh error seen with this calculation is 0.56% with 95% confidence.

No large amount of error exists between the implementation of values between STAR-CCM+ and SERPENT 2 therefore, the oscillations observed in Figure 4.1.1 can conclusively be shown to be originating from the Serpent 2 code itself. They are a result of what is known as flux “drift”. As a

Serpent 2 transient progresses and the transient conditions stray further and further from the initial steady state conditions, the overall flux of neutrons being simulated can begin to oscillate. This is known to occur within Serpent 2 transient simulations and was observed when simulating the SPERT experiments [25]. The main reason this “drift” occurs is not accounting for the total heating and cooling of materials as the transient progresses.

Another contributing factor is the time-step and total number of neutrons being simulated in Serpent 2 as they play a direct role in the accuracy of the simulation. A small time-step like 2E-6 seconds is necessary for the simulation of HENRI as it allows for the capture of the shockwave propagation as discussed in Section 3.5. However, when considering the Monte-Carlo nature of Serpent 2 such a small time-step requires a large number of neutrons to be simulated in that small timeframe, this can become exceedingly computationally intensive. To reach a higher spatial resolution, and thus possibly reduce the oscillations seen in Figure 4.1.1 simulating more than 200,000,000 neutrons could be undertaken. However, acquiring the computational power necessary to achieve this is outside the scope of the report. As such the current number of neutrons being simulated is deemed adequate and is used in the other simulation presented.

With these results, it can be shown that the coupling of Serpent 2 and STAR-CCM+ can be verified as any information passed by Serpent 2 is correctly simulated by STAR-CCM+ with a slight error due to the mismatch in spatial overlay. It can also be shown that the oscillations seen in verification originate from the Serpent 2 code and can possibly be corrected by adding extra consideration for heating and cooling of materials, number of neutrons being simulated, and time-step sensitivities.

4.2.Helium-3 Self Shielding

An important reason that a coupled multi-physic code was developed to be used in the evaluation of the HENRI cartridges was the potential for the phenomena of self-shielding. As discussed previously, self-shielding occurs when a material effectively shields itself from an incoming neutron flux. In other words, the higher density or absorption cross section of a material, the more likely that neutronic reactions will happen on the periphery of the material rather than on the inside as neutrons cannot penetrate as deep in the material.

With regards to HENRI this phenomenon can be important to the overall progression of the helium-3 injection. As the density within HENRI increases more of the neutronic reactions will

occur on the periphery of the HENRI cartridge. This will result in a vast majority of the energy deposition and thus potential temperature increase to occur there. It also leads to non-uniform temperature, and density radial distribution. To understand the effect this may have on the axial density time-evolution of the helium-3 gas, the Serpent 2 detector inputs discussed in Section 3.8.4 were developed. These inputs divide the HENRI fuel assembly discussed in Section 1 radially so that the effect of self-shielding may be studied.

Seven steady state-simulations of the HENRI-TREAT model were executed. In these simulations the overall density of the helium-3 gas within the HENRI cartridge was modified. The density of helium-3 was assumed to be uniform across all of HENRI. The density values studied ranged from 0.05 kg/m^3 to 10.0 kg/m^3, these values were chosen in order to cover a wide basis. The steady state runs were performed using 50 inactive cycles, 320 active cycles, and 100,000 neutrons with the power of the reactor set to 16,000 MW.

To visually represent the effect of self-shielding, the following figures were created with a mesh plot in Serpent 2. The mesh-plots depicted in Figure 4.2.1 show the neutron-capture reactions occurring within the HENRI cartridge, where areas of red indicate a large number of interactions and areas of blue indicate a small number of interactions.

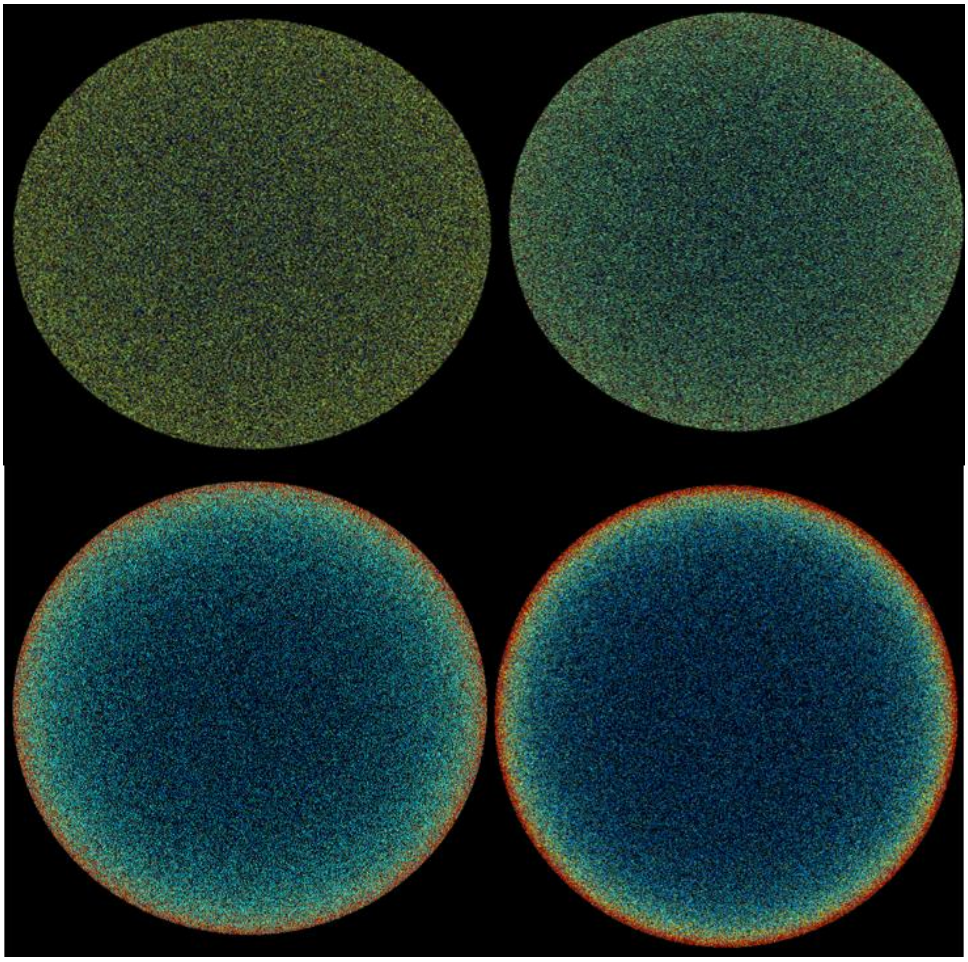


Figure 4.2.1 Helium-3 self-shielding, 0.5 kg/m^3 (top left), 1.0 kg/m^3 (top right), 5.0 kg/m^3 (bottom left), 10 kg/m^3 (bottom right)

From Figure 4.2.1, once the density of helium-3 within the HENRI cartridge rises above 1 kg/m^3 self-shielding starts to become more dominant. The figure of merit of the detector discussed in Section 3.8.4 is the volumetric heating that occurs due to the (n,p) and (n,γ) reactions that occur within helium 3.

In order to capture this effect within Serpent 2 the radial resolution of the volumetric heating detector discussed in Section 3.8.4 must be defined. A number of detectors were tested that changed the overall radial resolution. The resolutions tested were 5, 8, and 10 radial points.

Figure 4.2.2 through Figure 4.2.4 show the radial volumetric heating that occurs within the HENRI cartridge for the seven HENRI-TREAT steady state runs using multiple radial definitions.

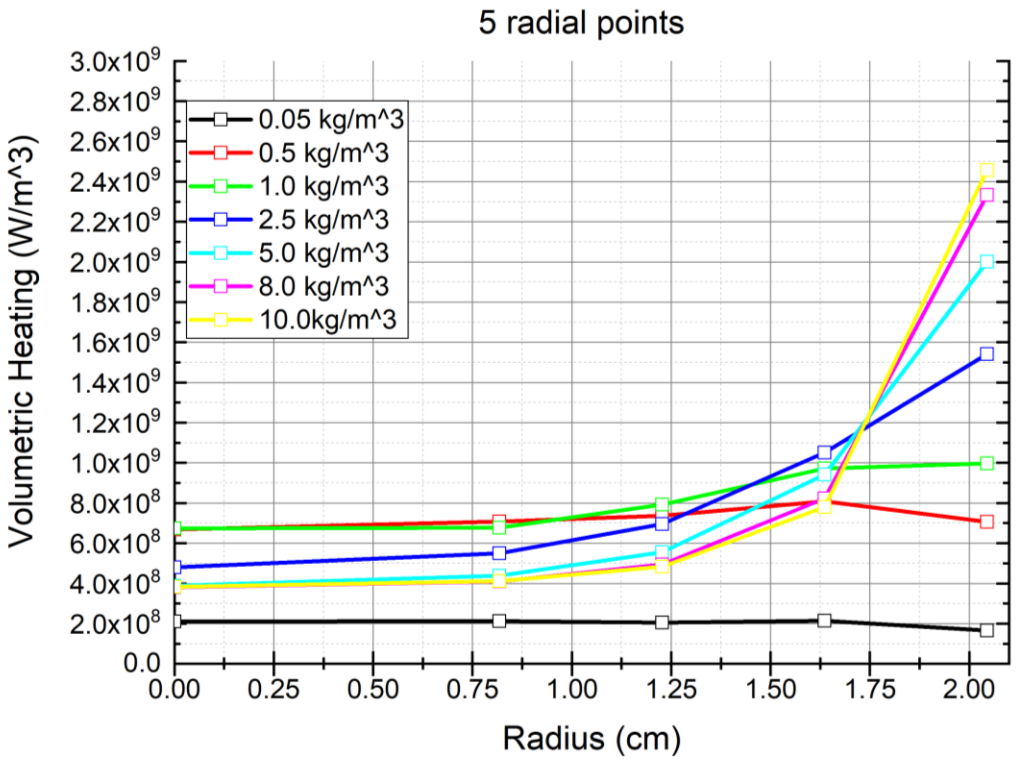


Figure 4.2.2. HENRI radial (5 points) volumetric heating

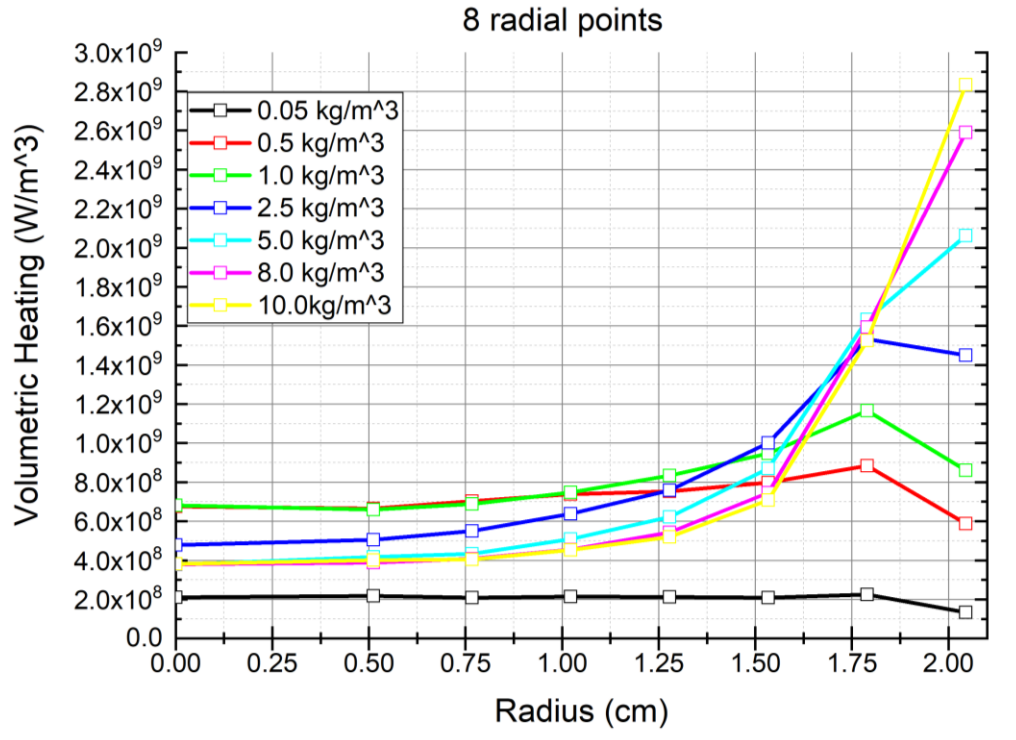


Figure 4.2.3. HENRI radial (8 points) volumetric heating

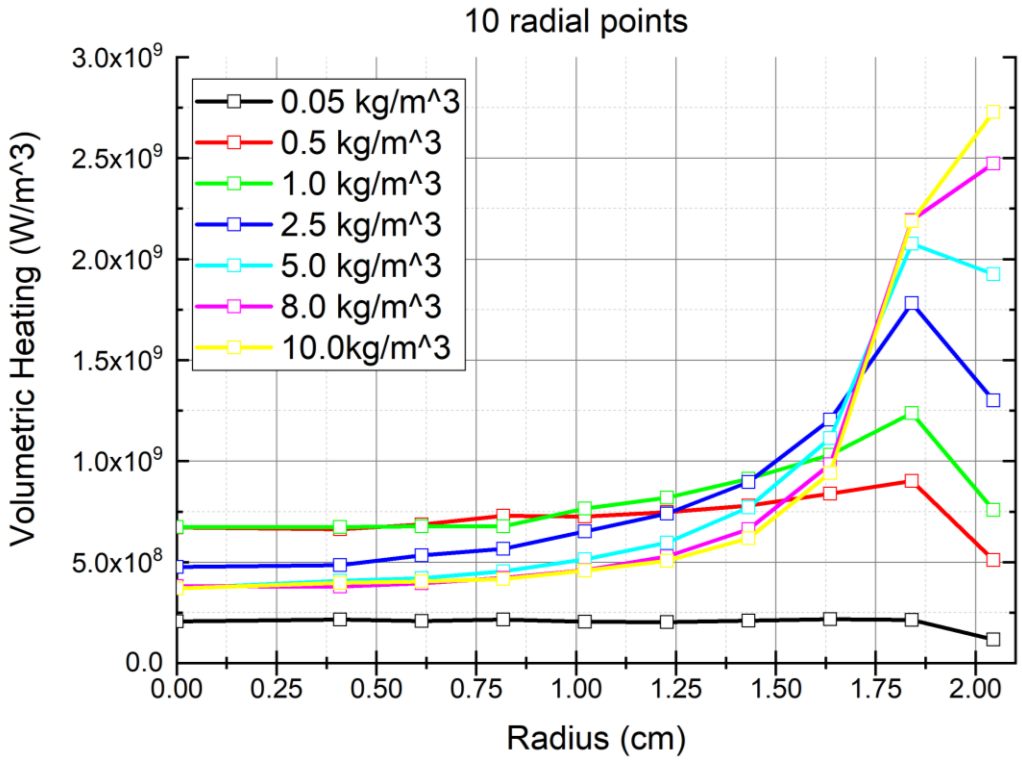


Figure 4.2.4. HENRI radial (10 points) volumetric heating

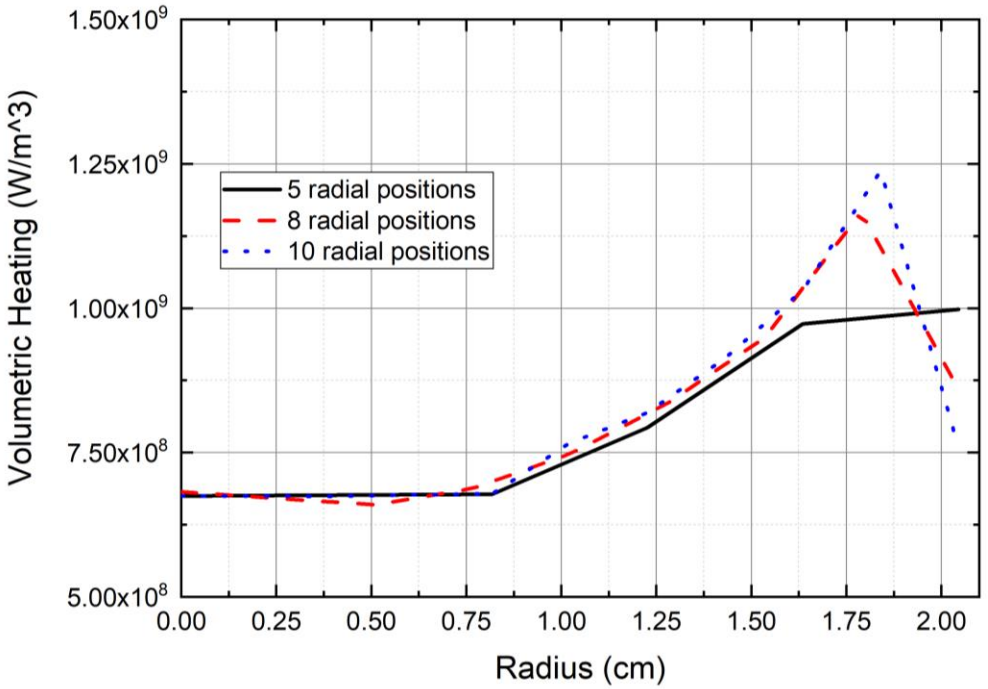


Figure 4.2.5. Self-Shielding radial resolution comparison at 1.0 kg/m^3

From Figure 4.2.2 through Figure 4.2.4, as the radial resolution increases so does the definition of the self-shielding. Figure 4.2.5 compares the radial volumetric heating for all the radial resolutions considered for a density of 1.0 kg/m^3. With 10 radial points the penetration depth of helium-3 is well defined with much of the volumetric heating occurring between 1.75 cm and 1.84 cm. This would lead to the conclusion that using a radial definition of 10 would be the most accurate. However, an increasing radial resolution leads to a decreasing size in the detector volume. The volume of each detector mesh when using a radial definition of 10 is 0.2239 cm. This decreasing size can have a profound effect on the statistical accuracy of the Serpent 2 detector in transient simulations. To combat the decrease in statistical accuracy a larger number of neutrons would need to be simulated thus increasing the computational cost of the simulation.

It should be noted that due to Serpent 2 taking the average across each detector volume the total amount of volumetric heating stays the same in the overall detector regardless of the radial resolution, the only change being the location of the volumetric heating along the radius of HENRI.

Therefore, though the accuracy of using a radial resolution of 8 or 10 may be higher, the increase in statistical uncertainty and subsequent computational cost to correct that uncertainty is undesirable. It is recommended to use a radial resolution of 5 when the effect of self-shielding is desired to be captured within HENRI. As a resolution of 5 is sufficient to capture the overall profile of the observed self-shielding while maintaining a sufficient enough size in the detector mesh that statistical uncertainty is kept to a minimum.

4.3.Full HENRI Coupling Run

With CONSTELATION verified in Section 4.1 and an understanding of the effect of self-shielding on helium-3, a full coupling run was performed. This run consisted of the HENRI driver tank pressured to 3.45 MPa [500 psi] an initial test section temperature of 875 K . The wall boundary conditions of the STAR-CCM+ simulation were set to a best-estimate convective boundary condition. In Serpent 2 the radial definition of the volumetric heating detector was set to 5. Serpent 2 was run with 200,000,000 neutrons over 500 batches per timestep. The calculation was conducted for a peak power of 16,000 MW with four HENRI cartridges.

4.3.1 Density, Pressure, and Temperature Evolution within HENRI

Figure 4.3.1 shows the point locations used to extract the time evolution data of the coupled simulation. These points correspond to the location of thermocouples and pressure transducers found in the original HENRI facility at OSU. The relative location of the points to the top and bottom of TREAT is also delineated. TS05 is not shown here but is located at the end of the extension section of HENRI detailed in Figure 1.1.1. The density and pressure time-evolution of the helium-3 gas are presented for the 1st Group of HENRIs in Figure 4.3.2 and for the 2nd Group of HENRIs in Figure 4.3.3. Temperature evolution for both groups are presented in Figure 4.3.5 and Figure 4.3.6. These properties evolution illustrates the fluid dynamics side of the coupled phenomena being investigated. Note that the overall density of helium-3 seen within HENRI drives the overall negative reactivity insertion i.e., a higher density of helium-3 results in a larger negative reactivity insertion. On the other hand, temperature and pressure will affect the safety margins that need to be considered in the design and operation of the HENRI cartridges in TREAT.

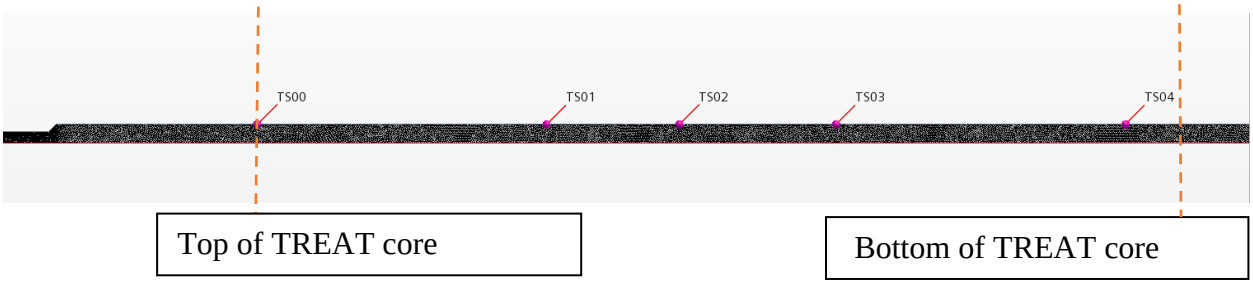


Figure 4.3.1. HENRI data point locations

The following plots show the density, pressure, and temperature time-evolution obtained from the STAR-CCM+ simulations for both “groups” of HENRIs along the points shown in Figure 4.3.1.

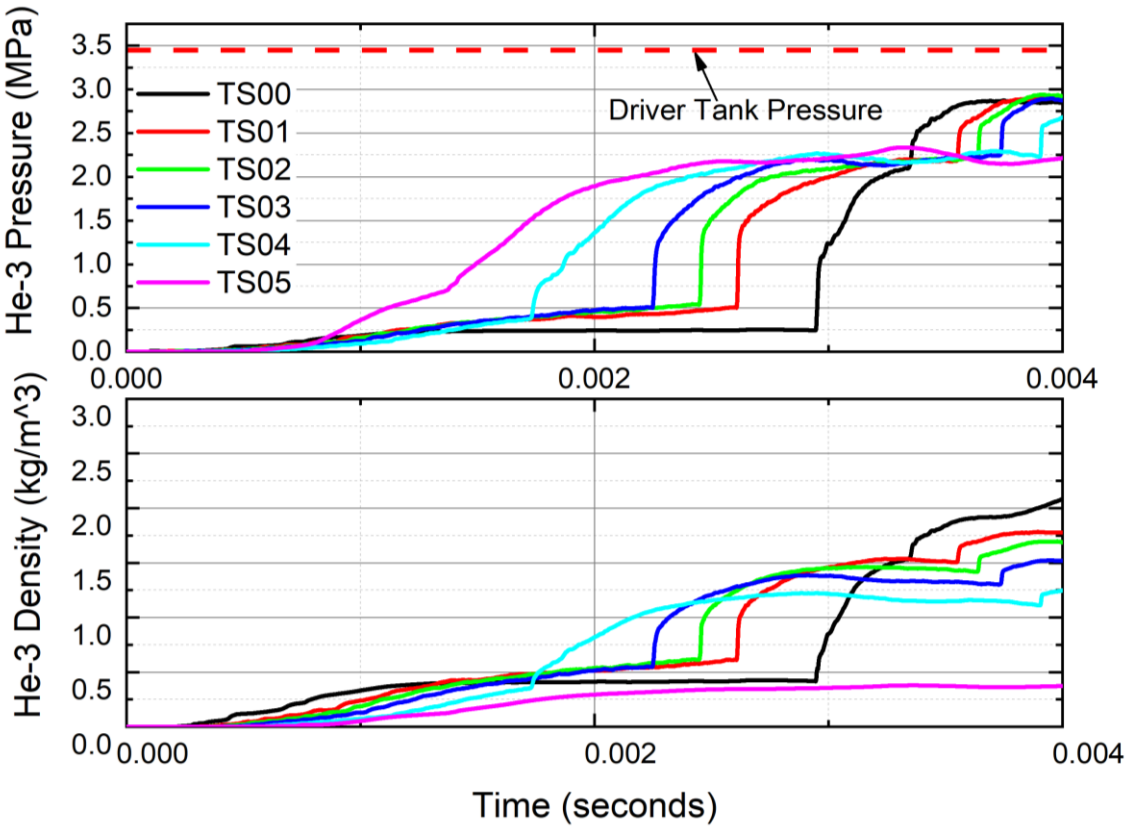


Figure 4.3.2. Pressure and density evolution for 1st Group of HENRI cartridges, experiencing lower neutron flux

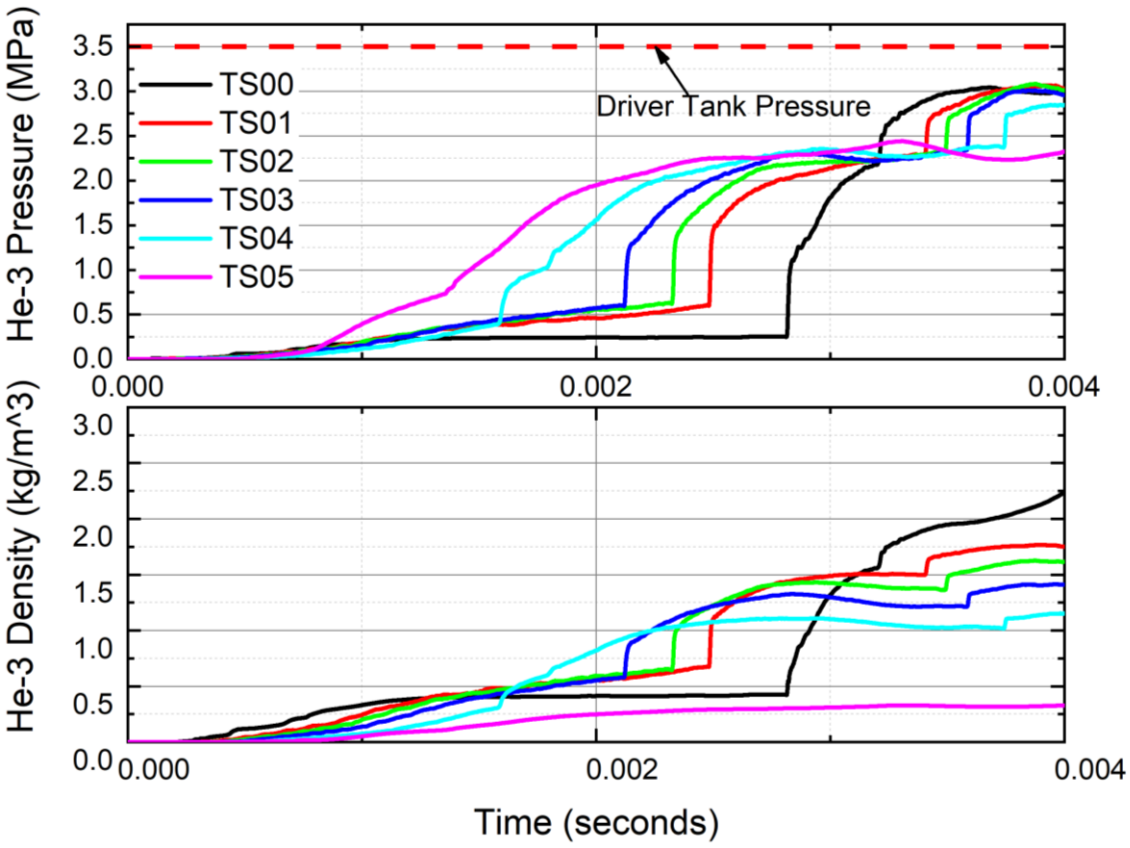


Figure 4.3.3. Pressure and density evolution for 2nd Group of HENRI cartridges, experiencing higher neutron flux.

The above plots demonstrate the wave propagation of the helium-3 gas through the HENRI test section. This wave propagation is caused by the pressure difference between the driver tank and the close to vacuum test section. When the transient is started the vast pressure difference between the driver tank and test section causes a pressure wave of helium-3 to begin moving through the test section.

As seen in both Figure 4.3.2 and Figure 4.3.3, the first notable increase in pressure occurs at the TS05 data point. This point is located at the very end of the HENRI extension section, and the increase in pressure is a result of the helium-3 pressure wave striking the end of the extension section. From the beginning of the simulation, a slight increase in pressure can be seen through all six data points, this is a result of the flow of helium-3 gas from driver tank following the initial pressure wave. The next notable increase in pressure is seen at data point TS04, which is located just at the bottom of TREAT as noted in Figure 4.3.1. This is a result of the pressure wave reflecting off the end of extension section and traveling back along the length of the test-section. This

reflected shockwave continues to pass through the test section resulting in local pressure “spikes” for the remaining data points.

The direct effect of these pressure increases is the increase of local density. As seen in all data points but that of TS05, the density of helium-3 gas is constantly increasing. This is a result of the flow of helium-3 gas from the 3.45 MPa (500 psi) pressurized driver tank into the less pressurized test section. TS05 does not result in a large density increase, despite having a high local pressure, this is a result of the extremely high temperatures at that position in the HENRI cartridge as seen in Figure 4.3.5 and Figure 4.3.6.

A noticeable effect of the inclusion of volumetric heating can be seen in the density gradient seen in the HENRI cartridge test-section. The difference in density across the 1st Group of HENRI cartridges is 1.0 kg/m^3 and the 2nd Group has a difference of 1.25 kg/m^3 This is a direct result of the inclusion of volumetric heating in the coupling. As the local density increases, so does the local volumetric heating which results in a local temperature increase. This is coupled with the fact that throughout this process colder helium-3 gas from the driver tank, which is assumed to be at 300 K, is constantly flowing into the test-section. As detailed in Figure 4.3.1, TS00 is located just at the top of the TREAT core, this results in the helium-3 gas at TS00 entering at close to 300K and only partially being subjected to volumetric heating before passing through and being heated continuously through the test-section resulting in overall lower densities through the remaining data points.

Figure 4.3.2 and Figure 4.3.3 show that the pressure after 4 msec of simulation for the coupled simulation reaches 2.76 MPa [400 psi] and the density at this time is well above 1.0 kg/m^3. In fact, the density of the majority of the test section of the HENRI cartridge rises above 1.0 kg/m^3 around 2.5 msec meaning that for about half of the stated mission time the effect of self-shielding will be noticeable.

Figure 4.3.4 details the pressure and density evolution of a stand-alone STAR-CCM+ simulation of a single HENRI cartridge. This simulation is not coupled with Serpent 2 and does not account for any volumetric heating caused by the absorption of neutrons within the helium-3 gas. This is used to observe the effect of the coupling process and the importance of its inclusion.

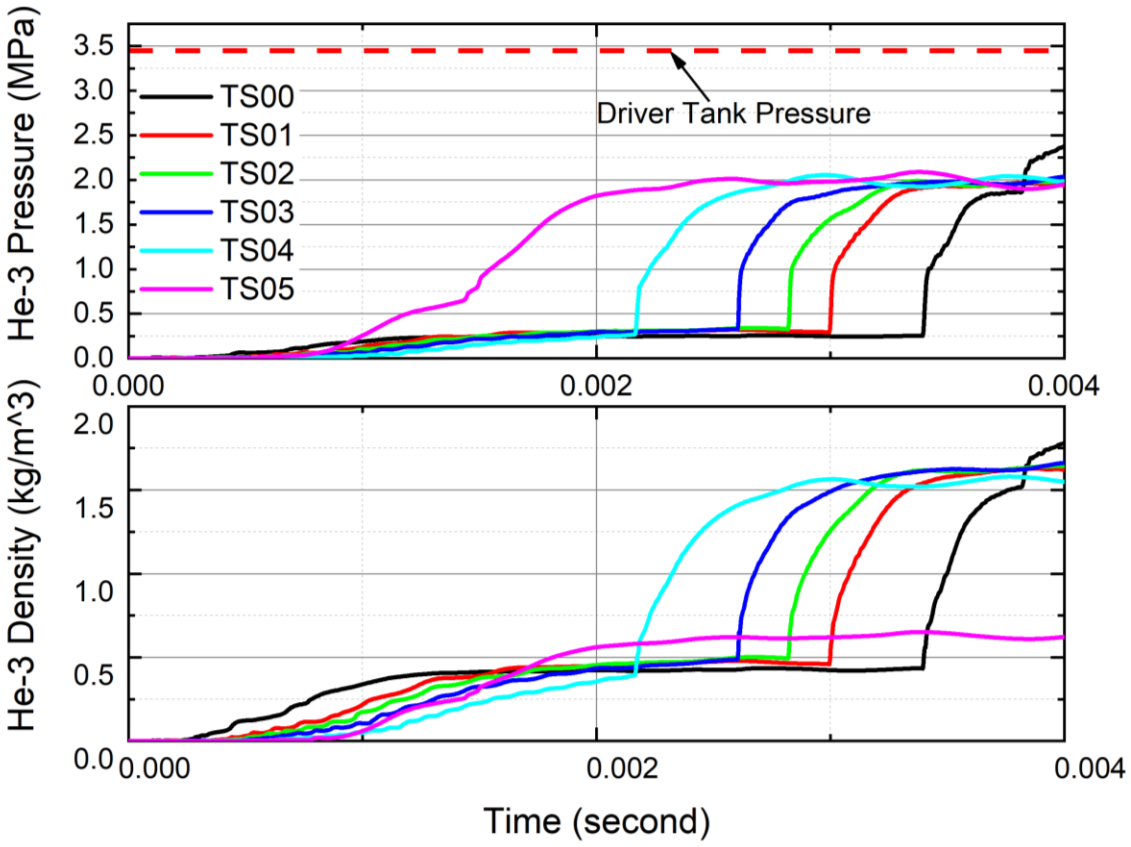


Figure 4.3.4. Pressure and density evolution for non-coupled HENRI cartridge

From Figure 4.3.4 the same general progression of local pressure increases occurring due to the passing of the pressure wave is observed. Three main differences can be observed when comparing the stand-alone simulation to that of the coupled one. The first being the obvious timing differences in the pressure wave, the second being the overall lower pressures and on average higher densities, and the third being the lack of a noticeable density gradient.

Shockwave propagation is dependent on variety of phenomenon with one being the temperature of the medium and the heat input as it is propagating through. The reason being that as temperature increases the speed of sound increases. Thus, the higher the local temperature the faster a shockwave, or pressure wave, can pass through it. Figure 4.3.5 through Figure 4.3.7 detail the observed temperature evolution for the two groups of HENRI cartridges in the coupled simulation and the non-coupled HENRI cartridge.

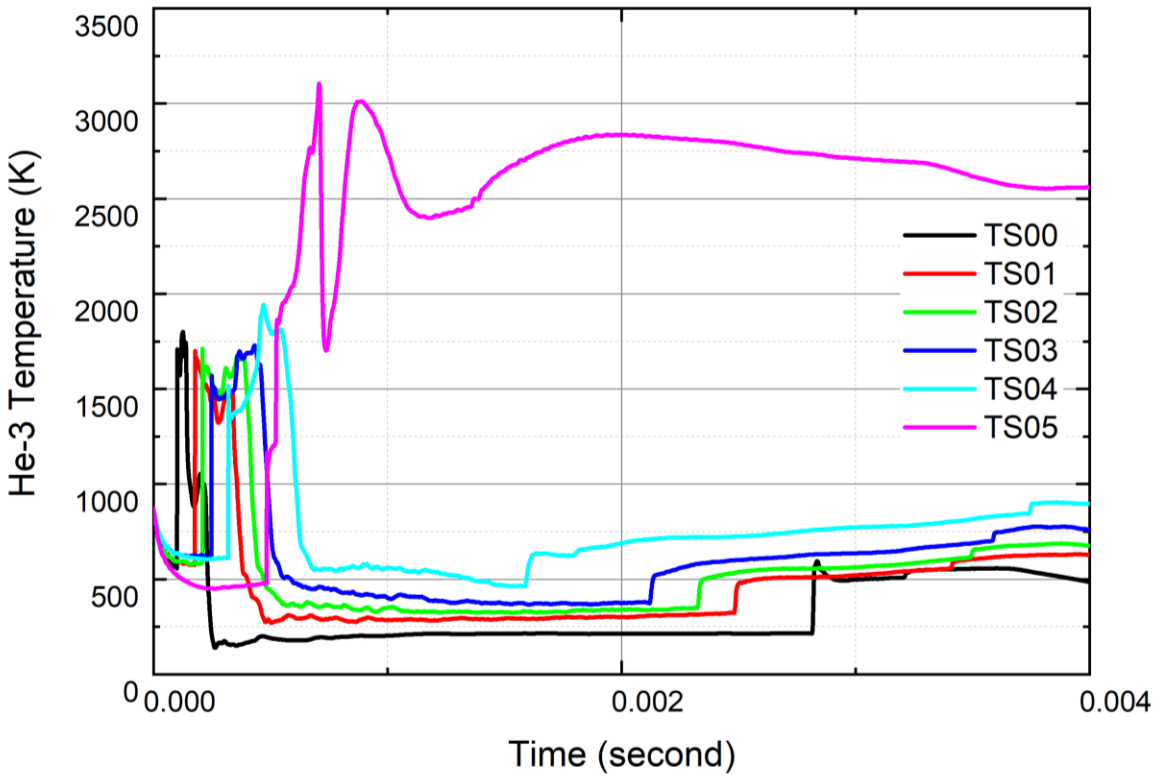


Figure 4.3.5. Temperature evolution for 1st group of HENRI cartridges.

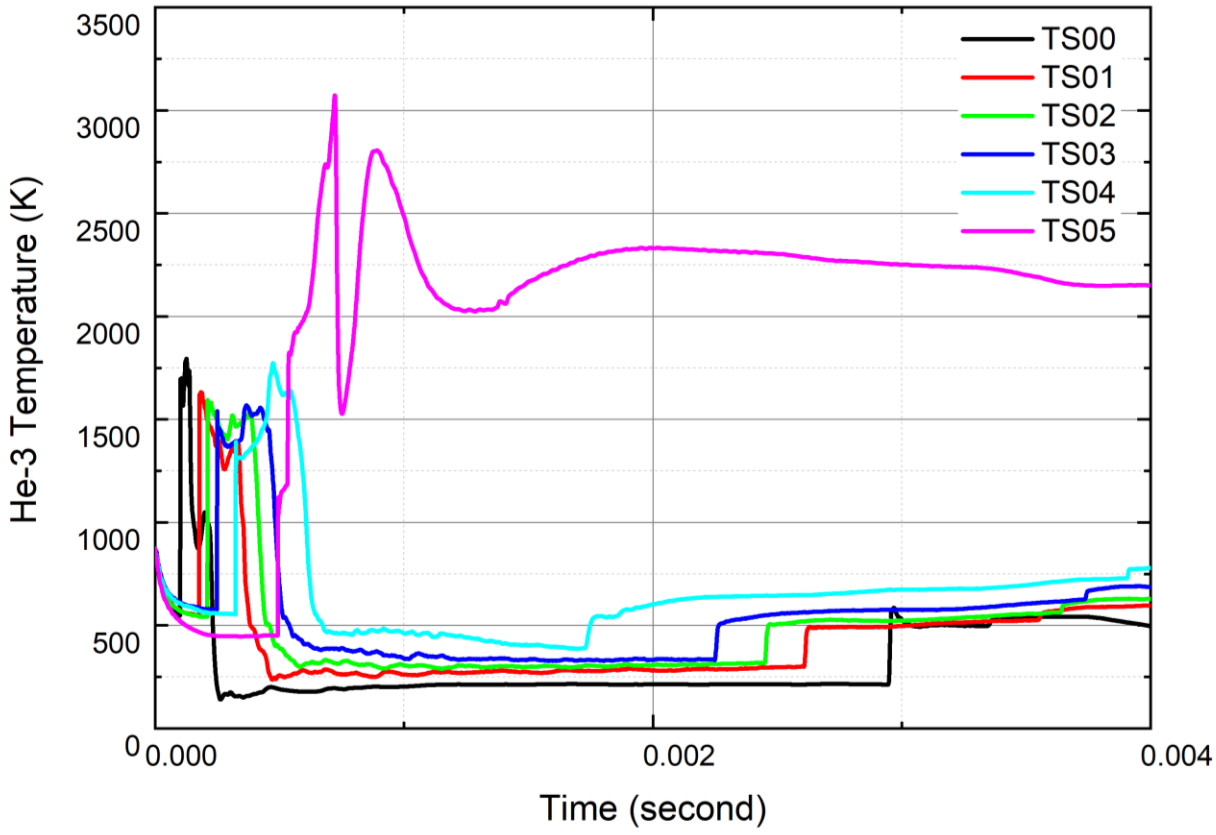


Figure 4.3.6. Temperature evolution for 2nd group of HENRI cartridges.

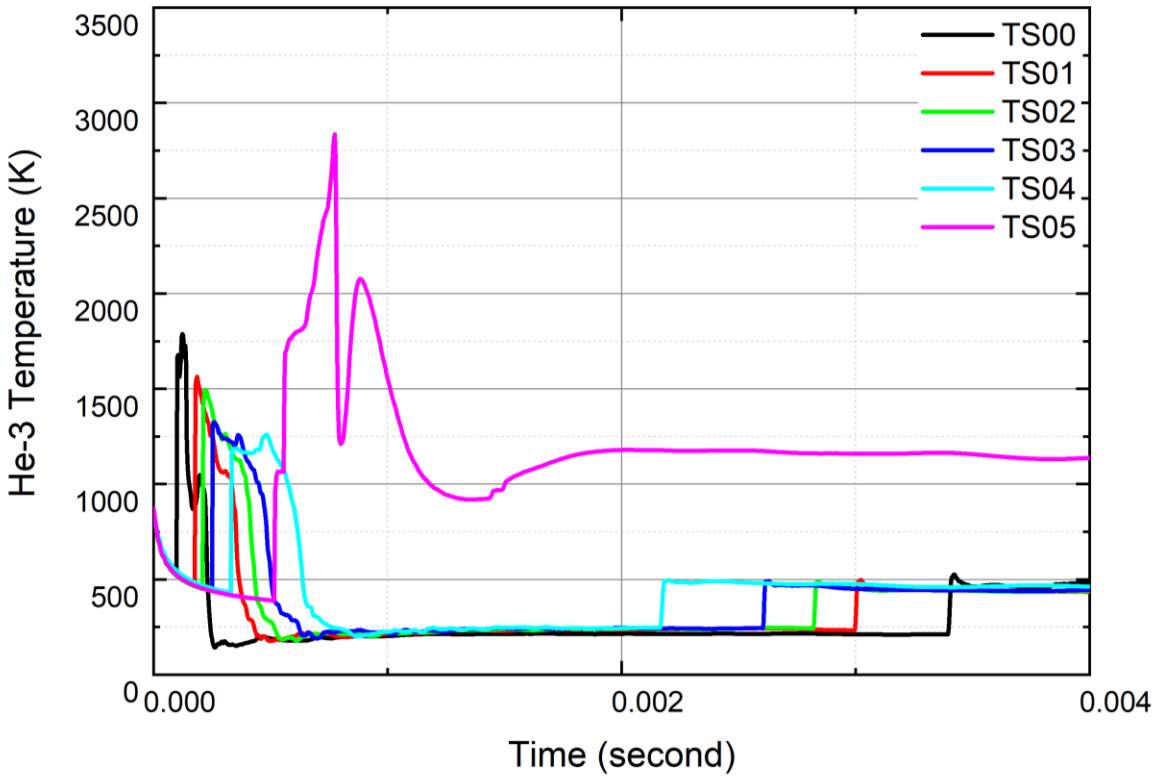


Figure 4.3.7. Temperature evolution for non-coupled HENRI cartridge.

For both the coupled and non-coupled simulations, the temperature of the helium-3 gas within the HENRI test section initially starts at the assumed temperature of the gas at 873 K (600 ^{O}C), while the temperature of the driver tank, not pictured here is assumed to be at a room temperature of 300 K (25 ^{O}C). The temperature initially starts to decrease due to the assumed convective boundary and the initial injection of the 300 K helium gas entering the test section from the driver tank . The initial spikes seen are a result of the initial shock wave of helium-3 passing through the system.

For the coupled simulation, after the initial shock wave passes through the test section the temperature in the test section drops to an average of 400 K with a noticeable temperature gradient. It is not until the density of the helium-3 gas rises above 0.5 kg/m^3 (Figure 4.3.2 and Figure 4.3.3) that the temperature rises to around 600 K. From here the volumetric heating occurring within the helium-3 gas causes the temperature to increase for the rest of the transient resulting in a value of 776 K for the 1st Group and 885 K for the 2nd Group after 4 milliseconds of simulation. For the 1st Group there is a temperature difference across the test section of 298 K and for the 2nd Group there is a temperature difference across the test section of 420 K after 4 milliseconds of simulation.

For the non-coupled simulation, the temperature of the test section is significantly lower. The temperature in the test section drops to an average of 300 K just after the shockwave initially passes through the test section. The first 4 milliseconds of simulation ends with a max temperature of 500 K (Figure 4.3.7). These lower temperatures are a direct result of not including the volumetric heating expected to occur when injecting helium-3 gas into TREAT. It is also observed that the non-coupled simulation experiences no significant gradient in temperature.

Figure 4.3.7 allows for the differences in the results of the non-coupled and coupled simulations observed previously to be explained. As noted, shockwave propagation is partially dependent on the temperature of the medium. As the temperature of the test section is significantly lower in the non-coupled simulation the shockwave of helium-3 gas moves slower through HENRI resulting in a delay of the pressure spikes observed in Figure 4.3.4. In addition to this slower propagation, the overall lower pressures and higher densities is directly explained by these lower temperatures as higher local temperatures result in higher local pressures and lower local densities. The lack of a gradient seen in the non-coupled simulation can be explained by the lack of volumetric heating, as volumetric heating is directly related to local density i.e., as local density increases so does local volumetric heating. Therefore, in the coupled simulation when the density increases, so does the volumetric heating, which results in an increase in local temperature. As the density of the helium-3 gas is not even throughout the test section this results in a gradient within the coupled simulation which is not present in the non-coupled simulation.

For both the coupled and non-coupled simulations TS05, located at the end of the extension section, below the active core, sees higher overall temps. This is a result of the helium-3 gas being compressed against the end of the extension section due to the shockwaves that pass through the system coupled with the flow of helium pushing through the test section. This compression causes a large amount of heat-up. This high amount of heat up-also results in a lower overall density as noted in Figure 4.3.2 through Figure 4.3.4.

In an effort to directly observe the difference seen between the 1st and 2nd Group of HENRI cartridges, a series of plots are compared in Figure 4.3.8 through Figure 4.3.10. These plots compare the pressure, density, and temperatures of the six data points detailed in the above figures. The 1st Group of HENRI cartridges are shown with a dotted line, while the 2nd Group of HENRI cartridges is shown with a solid line.

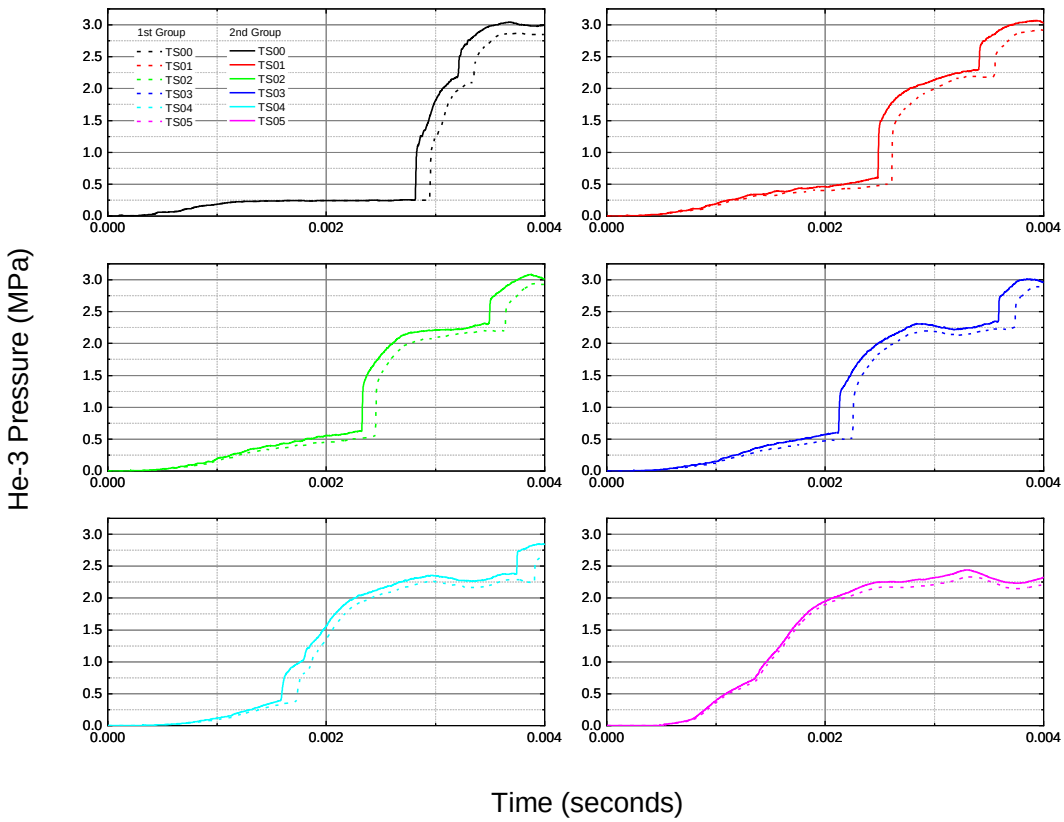


Figure 4.3.8. Pressure comparison of the 1st and 2nd group of HENRI cartridges.

Figure 4.3.8 details the pressure difference for all six data points. From this comparison it can be seen that the 2nd Group of HENRI cartridges, which are subject to a higher flux than the 1st Group, have an overall higher gas pressure with an average difference of 0.35 MPa (50 psi) greater than the1st Group. This can be explained by more heat being added to the gas system at the test section level. It should be noted that a slight difference in timing exists between the “jumps” in pressure. These jumps are 1/10th of a millisecond apart. The reason for this difference can be seen in Figure 4.3.9 where the temperature difference between the two groups of HENRI cartridges is detailed

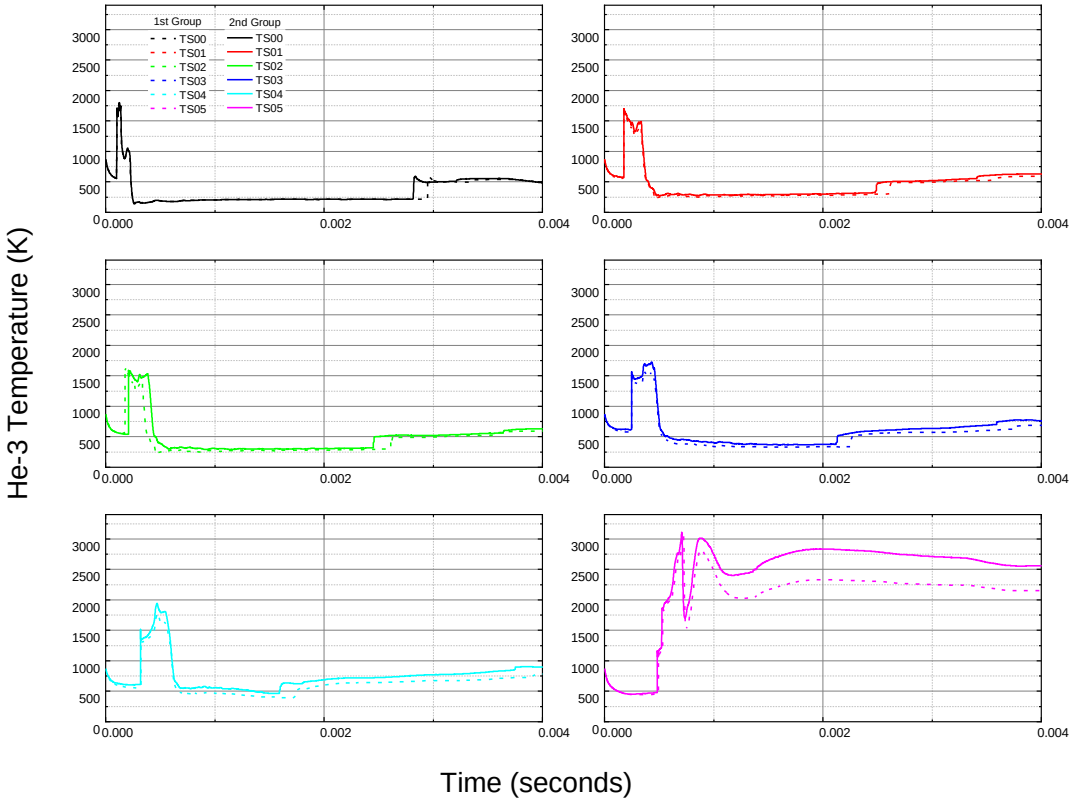


Figure 4.3.9. Temperature comparison of 1st and 2nd group of HENRI cartridges

The 2nd Group of HENRI cartridges subjected to the higher flux from TREAT has an overall higher temperature of the helium-3 gas. The max difference is seen in TS04 where the difference is 150 K with the average difference being 50 K. This trend is observed for the other data points but with a slightly smaller difference. TS04 sees the highest difference as it is the location in the test-section where the density increases first, as detailed in Figure 4.3.10

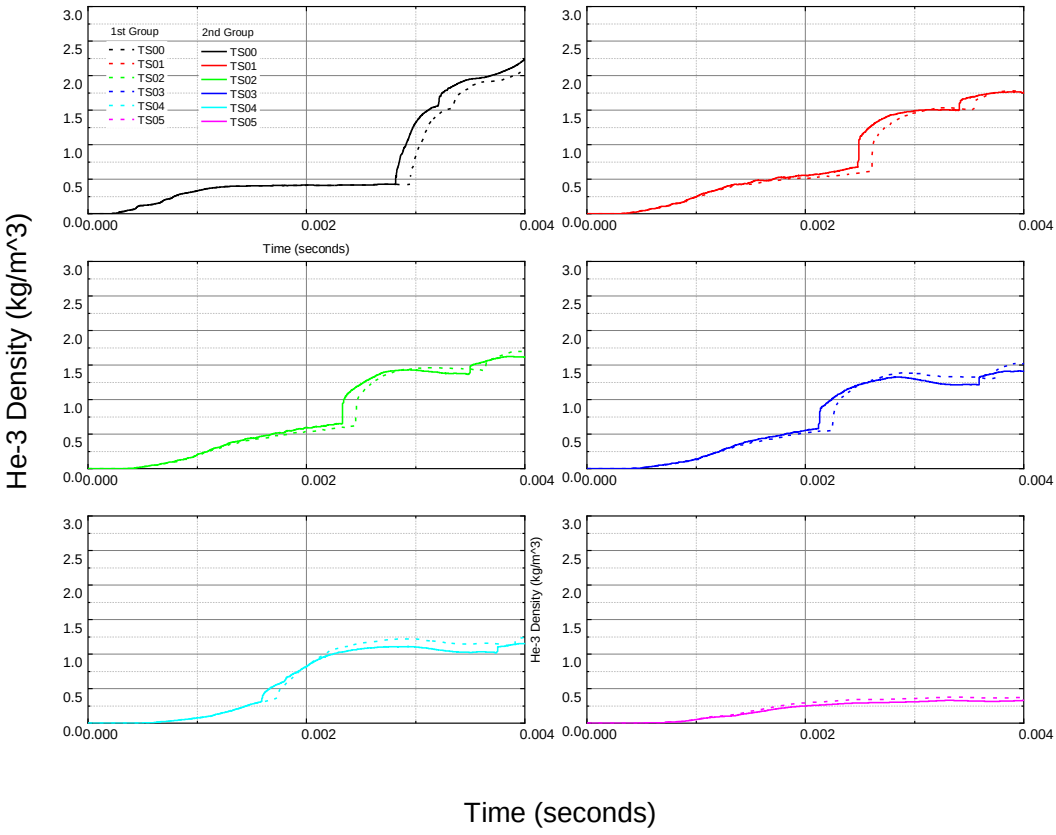


Figure 4.3.10. Density comparison of 1st and 2nd group of HENRI cartridges

With a higher density the volumetric heating due to the neutron flux of TREAT increases, therefore TS04 is subjected to a longer period of volumetric heating than the other data points resulting in a higher overall temperature. In this same vein the slight difference in timing seen by the 1st and 2nd Groups of HENRI cartridges can be explained. As with a higher local temperature, the shockwave of helium-3 gas within the 2nd Group of HENRI cartridges moves at a faster speed than the shockwave of helium-3 gas within the 1st Group of HENRI cartridges.

From Figure 4.3.10, the density difference between the two groups of HENRI cartridges is less profound with an average difference of 0.1 kg/m^3. The max difference is seen in TS00 where the difference is 0.25 kg/m^3 with the 2nd Group of HENRI cartridges having the larger of the two densities, this can be explained due to the temperature difference discussed above resulting in the shockwave propagation occurring slightly faster in the 2nd Group of HENRI cartridges.

4.3.2 Volumetric Heating within HENRI

The volumetric heating obtained from SERPENT 2 over the course of the full coupling run is presented in Figure 4.3.11 through Figure 4.3.14. To obtain these volumetric heating profiles, the axial density distribution from the initial coupling run is taken and inserted into a criticality run of TREAT using Serpent 2. This is done in order to reduce the statistical uncertainty of the results that were observed in the verification and provide a more representative result of the volumetric heating occurring within the helium-3 gas.

The results are presented axially across TREAT for distinct points in time. The results from Serpent 2 are averaged across each detector point. This is done to get a representative understanding of the effect that the neutron flux found within TREAT has on the overall volumetric heating when considering the changing density of the helium-3 gas within HENRI.

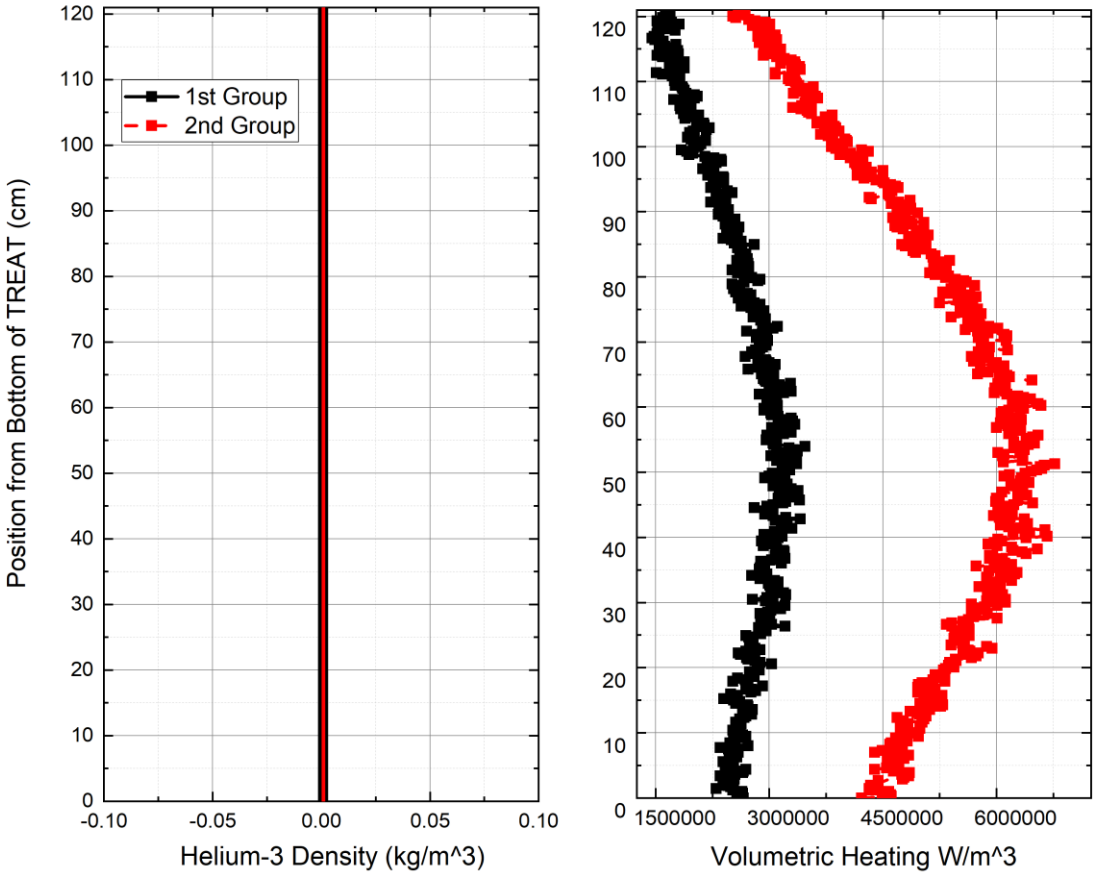


Figure 4.3.11. Axial density (left) and volumetric heating (right) of helium-3 gas within HENRI @ 0 msec

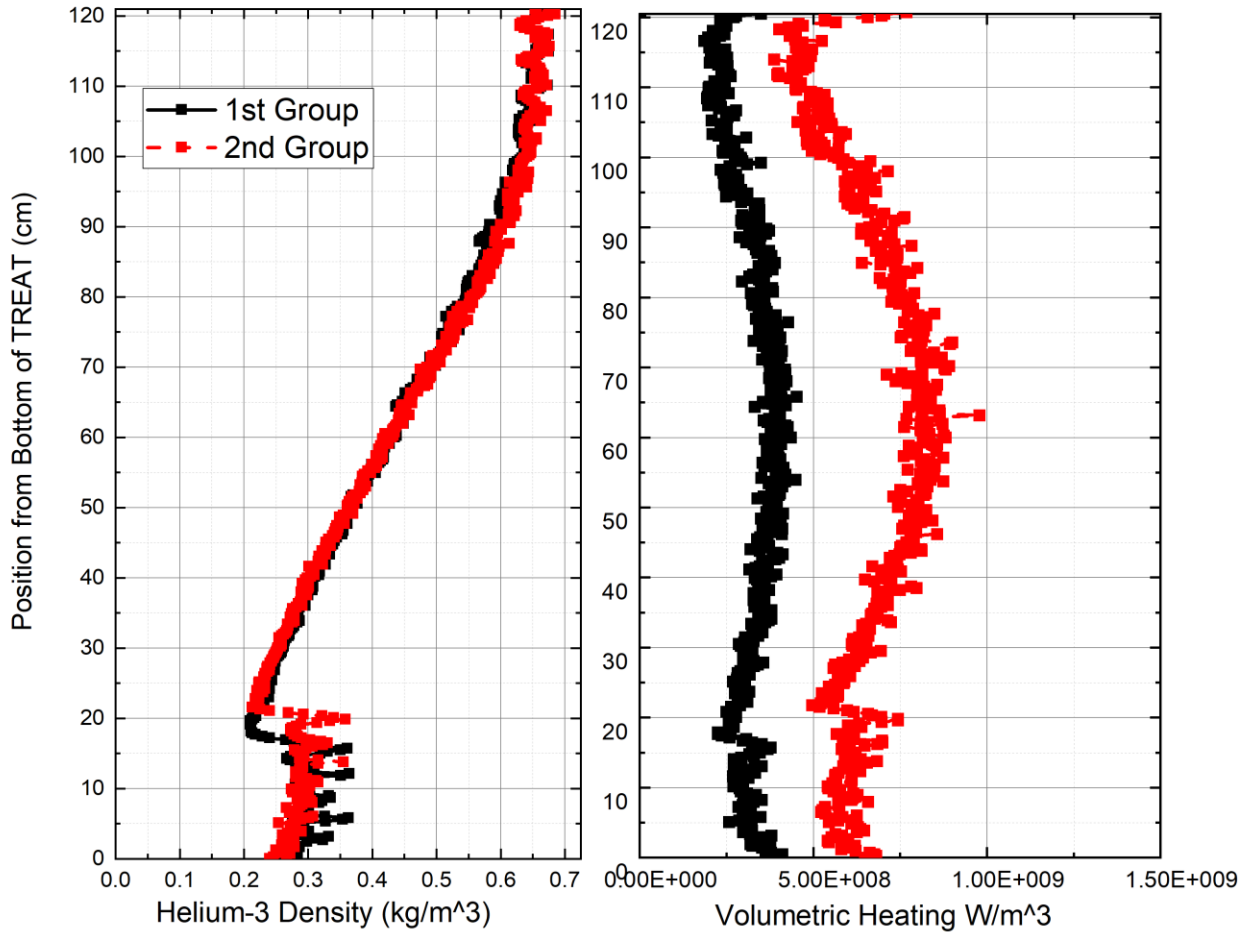


Figure 4.3.12. Axial density (left) and volumetric heating (right) of Helium-3 gas within HENRI @ 1.25 msec

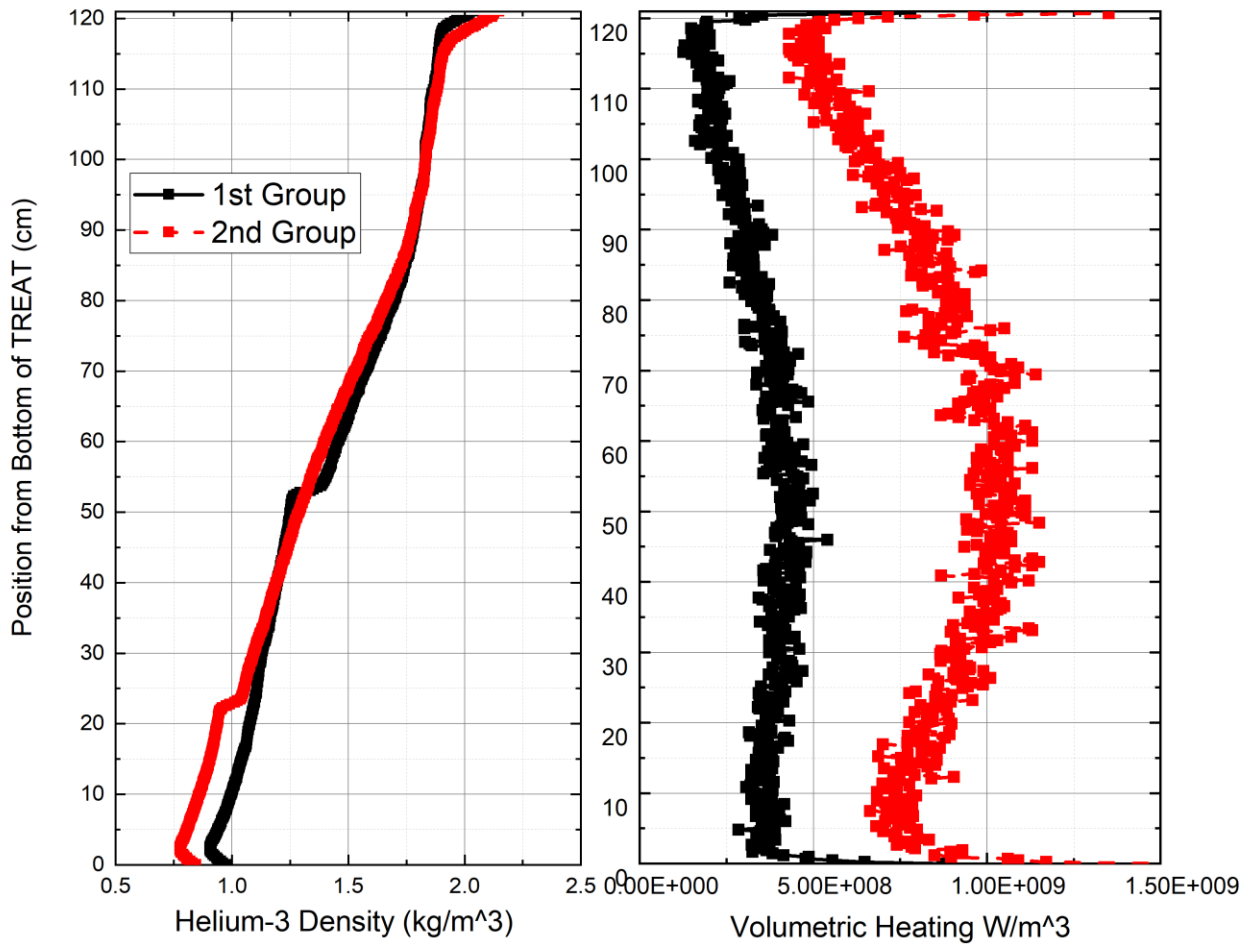


Figure 4.3.13. Axial density (left) and volumetric heating (right) of helium-3 gas within HENRI @ 3.75 msec

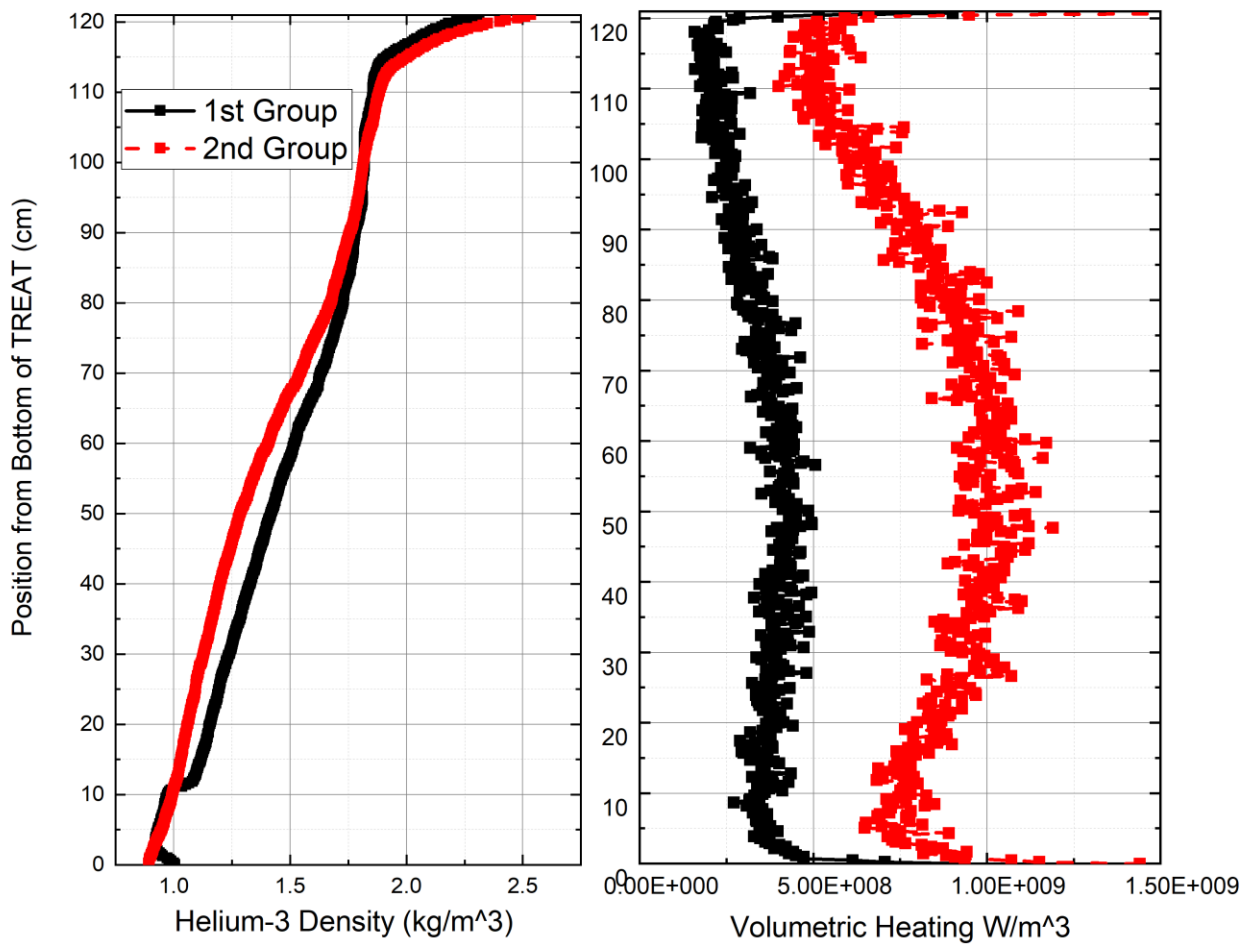


Figure 4.3.14. Axial density (left) and volumetric heating (right) of helium-3 gas within HENRI @ 4.0 msec

From the plots shown above, the volumetric heating exhibits the expected results. In a nuclear reactor the neutron flux is higher near the centerline of the reactor, which is observed in all of the figures. The spikes observed at z = 0 and z = 120 are a result of the upper and lower core reflectors found in TREAT, which reflect a large number of neutrons escaping the reactor back into the active core region. In addition, the volumetric heating seen by the 1st Group of HENRIs detailed in Figure 3.8.6 is noticeably lower than that seen by the 2nd Group of HENRIs even for similar densities. This is a result of the different fluxes seen by the two groups of HENRI as is detailed in Figure 3.8.5.

From Figure 4.3.12, it can be seen that Serpent 2 captures the effect of the shockwave. This can be seen as the change in density near the bottom of TREAT is noticeable in the volumetric heating calculated by Serpent 2. Additionally, the effect of self-shielding is reaffirmed with these results

as, in Figure 4.3.12, the volumetric heating changes proportionally with the density profile, however in both Figure 4.3.13 and Figure 4.3.14, where the average density is above 1 kg/m^3 the profile of volumetric heating maintains a consistent shape with only the magnitude of the heating changing with increasing density.

4.3.3 Negative Reactivity Worth of HENRI

The overall goal of utilizing the HENRI system in TREAT is to be capable of inserting 5% negative reactivity to hasten the clipping of the reactor and better simulate a postulated RIA accident within an LWR. The initial coupling results detailed in the previous sections can be used to demonstrate whether this goal is feasible.

The percent amount of reactivity, positive or negative, inserted into a nuclear reactor can be calculated using the following equation.

$$\Delta\rho\left[\%\frac{\Delta k}{k}\right] = \frac{k_2 - k_1}{k_2 k_1} * 100 \qquad \text{Eq.(21)}$$

Where $\Delta\rho$ is percent change in reactivity, k_1 is the k effective of the reactor at the initial state, and k_2 is the k effective of the reactor at some altered state. Eq.(21) uses the k effective of a reactor, which is defined in Eq.(2) and is essentially a measure of how many neutrons are being born and lost in a reactor at any given moment in time. As such, a k effective value is only obtainable in an assumed steady state configuration and cannot be accurately calculated for transient conditions, as moving the neutrons through time causes the information of how many neutrons are being born and lost in an instant to be lost.

Serpent 2 is designed to calculate the k effective for any given system using its criticality calculation mode but cannot accurately predict the k effective of a system during the execution of a transient. This does not mean that the overall negative reactivity of a system cannot be calculated for transient. It can still be captured by taking discrete data sets obtained during the transient calculation and running them with Serpent 2 in the criticality calculation mode. These discrete data points consist of the local temperature and local density distribution of the helium-3 gas within a HENRI cartridge. This essentially creates a snapshot of the reactor during a transient and can be used as an equivalent way to calculate the k effective of a system during transient conditions.

This method was implemented to calculate the overall negative reactivity inserted during the initial coupling run detailed in Section 4.3. The overall density profile for both groups of HENRIs, the current power of TREAT, as well as the fuel temperature were extracted for discrete points of time and then implemented into the input deck described in Section 3.8.3. Serpent 2 was then executed in steady state mode and the k effective of the simulation taken and used to calculate the overall negative reactivity inserted at that point in time. The initial state k effective was taken to be the k effective of the unaltered input deck detailed in Section 3.8.3.

Using the method detailed above and Eq.(21), the overall negative reactivity insertion from four HENRIs was calculated and is shown in the following plot for the first 4 msec of the initial coupling simulation.

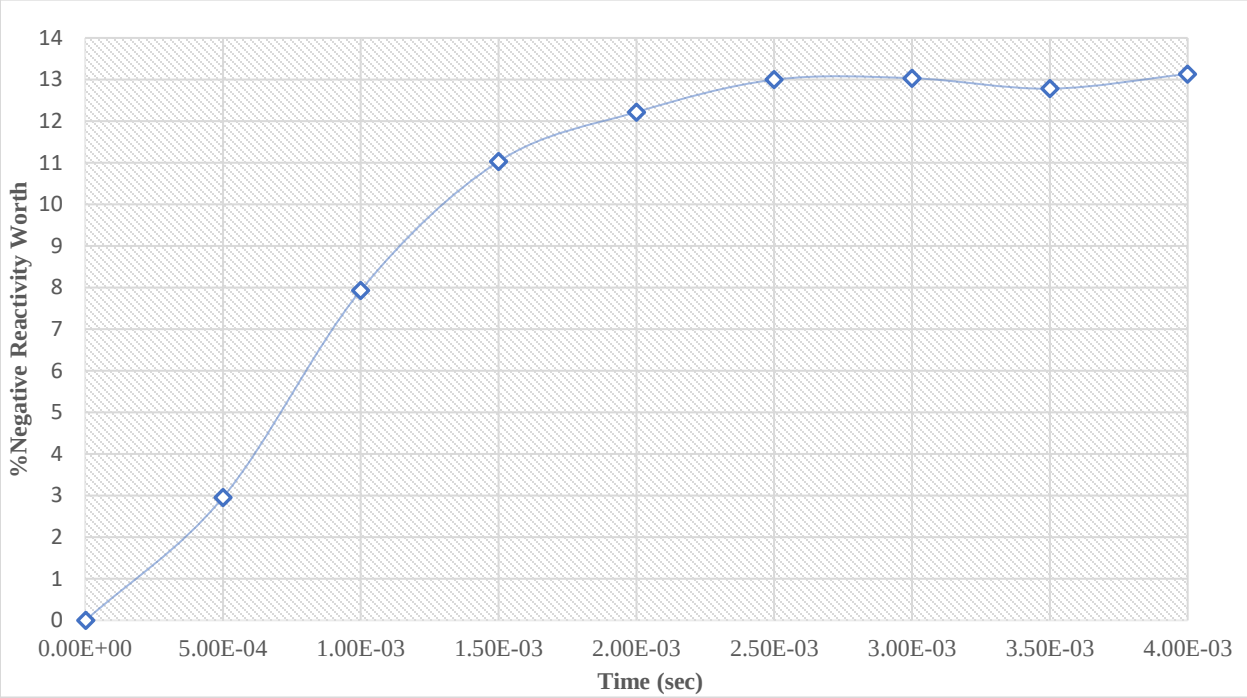


Figure 4.3.15. Negative reactivity worth

From Figure 4.3.15, with four active HENRIs the max amount of negative reactivity inserted into TREAT within 4 msec is 13%. The stated goal of a 5% negative reactivity insertion is reached within 0.75 msec. This gives the HENRI system plenty of margin for operation while clipping pulses.

5. CONCLUSION

The testing of current and future nuclear fuels in light water reactors under postulated RIA accident conditions is of great interest in the current nuclear industry. RIA accidents have the potential of causing significant damage to not only the nuclear fuel but the reactor as a whole as they can release a large amount of particulates into the system. The TREAT facility at INL is well equipped to perform this RIA testing. With TREAT's pulse capability the boundary conditions representative of RIA accidents can be mimicked, however the current pulse widths of TREAT are longer than what is desired. To decrease the pulse width to the desired length of around 40 milliseconds it has been proposed to inject helium-3 gas into a number of evacuated canisters placed into the TREAT core. As helium-3 is a strong neutron absorber it will clip the reactor faster than the current mechanical transient rods that are employed at TREAT. The desired amount of clipping is stated to be 5% negative reactivity within 5 milliseconds. To assess the feasibility of this goal the HENRI facility and a CFD model capable of characterizing the density evolution of the HENRI facility have been built and validated at Oregon State University. However, these two tools alone do not capture the highly coupled phenomena that exists when helium-3 gas is injected into a nuclear reactor. Therefore, a code that can couple this CFD model with a reactor physics simulation of TREAT and thus capture the effect of this coupled phenomena is desired. This coupled code will then be able to provide a more encapsulating best estimate of the effectives and operation of HENRI when it is inserted into the TREAT facility.

The main objective of this study is to develop and verify a coupled code that can accurately simulate the injection of helium-3 gas using four HENRI cartridges into the TREAT core with high confidence. The development took a systematic approach focusing on leveraging work previously done with STAR-CCM+ to model helium-3 HENRI gas dynamics, the creation of a Serpent 2 model of TREAT with HENRI cartridges installed, and the creation of the wrapping code CONSTELATION. The observations gathered in this undertaking are summarized here.

5.1 Observations

From the verification process, the coupling code was executed with a simplified version of the HENRI-TREAT model detailed in Section 3.8. In this process the mass error of the simulation of the CFD model was tracked throughout the execution of simulation and the volumetric heating wattage calculated by Serpent 2 and its implementation into the CFD model was compared and contrasted. From this analysis, it was observed that the mass error in the simulation was negligible. In addition, it was identified that initially the implementation of the wattage values between both Serpent 2 and STAR-CCM+ is fairly converged but as the transient progresses the wattage values begin to oscillate between each time-step. This phenomenon was investigated and resulted in two conclusions. The first conclusion quantified the mesh error between the Serpent 2 and STAR-CCM+ meshes as outlined in Section 3.4. This was found to be .54% with a 0.95% confidence. The second conclusion was that the oscillations seen in the comparison originate from Serpent 2 and are a combined result of not accounting for the complete heating and cooling of materials, the time step of the coupled simulation, and the number of neutrons being simulated per time-step.

From the helium-3 self-shielding study, a number of steady state simulations of the HENRI-TREAT model were performed in order to quantify the effect that the self-shielding phenomena may have on the volumetric heating of helium-3. These simulations varied the overall density of helium-3 within the four HENRI cartridges. From this, two observations were made. The first observation showed that the effect of self-shielding becomes dominant within a HENRI cartridge placed inside TREAT when the overall density of helium-3 is 1 kg/m^3. The second observation was the radial resolution needed within Serpent 2 to capture the effect of self-shielding while executing the coupled code, CONSTELATION. The radial resolution of 5 was found to be a good balance between being able to capture the self-shielding effect and to reduce the statistical uncertainty that occurs with larger resolutions.

The final study of the report was a simulation of the coupled code with realistic thermal-hydraulic boundary conditions for the STAR-CCM+ simulation and the decided upon radial resolution being implemented in the HENRI-TREAT model of Serpent 2. Several observations were made from this simulation. The first being that the average density of all four HENRI cartridges rises above 1 kg/m^3 after 2.5 milliseconds of simulation time, which is half the stated mission time of 5 milliseconds. This leads to the conclusion that self-shielding will be dominant and accounting for

it is necessary for future simulations. In addition to the average density rising above 1 kg/m^3, a distinct difference in the time evolution of density, pressure, and temperature was observed for the two groups of HENRI. This is a result of the different neutron fluxes seen by the two assumed groups of HENRI. It was also observed that the coupled code can capture the shockwave and the changes in density occurring axially within HENRI by studying the axial volumetric heating.

Another observation made in the coupling run was that using four HENRI cartridges pressurized to 3.45 MPa (500 psi) results in the 5% negative reactivity insertion within 5 milliseconds goal being reached. Based on the results shown in Figure 4.3.15 the max percentage of negative reactivity inserted within 4 milliseconds was observed to be 13%, and the 5% goal was reached within 1 millisecond.

5.2 Relevance of Work

The testing of advanced nuclear fuels and materials is expected to take place in the TREAT facility, specifically testing under RIA conditions. This will be accomplished by pulsing using transient rods. However, as TREAT is currently configured these pulses are longer than needed to reproduce the neutronic and thermal-hydraulic conditions. Using the HENRI cartridges to inject helium 3 into TREAT, these pulses can be clipped. This clipping is a result of the large neutron absorption cross section of helium-3, which when present in large quantities within TREAT will cause the power of the reactor to drop by a significant amount, to the point where the desired boundary conditions of postulated RIA accidents can be maintained. The injection of helium-3 into a nuclear reactor is a complex multi-physics process where the gas dynamics of the helium-3 gas and the neutronics of the reactor are highly coupled This complex process can be captured accurately though the use of a coupled code, in which a CFD model of the gas dynamics of HENRI and a reactor physics model of TREAT are connected and able to pass information to one another. Therefore, this study was conducted into order to create a coupled code that can accurately simulate the density evolution of the HENRI cartridges as well as the neutronics of TREAT. The successful verification of this code with a high degree of confidence allows for the code to be used to support the further design of HENRI. With verification the coupled code can be used to analyze the ability of the HENRI cartridges to effectively clip TREAT power pulses and reach the desired 5% negative reactivity insertion in 5 milliseconds goal.

Along with affirming the ability of HENRI cartridges to clip TREAT the coupled code can be used to support the safety analysis review of HENRI. This can be done by using the results for the density, temperature, and pressure within the HENRI cartridge to be modeled and compared to safety standards implemented by the TREAT facility before final deployment.

Beyond just supporting HENRI, the methodology and wrapping code developed and presented in this study can be extended to support other experiments. The coupled code can, with modification, be feasibly expanded to simulate a variety of experiments at the TREAT facility or other reactors.

5.3 Assumptions and Limitations

Throughout this study, several assumptions were made in order to obtain the results presented. One of the main assumptions made during the development of the coupled code was the passing of data between the two codes. Specifically, the density of helium-3 within a HENRI cartridge was not expected to change radially. This allowed for only axial changes in density to be passed by the coupled code and read in by Serpent 2. This assumption was verified by tracking the radial density of HENRI during a STAR-CCM+ simulation and while some variation was observed due to the initial shockwave propagation it was found that the difference in radial density decreases as time progresses. The decrease in error is significant, with an initial difference of 39% at 0.5 msec of simulation time to 1% after 4 msec of simulation time. While the percent difference was deemed appropriate for this study, a further study possibly accounting for the radial change in density could be beneficial in further increasing the accuracy of the simulation.

A similar assumption was made when collecting volumetric heating information from Serpent 2 using detector cards. Serpent 2 detector cards capture information along a cartesian mesh, whereas the mesh of the CFD model is a fine polyhedral mesh. As these meshes do not match up one to one, some amount of error in the implementation of the volumetric heating by the CFD model was expected. This error was found to be low, on the order of 0.56% and thus the assumption is considered valid.

The second major assumption made was the value of the coupled time-step. The value of the coupled time-step was set to a value larger than that of the CFD model, leading to a situation where in the CFD model the volumetric heating is assumed to be constant for the entirety of that coupled time-step. Setting the coupled time-step to that of the CFD model was found to drastically increase

the computational cost of the code. Multiple coupled time-steps were tested before the value of 2E-6 seconds was chosen as it was found to be the best balance of computational cost and accuracy of the solution.

The third major assumption made was that an actual pulse in TREAT need not be simulated. Instead, the power of the HENRI-TREAT model was set to the peak pulse power at the start of the simulation. The HENRI cartridges are expected to be used at the peak of the pulse, and the simulation of a TREAT pulse would significantly increase the already high computational cost of the code. This assumption is conservative in nature as without simulating the pulse during the simulation, the only mechanics for negative reactivity insertion is through the injection of helium-3 gas into TREAT. During a pulse simulation the transient rods would also be inserting some amount of negative reactivity into TREAT and thus cause a larger amount of total negative reactivity to be inserted.

The main limitation encountered was the computational resources available. The computational cost of the coupled code is large, primarily due to the transient simulation of Serpent 2. The results obtained in this study can be improved by using a fine self-shielding radial resolution, detector mesh, and smaller coupled time-step. However, all of these come with an increase in computational cost, and as the computational resources for this study were limited the values presented in this report were assumed to obtain results in a timely fashion. When simulating helium-3 within the CABRI reactor [4], a code that propagated uncertainty through the simulation including changing input parameters was used. This was then coupled with an artificial intelligence software to help determine results without having to re-run simulations. The use of similar codes could be used in further development of the coupled code discussed in this study to reduce the computational cost.

6. FUTURE WORK

There are two major areas of this study that require future analysis. The first being the implementation of a TREAT pulse into the coupling code. As it is presented here, no pulse of TREAT is simulated instead TREAT is assumed to be at the peak power of a pulse at the start of the simulation. Serpent 2 is capable of simulating pulses in reactors, and a preliminary simulation shows it can mimic a TREAT pulse. However, the pulse does not closely match the experimental data available. This is most likely due to not accounting for the heating and cooling of materials. The verification and validation of a TREAT pulse using SERPENT 2 must be performed before it can be implemented into the coupling code presented in this report.

While this study presents the verification of the coupled code, its validation is beyond the scope. This is due to no current experimental data for the fast injection of helium-3 into a nuclear reactor being available to the authors. In order to validate the coupled code and thus the results presented, an experiment using the TRIGA reactor at OSU is being developed. This experiment will involve a small canister of helium-3 which will be placed in front of a neutron beam. This pressure and temperature of the helium-3 will then be measured. A similar setup can then be modeled using the coupled code and the results of the code validated against the experiment.

7. REFERENCES

[1] N. E. Woolstenhulme and J. D. Wiest, "ATF Transient Testing Pre-Conceptual Engineering Status Summary," Idaho National Laboratory, Idaho Falls, 2013.

[2] OECD “Nuclear Fuel Behaviour Under Reactivity-initiated Accident (RIA) conditions.” ISBN 978-92-64-99113-2

[3] D.C Crawford, A.E. Wright, R.W. Swanson, and R. E. Hotlz “RIA Testing Capability of the Transient Reactor Test Facility,” Proceedings of the IAEA Technical Committee Meeting on Fuel Cycle Options for LWRs and HWRS, Victoria, Canada, May 1998, IAEA-TECDOC-1122, pp 99-109.

[4] O. Clamens, J. Lecerf, B. Duc, J-P. Hudelot, : Assessment of the CABRI Transients Power Shape by using CFD and Point Kinetics Codes,” Proc. PHYSOR 2016, Sun Valley, ID (1-5 May 2016).

[5] T. Holschuh, N. Woolstenhulme, B. Baker, J. Bess, C. Davis and J. Parry, "Transient Reactor Test Facility Advanced Transient Shapes," Nuclear Technology, 2019.

[6] C. Folsom, N. Woolstenhulme, T. Shorthill, H. Zhao, J. Bess, C. Davis, L. Dusanter, and J. Parry, "Narrowing Transient Testing Pulse Widths to Enhance LWR RIA Experiment Design in the TREAT Facility," Idaho National Laboratory, February 2019.

[7] S. Balderrama, “Computational Fluid Dynamics Modeling of the Density Evolution Inside of the Helium-3 Enhanced Negative Reactivity Insertion (HENRI) System,” [Master’s thesis Oregon State University].

[8] H. Petersen, “The Properties of Helium: Density, Specific Heats, Viscosity, and Thermal Conductivity at pressures from 1 to 100 bar and from room temperature to about 1800K,” Roskilde, Denmark: Risø National Laboratory. Denmark. Forskningscenter Risoe. Risoe-R, No. 224,0 1970.

[9] F. D'Auria, A. Salah, G. Giorgio, J. Vedovi, F. Reventos, A. Cuadra, J.L. Gago, A. Sjöberg, O. Sandervåg, N. Garis, C. Anhert, J.M. Aragonés, G. Verdú, R. Mirò, D. Ginestar, A.M. Sanchez Hernandez, F. Maggini, J. Hadek, D. Panayotov. “CRISSUE-S – WP2 Neutronics/Thermal-hydraulics Coupling in LWR Technology: State-of-the-art Report (REAC-SOAR)”. (2004). 10.13140/RG.2.1.1087.3765.

[10] Zhang K. “The multiscale thermal-hydraulic simulation for nuclear reactors: A classification of the coupling approaches and a review of the coupled codes.” Int J Energy Res. 2020; 44:3295–3315. https://doi.org/10.1002/er.5111.

[11] Daeubler, Miriam, Aleksandar Ivanov, Bart L. Sjenitzer, Victor Sanchez, Robert Stieglitz, and Rafael Macian-Juan. “High-Fidelity Coupled Monte Carlo Neutron Transport and Thermal-Hydraulic Simulations Using Serpent 2/SUBCHANFLOW.” *Annals of Nuclear Energy* 83 (September 2015): 352–75. https://doi.org/10.1016/j.anucene.2015.03.040.

[12] Erfaninia, Ali, Afshin Hedayat, S.M. Mirvakili, and M.R. Nematollahi. “Neutronic-Thermal Hydraulic Coupling Analysis of the Fuel Channel of a New Generation of the Small Modular Pressurized Water Reactor Including Hexagonal and Square Fuel Assemblies Using MCNP and CFX.” *Progress in Nuclear Energy* 98 (July 2017): 213–27. https://doi.org/10.1016/j.pnucene.2017.03.025.

[13] Seker, Volkan, J.W. Thomas, and T.J. Downar. “Reactor Physics Simulation with Couple Monte Carlo Calculation and Computational Fluid Dynamics,” 2007, 6.

[14] Henry, R., I. Tiselj, and L. Snoj. “CFD/Monte-Carlo Neutron Transport Coupling Scheme, Application to TRIGA Reactor.” *Annals of Nuclear Energy* 110 (December 2017): 36–47. https://doi.org/10.1016/j.anucene.2017.06.018.

[15] J. Hu, U. Rizwan. “Coupled neutronics and thermal-hydraulics simulations using MCNP and FLUENT.” *Trans. Am. Nucl. Soc.* (2008): pp 606-608.

[16] Agung, Alexander, József Bánáti, Mathias Stålek, and Christophe Demazière. “Validation of PARCS/RELAP5 Coupled Codes against a Load Rejection Transient at the Ringhals-3 NPP.” *Nuclear Engineering and Design* 257 (April 2013): 31–44. https://doi.org/10.1016/j.nucengdes.2012.12.023.

[17] L.S. Li, H.M. Yuan, K. Wang. “Coupling of RMC and CFX for analysis of pebble bed-advanced high temperature reactor core” Nucl. Eng. Des., 250 (2012), pp. 385-391

[18] Rao, Junjie, Xiaotong Shang, Ganglin Yu, and Kan Wang. “Coupling RMC and CFD for Simulation of Transients in TREAT Reactor.” *Annals of Nuclear Energy* 132 (October 2019): 249–57. https://doi.org/10.1016/j.anucene.2019.04.039.

[19] Gleicher, Frederick N, Javier Ortensi, Benjamin W Spencer, Yaqi Wang, Stephen R NovasconThe Coupling of the Neutron Transport Application RATTLESNAKE to the Nuclear Fuels Performance Application BISON under the MOOSE Framewrok," n.d., 20.

[20] Darnowski, Piotr, Kacper Potapczyk, Michał Gatkowski, and Grzegorz Niewiński. "Development of One-Way-Coupling Methodology between Severe Accident Integral Code MELCOR and Monte Carlo Neutron Transport Code Serpent." *Procedia Engineering* 157 (2016): 207–13. https://doi.org/10.1016/j.proeng.2016.08.358.

[21] Allelein, H.-J., S. Kasselmann, A. Xhonneux, F. Tantillo, A. Trabadela, and D. Lambertz. "First Results for Fluid Dynamics, Neutronics and Fission Product Behavior in HTR Applying the HTR Code Package (HCP) Prototype." *Nuclear Engineering and Design* 306 (September 2016): 145–53. https://doi.org/10.1016/j.nucengdes.2016.04.028.

[22] R. Henry, I. Tiselj, and L. Snoj. "Transient CFD/Monte-Carlo Neutron Transport Coupling Scheme for Simulation of a Control Rod Extraction in TRIGA Reactor." *Nuclear Engineering and Design* 331 (May 2018): 302–12. https://doi.org/10.1016/j.nucengdes.2018.03.015.

[23] J.J. Duderstadt and L.J. Hamilton, "Nuclear Reactor Analysis", John Wiley and Sons (1976).

[24] E.E. Lewis and W.F. Miller, Jr., "Computational Methods of Neutron Transport", American Nuclear Society, Inc. (1993).

[25] A. Levinsky, V. Valtavirta, F.P. Adams , Anghel V.N.P. "Modeling of the SPERT transients using Serpent 2 with time-dependent capabilities" (2019) *Annals of Nuclear Energy*, 125 , pp. 80-98.

[26] V. Valtavirta, M. Hessan, Leppänen. "Delayed Neutron Emission Model for Time Dependent Simulations with the Serpent 2 Monte Carlo Code-First Results", 2016.

[27] Frederic, Damian & Brun, E... "ORPHEE research reactor: 3D core depletion calculation using Monte-Carlo code Tripoli-4®". Annals of Nuclear Energy (2015). 82. 203-216. 10.1016/j.anucene.2014.08.003.

[28] Nyalunga, Gezekile & Naicker, V.V. & Ivanov, Kostadin. . "Quantification and propagation of neutronics uncertainties of the Kozloduy-6 VVER-1000 fuel assembly using SCALE 6.2.1 within the NEA/OECD benchmark for uncertainty analysis in

modelling of LWRs". Annals of Nuclear Energy (2019). 133. 732-749. 10.1016/j.anucene.2019.07.016.

[29] "Standard for verification and validation in computational fluid dynamics and heat transfer: An American national standard." (2009). New York: American Society of Mechanical Engineers.

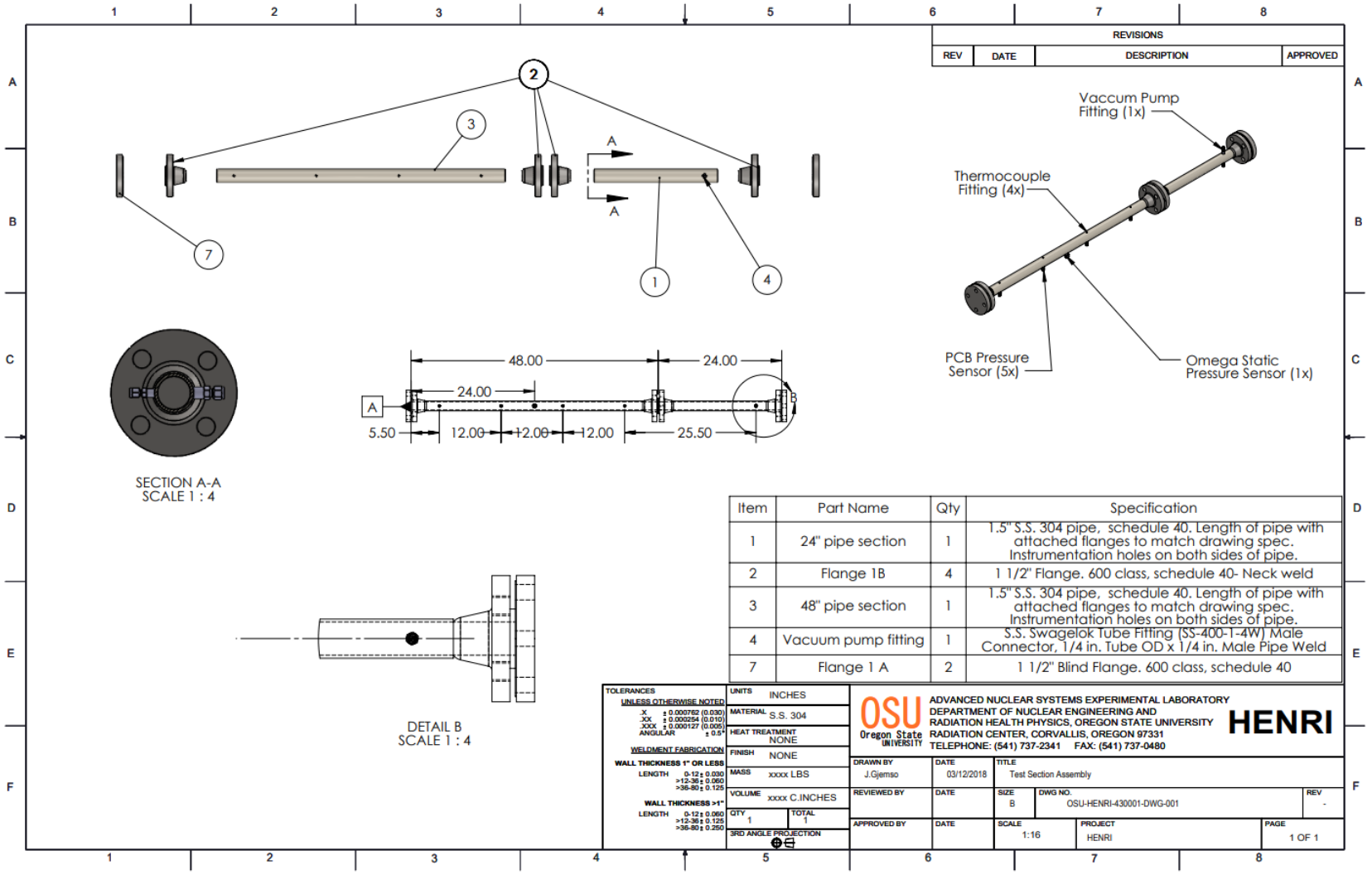
[30] J. Leppänen, M. Pusa, T. Viitanen, V. Valtavirta, and T. Kaltiaisenaho. "The Serpent Monte Carlo code: Status, development and applications in 2013." Ann. Nucl. Energy, **82** (2015) 142-150

[31] Leppänen, J., Pusa, M., Viitanen, T., Valtavirta, V., & Kaltiaisenaho, T. (n.d.). *Coupled multi-physics calculations*. Coupled multi-physics calculations - Serpent Wiki. http://serpent.vtt.fi/mediawiki/index.php/Coupled_multi-physics_calculations.

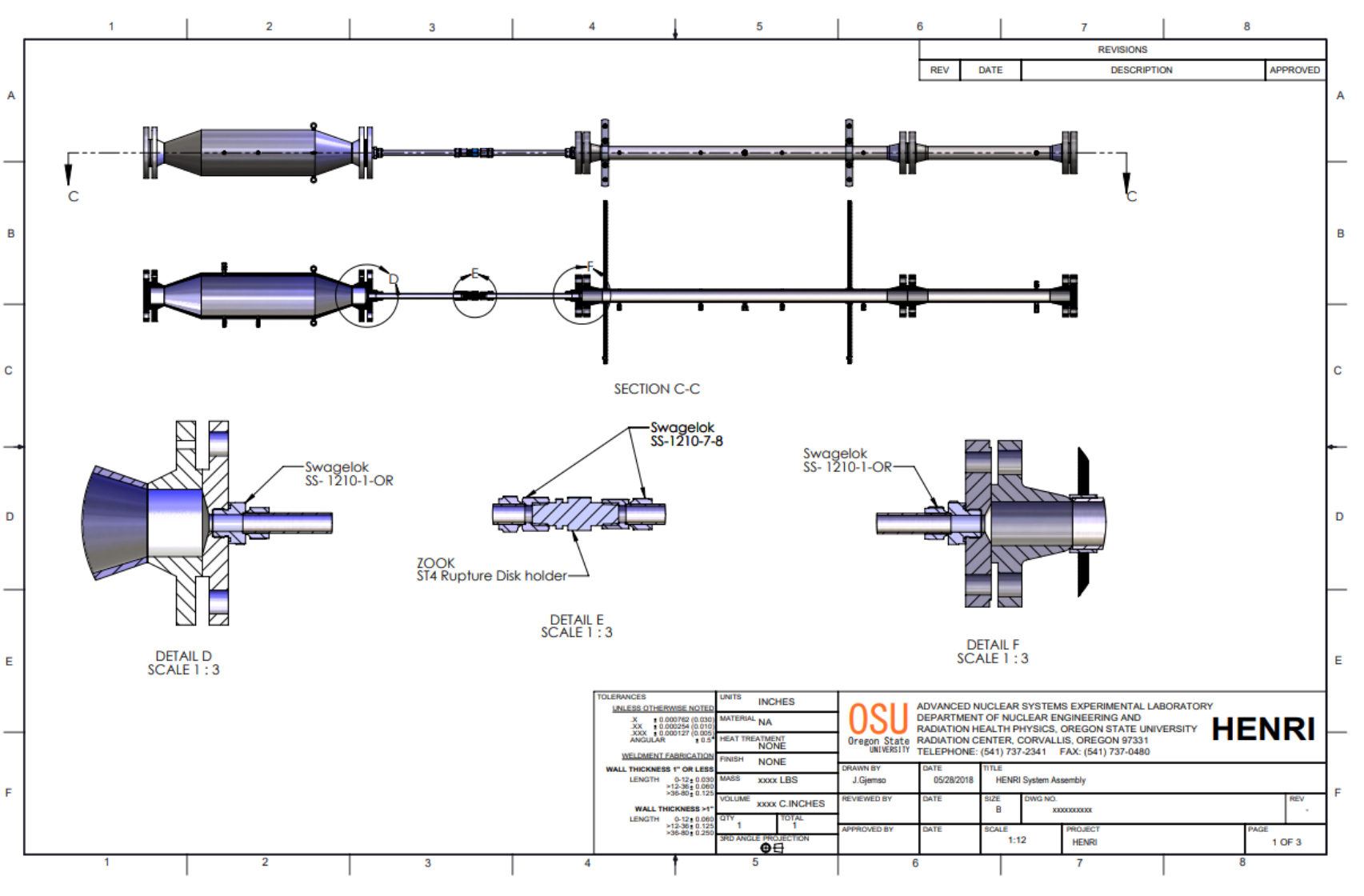
[32] T. Downar, and V. Seker. "Benchmark Specifications for the TREAT Minimum Critical and M8CAL Cores". INL, (September 2016)

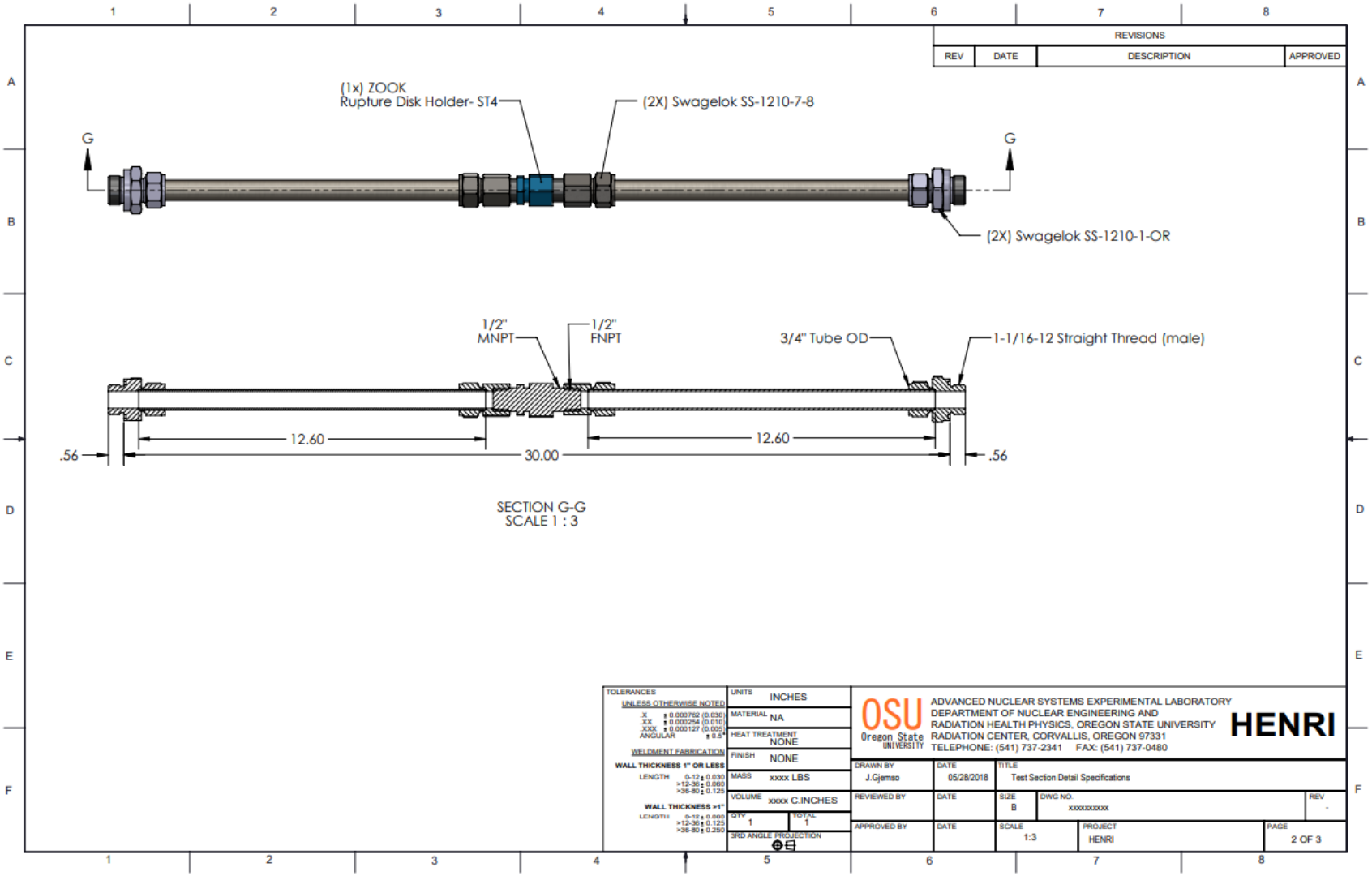
[33] B. Heath "Parametric Thermal Models of the Transient Reactor Test Facility (TREAT)" [Master's Thesis Idaho State University]

Appendix A: HENRI Drawings



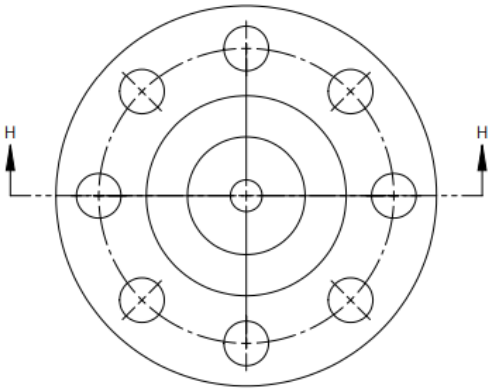
Item	Part Name	Qty	Specification
1	24" pipe section	1	1.5" S.S. 304 pipe, schedule 40. Length of pipe with attached flanges to match drawing spec. Instrumentation holes on both sides of pipe.
2	Flange 1B	4	1 1/2" Flange. 600 class, schedule 40- Neck weld
3	48" pipe section	1	1.5" S.S. 304 pipe, schedule 40. Length of pipe with attached flanges to match drawing spec. Instrumentation holes on both sides of pipe.
4	Vacuum pump fitting	1	S.S. Swagelok Tube Fitting (SS-400-1-4W) Male Connector, 1/4 in. Tube OD x 1/4 in. Male Pipe Weld
7	Flange 1 A	2	1 1/2" Blind Flange. 600 class, schedule 40


REVISIONS
REV
DATE
DESCRIPTION
APPROVED
C
C
D
E
F
SECTION C-C
Swagelok
SS- 1210-1-OR
DETAIL D
SCALE 1 : 3
Swagelok
SS-1210-7-8
ZOOK
ST4 Rupture Disk holder
DETAIL E
SCALE 1 : 3
Swagelok
SS- 1210-1-OR
DETAIL F
SCALE 1 : 3
TOLERANCES
UNLESS OTHERWISE NOTED
WELDMENT FABRICATION
WALL THICKNESS 1" OR LESS
WALL THICKNESS >1"
UNITS INCHES
MATERIAL NA
HEAT TREATMENT NONE
FINISH NONE
MASS xxxx LBS
VOLUME xxxx C.INCHES
QTY 1
TOTAL 1
3RD ANGLE PROJECTION
OSU
Oregon State University
ADVANCED NUCLEAR SYSTEMS EXPERIMENTAL LABORATORY
DEPARTMENT OF NUCLEAR ENGINEERING AND
RADIATION HEALTH PHYSICS, OREGON STATE UNIVERSITY
RADIATION CENTER, CORVALLIS, OREGON 97331
TELEPHONE: (541) 737-2341 FAX: (541) 737-0480
HENRI
DRAWN BY J.Gjemso
DATE 05/28/2018
TITLE HENRI System Assembly
REVIEWED BY
DATE
SIZE B
DWG NO. xxxxxxxxx
REV -
APPROVED BY
DATE
SCALE 1:12
PROJECT HENRI
PAGE 1 OF 3


REVISIONS
REV
DATE
DESCRIPTION
APPROVED
(1x) ZOOK
Rupture Disk Holder- ST4
(2X) Swagelok SS-1210-7-8
G
G
(2X) Swagelok SS-1210-1-OR
1/2"
MNPT
1/2"
FNPT
3/4" Tube OD
1-1/16-12 Straight Thread (male)
.56
12.60
30.00
12.60
.56
SECTION G-G
SCALE 1 : 3
TOLERANCES
UNLESS OTHERWISE NOTED
WELDMENT FABRICATION
WALL THICKNESS 1" OR LESS
WALL THICKNESS >1"
UNITS
INCHES
MATERIAL
NA
HEAT TREATMENT
NONE
FINISH
NONE
MASS
xxxx LBS
VOLUME
xxxx C.INCHES
QTY
1
TOTAL
1
3RD ANGLE PROJECTION
OSU
Oregon State University
ADVANCED NUCLEAR SYSTEMS EXPERIMENTAL LABORATORY
DEPARTMENT OF NUCLEAR ENGINEERING AND
RADIATION HEALTH PHYSICS, OREGON STATE UNIVERSITY
RADIATION CENTER, CORVALLIS, OREGON 97331
TELEPHONE: (541) 737-2341 FAX: (541) 737-0480
HENRI
DRAWN BY
J.Gjemso
DATE
05/28/2018
TITLE
Test Section Detail Specifications
REVIEWED BY
DATE
SIZE
B
DWG NO.
xxxxxxxxx
REV
APPROVED BY
DATE
SCALE
1:3
PROJECT
HENRI
PAGE
2 OF 3

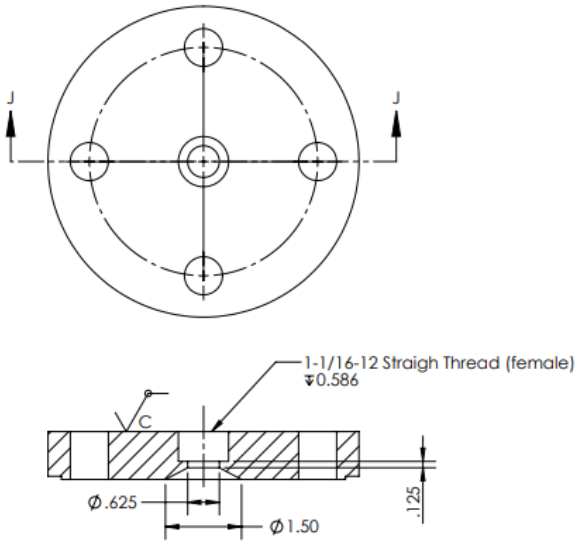
REVISIONS			
REV	DATE	DESCRIPTION	APPROVED

FLANGE T1



SECTION H-H

FLANGE T2



SECTION J-J

6.3
3.2 C — Smooth finish only on area covered by a radius (r= 0.875) of all selected flange faces

TOLERANCES
UNLESS OTHERWISE NOTED
.X ± 0.000762 (0.030)
.XX ± 0.000254 (0.010)
.XXX ± 0.000127 (0.005)
ANGULAR ± 0.5°
WELDMENT FABRICATION
WALL THICKNESS 1" OR LESS
LENGTH 0-12 ± 0.030, >12-36 ± 0.060, >36-80 ± 0.125
WALL THICKNESS >1"
LENGTH 0-12 ± 0.060, >12-36 ± 0.125, >36-80 ± 0.255

UNITS	INCHES
MATERIAL	NA
HEAT TREATMENT	NONE
FINISH	NONE
MASS	xxxx LBS
VOLUME	xxxx C.INCHES
QTY	1
TOTAL	1
3RD ANGLE PROJECTION	

OSU Oregon State University

ADVANCED NUCLEAR SYSTEMS EXPERIMENTAL LABORATORY
DEPARTMENT OF NUCLEAR ENGINEERING AND
RADIATION HEALTH PHYSICS, OREGON STATE UNIVERSITY
RADIATION CENTER, CORVALLIS, OREGON 97331
TELEPHONE: (541) 737-2341 FAX: (541) 737-0480

HENRI

DRAWN BY	DATE	TITLE			
J.Gjemso	05/28/2018	Flange T1 & T2 Specification			
REVIEWED BY	DATE	SIZE	DWG NO.		REV
		B	xxxxxxxxxx		-
APPROVED BY	DATE	SCALE	PROJECT	PAGE	
		1:2	HENRI	3 OF 3	

Appendix B: STAR-CCM+ JavaScript macro and CONSTELATION Script

Table 4.3.1. STAR-CCM+ JavaScript macro

```
// Simcenter STAR-CCM+ macro: load_data.java
// Written by Simcenter STAR-CCM+ 15.04.010
package macro;

import java.io.File;
import java.util.*;
import star.common.*;
import star.base.neo.*;
import star.meshing.*;
import star.energy.*;
import java.io.IOException;
import java.io.FileWriter;

public class load_dataBot extends StarMacro {

  public void execute() {
// Starts the STAR-CCM+ simulation
    execute0();
  }
```

```
  private void execute0() {
// Remeshes problem to ensure start from scratch
    Simulation simulation_0 =
      getActiveSimulation();

    MeshPipelineController meshPipelineController_0 =
      simulation_0.get(MeshPipelineController.class);

    meshPipelineController_0.clearGeneratedMeshes();

    meshPipelineController_0.generateVolumeMesh();

    Solution solution_0 =
      simulation_0.getSolution();

// Creates Table from provided .csv from the wrapping code
    FileTable fileTable_2 =
      (FileTable) simulation_0.getTableManager().createFromFile(resolvePath("STAR_HeatBot.csv"));

// Set Volumetric Heating input to use the table created from wrapping code
    Region region_0 =
```

```
    simulation_0.getRegionManager().getRegion("3.29.2019_Benchmark_1in");

   VolumetricHeatSourceProfile volumetricHeatSourceProfile_0 =
    region_0.getValues().get(VolumetricHeatSourceProfile.class);

   volumetricHeatSourceProfile_0.setMethod(XyzTabularScalarProfileMethod.class);

   volumetricHeatSourceProfile_0.getMethod(XyzTabularScalarProfileMethod.class).setTable(fileTable_2);

 // Initializes the simulation
   solution_0.initializeSolution();
// Grabs the user-set max stopping time
  PhysicalTimeStoppingCriterion physicalTimeStoppingCriterion_0 =
    ((PhysicalTimeStoppingCriterion) simulation_0.getSolverStoppingCriterionManager().getSolverStoppingCriterion("Maximum Physical
Time"));
// Grabs the current simulation time (should be 0.0 sec)
double startTimelevel = simulation_0.getSimulationIterator().getCurrentTimeLevel();

// Changes the number of time steps that one initalization of the STEP command performs
   simulation_0.getSimulationIterator().setNumberOfSteps(100);
// Sets the Max Stopping time to a number so it can be compared to current sim time
double maxstoppingtime = physicalTimeStoppingCriterion_0.getMaximumTime().getValue();
```

```
double TotalTimeSteps;
TotalTimeSteps = 100000000;
File f;
int sleep_time;
int wait_time;
int break_again;
wait_time = 100000000;
sleep_time = 0;
break_again = 0;
// While Loop dicating the refreshing of the Heating Table provided by the wrapping code
 double Current_Time;
 Current_Time = 0;
while (TotalTimeSteps > Current_Time)
{
    simulation_0.getSimulationIterator().step(100);
    sleep_time = 0;
//Create file says STAR-CCM+ is done simulating
   try {
       FileWriter writer = new FileWriter("STARBotDone.txt", true);
       writer.write("Hello World");
       writer.write("\r\n");   // write new line
       writer.write("Good Bye!");
```

```
        writer.close();
    } catch (IOException e) {
            e.printStackTrace();
    }


    // Check to see if wrapping code says Serpent 2 is done executing this step
    f = new File("SerpentDone.txt");
    boolean exist = f.exists();
// If it doesnt exist (pause execution for a second and check again
    while (!f.exists())
    {
     simulation_0.println("Could not find done file pausing...");
       try {
       Thread.sleep(1000);
       }
       catch(InterruptedException ex)
       {
        Thread.currentThread().interrupt();
       }
       sleep_time = sleep_time + 1;
// If file isn't created in specified wait time, break out of loops
        if (sleep_time > wait_time)
```

```
        {
          simulation_0.println("Breaking out of Running Loop");
             break_again = break_again+1;
           break;
        }
    }
// If it does exist then just reload and continue simulation
    if (f.exists())
    {
    fileTable_2.extract();
    }
// Breaks out of execution loop
    if (break_again > 0)
    {
    break;
    }
    Current_Time = simulation_0.getSimulationIterator().getCurrentTimeLevel();
    simulation_0.println("Current Time Step is :" + Current_Time);
  }
 }
}
```

Table 4.3.2. CONSTELATION Python Script

```
############################################################
#                                                          #
#         STAR and Serpent Coupling Script v 1.0           #
#                 CONSTELLATION                            #
#                                                          #
# Created by: Cole Leingang              2020/06/10        #
#                                                          #
############################################################

import os
import signal
import math
import time
import csv
import pandas as pd
import numpy as np
import re
from shutil import copyfile
# timestep used for simulation
timestep = 2E-6
# The number of time steps that STAR will simulate before checking for SERPENT completion and then export Data
STAR_STEP = 40
# Second variables used to stay constant in loop
step_length = 40
#########################################################
# Create the Serpent input-file for this run          #
# (process id or communication file must be appended) #
#########################################################

# Open original input for reading

file_in = open(r'Treat', 'r')
```

```
# Open a new input file for writing

file_out = open(r'coupledTreat', 'w')

# Write original input to new file

for line in file_in:
    file_out.write(line)

# Close original input file

file_in.close()
# Append Source File Location
file_out.write('\n')
file_out.write('set dynsrc Source 1\n')

# Do not make group constants
file_out.write('\n')
file_out.write('set gcu -1\n')

# Append signalling mode

file_out.write('\n')
file_out.write('set comfile com.in com.out\n')

# Append interface names

file_out.write('\n')
file_out.write('ifc HE3TOP.ifc\n\n')
file_out.write('\n')
file_out.write('ifc HE3BOT.ifc\n\n')
file_out.write('\n')
```

```
file_out.write('ifc fuel.ifc\n\n')

# Close new input file

file_out.close()

################################################
# Write the initial He3 interface file for 1st Star Run                    #
# (He3 temperature and density for top two HENRI's will be updated)        #
################################################

file_out = open('HE3TOP.ifc', 'w')

# Write the header line (TYPE MAT OUT)

file_out.write('2  He3Top 0\n')

# Write the mesh type

file_out.write('1\n')

# Write the mesh data (NX XMIN XMAX NY YMIN YMAX NZ ZMIN ZMAX)

file_out.write('1 -100 100 1 0 100 500 -60.48375 60.48375\n')

# Write the Mesh Data
STAR_Points = 501
# Check to see if STAR-CCM_ .csv file has been created and wait for a predetermined amount of time if it hasn't
time_to_wait = 10
time_counter = 0
filename = r'./ExtractedData/He3Data_table.csv'
while not os.path.exists(filename):
   time.sleep(1)
```

```
    time_counter += 1
    if time_counter > time_to_wait: break
if os.path.isfile(filename):
    # Take the data from STAR-CCM+ .csv file and write to ifc
    columns = ['Position in Cartesian 1[X] (cm)', 'Density(g/cm^3) (kg/m^3)', 'Temperature (K)']
    df = pd.read_csv(filename, usecols=columns)
    # Position = df['Position in Cartesian 1[X] (cm)']
    # Density =  df['Density(g/cm^3) (kg/m^3)']
    # Temp = df['Temperature (K)']

    # Sorts data from STAR-CCM+ in ascending order by position
    df_convert = df.to_numpy()
    numpy_array = df_convert[df_convert[:, 0].argsort()]

    # Cross Section Data for He-3 in SERPENT2 only exists for 300 K or above, this will only happen early on in the transient and temp
    # should never fall that far below 300K so setting this arbitrary constraint should have minimal change in results
    for i in range(STAR_Points):
        if numpy_array[i, 2] < 300.0:
            numpy_array[i, 2] = 300.0
    denstemp = numpy_array[:, [1, 2]]
    np.savetxt(file_out, denstemp, fmt="%1.6f")
else:
    raise ValueError("%s has not been created or could not be read" % filename)
# Close interface file

file_out.close()

################################################
# Write the initial He3 interface file for 2nd Star Run                    #
# (He3 temperature and density for bottom two HENRI's will be updated)     #
################################################

file_out = open('HE3BOT.ifc', 'w')
```

```
# Write the header line (TYPE MAT OUT)

file_out.write('2  He3Bot 0\n')

# Write the mesh type

file_out.write('1\n')

# Write the mesh data (NX XMIN XMAX NY YMIN YMAX NZ ZMIN ZMAX)

file_out.write('1 -100 100 1 -100 0 500 -60.48375 60.48375\n')

# Write the Mesh Data
STAR_Points = 501
# Check to see if STAR-CCM_ .csv file has been created and wait for a predetermined amount of time if it hasn't
time_to_wait = 10
time_counter = 0
filename = r'./ExtractedData/He3Data_table.csv'
while not os.path.exists(filename):
    time.sleep(1)
    time_counter += 1
    if time_counter > time_to_wait: break
if os.path.isfile(filename):
    # Take the data from STAR-CCM+ .csv file and write to ifc
    columns = ['Position in Cartesian 1[X] (cm)', 'Density(g/cm^3) (kg/m^3)', 'Temperature (K)']
    df = pd.read_csv(filename, usecols=columns)
    # Position = df['Position in Cartesian 1[X] (cm)']
    # Density =  df['Density(g/cm^3) (kg/m^3)']
    # Temp = df['Temperature (K)']

    # Sorts data from STAR-CCM+ in ascending order by position
    df_convert = df.to_numpy()
```

```
    numpy_array = df_convert[df_convert[:, 0].argsort()]

    # Cross Section Data for He-3 in SERPENT2 only exists for 300 K or above, this will only happen early on in the transient and temp
    # should never fall that far below 300K so setting this arbitrary constraint should have minimal change in results
    for i in range(STAR_Points):
        if numpy_array[i, 2] < 300.0:
            numpy_array[i, 2] = 300.0
    denstemp = numpy_array[:, [1, 2]]
    np.savetxt(file_out, denstemp, fmt="%1.6f")
else:
    raise ValueError("%s has not been created or could not be read" % filename)
# Close interface file

file_out.close()

################################################
# Write the initial fuel interface           #
################################################

file_out = open('fuel.ifc', 'w')

# Write the header line (TYPE MAT OUT)

file_out.write('2  fuel1 0\n')

# Write the mesh type

file_out.write('1\n')

# Write the mesh data (NX XMIN XMAX NY YMIN YMAX NZ ZMIN ZMAX)

file_out.write('1 -200 200 1 -200 200 10 -60.48375 60.48375\n')
```

```
# Write initial fuel temperatures and densities
# TREAT has graphite mixed fuel (hence low density close to graphite)

for i in range(10):
    file_out.write('-1.72 300.0\n')
    if i == 7:
        file_out.write('-1.72 330.0\n')

# Close interface file

file_out.close()

##########################################
# Initialize the fuel temperature solution #
##########################################

TBOI = []
TEOI = []

for i in range(10):
    TBOI.append(300.0)
    TEOI.append(300.0)

################################
# Start the Serpent simulation #
################################

# Submit SERPENT2 submission script to server
run_SERP = "qsub SERPENT_job.sh"
os.system(run_SERP)


# Reset time step
```

```
curtime = 0
# Pause Simulation unitl SERPENT2 starts simulating
SERPENTWait = 500000000
time_counter = 0
Serpname = r'com.out'
while not os.path.exists(Serpname):
    time.sleep(5)
    time_counter += 1
    if time_counter > SERPENTWait:
        raise ValueError("%s has not been created or could not be read" % Serpname)
        break
########################
# Loop over time steps #
########################

simulating = 1

while simulating == 1:
    ####################
    # Wait for signal #
    ####################
    sleeping = 1
    while sleeping == 1:
        # Sleep for two seconds
        time.sleep(5)
        # Open file to check if we got a signal
        fin = open('com.out', 'r')
        # Read line
        line = fin.readline()
        # Close file
        fin.close()
        # Check signal
```

```
        if int(line) != -1:
            if int(line) == signal.SIGUSR1:
                # Got the signal to resume
                print(signal.SIGUSR1)
                print("Resume Current Iteration")
                sleeping = 0
            elif int(line) == signal.SIGUSR2:
                # Got the signal to move to next time point
                print(signal.SIGUSR2)
                print('Move to Next Time Step')
                iterating = 0
                sleeping = 0
            elif int(line) == signal.SIGTERM:
                # Got the signal to end the calculation
                print(signal.SIGTERM)
                print('END The Simulation')
                iterating = 0
                sleeping = 0
                simulating = 0
            else:
                # Unknown signal
                print("\nUnknown signal read from file, exiting\n")
                print(line)
                # Exit
                quit()
            # Reset the signal in the file
            file_out = open('com.out', 'w')
            file_out.write('-1')
            file_out.close()
    # Check if simulation has finished and break out of iterating
    # loop
    if simulating == 0:
        break
```

```
    ########################
    # Import SERPENT2 Data ##
    ########################

    # First Convert given SERPENT2 output of .m file to .txt file to be manipulated
    outputfile = r'coupledTreat_det'+str(curtime) + '.m'
    textfile = r'coupledTreat_det' + str(curtime) + '.txt'
    time_counter = 0
    while not os.path.exists(outputfile):
        time.sleep(1)
        time_counter += 1
    if time_counter > SERPENTWait:
        raise ValueError("%s has not been created or could not be read" % Serpname)
        break
    copyfile(outputfile, textfile)
    ##########################################
    # Collect Data from converted output file #
    ##########################################

    # if copying error has occured just replace with previous timestep (band-aid fix)
    if os.stat(textfile).st_size == 0:
        name = r'coupledTreat_det'+str(curtime-1)
        outputfile = r'coupledTreat_det'+str(curtime-1) + '.m'
        textfile = r'coupledTreat_det' + str(curtime-1) + '.txt'
        copyfile(outputfile, textfile)
        # print to outfile to keep track of copy errors
        print('Copy Error at TimeStep = ' + str(curtime))
    else:
        name = r'coupledTreat_det'+str(curtime)
    # Finds Heat Production Detector values from SERPENT2 Output
    HeadTop_pattern = re.compile(r"\DETSerpent2STop\s")
    HeadBot_pattern = re.compile(r"\DETSerpent2SBot\s")
    # Finds Fuel Deposition Detector values from SERPENT2 Output
```

```
    Fuel_pattern = re.compile(r"\DETFuelDepositon\s")
    # Finds data after initial pattern has been found
    data_pattern = re.compile(r'(\d+)\s*(\d+)\s*(.*)$')
    # Finds Heat Production Detector Z values
    Second_pattern = re.compile(r"DETSerpent2STopZ(.*)$")
    # Second_pattern =  re.compile(r"DETSerpent2SBotZ(.*)$")
    # Number of Z points
    Zpoints = 100
    # Number of Fuel points
    Fpoints = 10
    # Finds Heat Production Detector X values
    Third_pattern = re.compile(r"DETSerpent2STopX(.*)$")
    # ThirdBot_pattern =  re.compile(r"DETSerpent2SBotX(.*)$")
    # Number of X points
    Xpoints = 1
    # Finds Heat Production Detector Y values
    FourthTop_pattern = re.compile(r"DETSerpent2STopY(.*)$")
    FourthBot_pattern = re.compile(r"DETSerpent2SBotY(.*)$")
    # Number of Y points
    Ypoints = 5
    # Number of overall points
    points = Zpoints*Ypoints*Xpoints

    def SERPENTExtract(F1, f2):
        global data
        global data_pass
        global datafuel
        global data_fuelpass
        global data_passBot
        global dataBot
        global YdataBot
        global Xdata
        global Ydata
```

```
        global Zdata
        global Fdata
        data_pass = []
        data = []
        data_passBot = []
        dataBot = []
        data_fuelpass = []
        datafuel = []
        Xdata = []
        Ydata = []
        YdataBot = []
        Zdata = []
        for line in f1:
         matchx = Third_pattern.search(line)
         matchyTop = FourthTop_pattern.search(line)
         matchyBot = FourthBot_pattern.search(line)
         matchz = Second_pattern.search(line)
         matchTop = HeadTop_pattern.match(line)
         matchBot = HeadBot_pattern.match(line)
         matchfuel = Fuel_pattern.match(line)
         # Finds Volumetric Heating for Top 2 HENRIs
         if matchTop is not None:
            iter = 0
            for line in f1:
               matchpoints = data_pattern.search(line)
               if matchpoints is not None:
                  savepoints = matchpoints.group(0)
                  if iter < 10:
                     # Save Data used to input x,y,z locations
                     cleanpoints = savepoints[34:48]
                     # Save Mean Value of Data
                     cleanpass = savepoints[48:60]
                     data.append(cleanpoints)
```

```
                data_pass.append(cleanpass)
            if iter < 999 and iter >= 10:
                # Save Data used to input x,y,z locations
                cleanpoints = savepoints[35:48]
                # Save Mean Value of Data
                cleanpass = savepoints[48:61]
                data.append(cleanpoints)
                data_pass.append(cleanpass)
            if iter >= 999:
                # Save Data used to input x,y,z locations
                cleanpoints = savepoints[34:50]
                # Save Mean Value of Data
                cleanpass = savepoints[51:63]
                data.append(cleanpoints)
                data_pass.append(cleanpass)
            iter = iter +1
            if iter == points:
                break
    # Finds and saves fuel deposition values
    if matchfuel is not None:
        iter = 0
        for line in f1:
            matchpoints = data_pattern.search(line)
            if matchpoints is not None:
                savepoints = matchpoints.group(0)
                if iter < 10:
                    # Save Data used to input x,y,z locations
                    cleanpoints = savepoints[34:48]
                    # Save Mean Value of Data
                    cleanpass = savepoints[48:60]
                    datafuel.append(cleanpoints)
                    data_fuelpass.append(cleanpass)
                if iter < 999 and iter >= 10:
```

```
                # Save Data used to input x,y,z locations
                cleanpoints = savepoints[35:48]
                # Save Mean Value of Data
                cleanpass = savepoints[48:61]
                datafuel.append(cleanpoints)
                data_fuelpass.append(cleanpass)
            if iter >= 999:
                # Save Data used to input x,y,z locations
                cleanpoints = savepoints[34:50]
                # Save Mean Value of Data
                cleanpass = savepoints[51:63]
                datafuel.append(cleanpoints)
                data_fuelpass.append(cleanpass)
            iter = iter +1
            if iter == Fpoints:
                break
    # Finds Volumetric Heating for Bottom 2 HENRIs
     if matchBot is not None:
        iter = 0
        for line in f1:
            matchpointsBot = data_pattern.search(line)
            if matchpointsBot is not None:
                savepointsBot = matchpointsBot.group(0)
                if iter < 10:
                    # Save Data used to input x,y,z locations
                    cleanpointsBot = savepointsBot[34:48]
                    # Save Mean Value of Data
                    cleanpassBot = savepointsBot[48:60]
                    dataBot.append(cleanpointsBot)
                    data_passBot.append(cleanpassBot)
                if iter < 999 and iter >= 10:
                    # Save Data used to input x,y,z locations
                    cleanpointsBot = savepointsBot[35:48]
```

```
                # Save Mean Value of Data
                cleanpassBot = savepointsBot[48:61]
                dataBot.append(cleanpointsBot)
                data_passBot.append(cleanpassBot)
            if iter >= 999:
                # Save Data used to input x,y,z locations
                cleanpointsBot = savepointsBot[34:50]
                # Save Mean Value of Data
                cleanpassBot = savepointsBot[51:63]
                dataBot.append(cleanpointsBot)
                data_passBot.append(cleanpassBot)
            iter = iter +1
            if iter == points:
                break
    if matchx is not None:
        iterx = 0
        for line in f1:
            matchx2 = data_pattern.search(line)
            if matchx2 is not None:
                savex = matchx2.group(0)
                cleanx = savex[24:]
                Xdata.append(cleanx)
                iterx = iterx +1
                if iterx == Xpoints:
                    break
    if matchyTop is not None:
        itery = 0
        for line in f1:
            matchy2 = data_pattern.search(line)
            if matchy2 is not None:
                savey = matchy2.group(0)
                cleany = savey[24:]
                Ydata.append(cleany)
```

```
                itery = itery +1
                if itery == Ypoints:
                    break
        if matchyBot is not None:
            itery = 0
            for line in f1:
                matchyBot = data_pattern.search(line)
                if matchyBot is not None:
                    saveyBot = matchyBot.group(0)
                    cleanyBot= saveyBot[24:]
                    YdataBot.append(cleanyBot)
                    itery = itery +1
                    if itery == Ypoints:
                        break
        if matchz is not None:
            iterz = 0
            for line in f1:
                matchz2 = data_pattern.search(line)
                if matchz2 is not None:
                    savez = matchz2.group(0)
                    cleanz = savez[23:]
                    Zdata.append(cleanz)
                    iterz = iterz +1
                    if iterz == Zpoints:
                        break
    if __name__ == '__main__':
        with open(name+'.txt', 'r') as f1:
         with open('STAR_Heat', 'wb') as f2:
            SERPENTExtract(f1,f2)
    ############################################################
    #### Print Data to CSV in format recognized by STAR-CCM+ #
    ############################################################
    Np = range(points)
```

```
nx = range(1,Xpoints+1)
ny = range(1,Ypoints+1)
nz = range(1,Zpoints+1)
data_pass = [float(i) for i in data_pass]
data_passBot = [float(i) for i in data_passBot]
Xdata = [float(i) for i in Xdata]
Ydata = [float(i) for i in Ydata]
YdataBot = [float(i) for i in YdataBot]
Zdata = [float(i) for i in Zdata]
# saves fuel data
data_fuelpass = [float(i) for i in data_fuelpass]
# Converts Data to be written to CSV
for point in Np:
 # Sets first and last set of Volumetric Heating to Zero, since this will define the Volumetric Heating Test Section
 if point <= Ypoints:
     data_pass[point] = data_pass[point]*0
     data_passBot[point] = data_passBot[point]*0
 elif point >= points-Ypoints:
     data_pass[point] = data_pass[point]*0
     data_passBot[point] = data_passBot[point]*0
 else:
 # Converts Mean Values of MeV/cm^3 to J/m^3-s to pass to STAR-CCM+
     data_pass[point] = (data_pass[point]*1E6)/timestep
     data_passBot[point] = (data_passBot[point]*1E6)/timestep
# Converts cm to m
for xpoint in nx:
 # Note: Due to STAR-CCM+ being a 2-D simulation, X-Values (which would be Z Values) are set to zero
 Xdata[xpoint-1] = (Xdata[xpoint-1]-20.405)*0
for ypoint in ny:
    # Y-Values stay the same in both codes (subtract SERPENT Distance from Origin to get STAR-CCM+ relative distance)
    Ydata[ypoint-1] = -1*(Ydata[ypoint-1]-40.605)/100
    # Bottom data uses "negative" position values
    YdataBot[ypoint-1] = -1*(YdataBot[ypoint-1]-40.605)/100
```

```
    for zpoint in nz:
     # Note: Due to orientation of STAR-CCM+ simulation Z Values are X Values in STAR
     Zdata[zpoint-1] = (Zdata[zpoint-1]/-100)+2.267903
    # Organizes Data to be passed to csv
    # Passes Top Data
    with open(r'STAR_HeatTop.csv', 'wb') as f2:
        # Sets up Title Headers for STAR-CCM+
        Title = ['X(m)', 'Y(m)', 'Z(m)', 'VolumetricHeat(W/m^3)']
        csv_writer = csv.writer(f2)
        csv_writer.writerow(Title)
    # Iterates through the number of points given in the SERPENT Output
        for point in Np:
         # Finds the integers used by SERPENT to show Z,Y,X Locations
         temp = re.findall(r'\d+', data[point])
         res = list(map(int,temp))
         # Replaces X integer with actual location from SERPENT Output
         for xpoint in nx:
            if res[2] == xpoint:
                res[2] = Xdata[xpoint-1]
         # Replaces Y integer with actual location from SERPENT Output
         for ypoint in ny:
            if res[1] == ypoint:
                res[1] = Ydata[ypoint-1]
         # Replaces Z Integer with actual location from SERPENT Output
         for zpoint in nz:
            if res[0] == zpoint:
                res[0] = Zdata[zpoint-1]
         # Adds Mean Value for that location
         res.append(data_pass[point])
         # Writes to csv
         csv_writer.writerow(res)
    # Passes Bottom Data
    with open(r'STAR_HeatBot.csv', 'wb') as f2:
```

```
        # Sets up Title Headers for STAR-CCM+
        Title = ['X(m)', 'Y(m)', 'Z(m)', 'VolumetricHeat(W/m^3)']
        csv_writer = csv.writer(f2)
        csv_writer.writerow(Title)
    # Iterates through the number of points given in the SERPENT Output
        for point in Np:
         # Finds the integers used by SERPENT to show Z,Y,X Locations
         temp = re.findall(r'\d+', data[point])
         res = list(map(int,temp))
         # Replaces X integer with actual location from SERPENT Output
         for xpoint in nx:
            if res[2] == xpoint:
               res[2] = Xdata[xpoint-1]
         # Replaces Y integer with actual location from SERPENT Output
         for ypoint in ny:
            if res[1] == ypoint:
               res[1] = YdataBot[ypoint-1]
         # Replaces Z Integer with actual location from SERPENT Output
         for zpoint in nz:
            if res[0] == zpoint:
               res[0] = Zdata[zpoint-1]
         # Adds Mean Value for that location
         res.append(data_passBot[point])
         # Writes to csv
         csv_writer.writerow(res)
        time_counter = 0
    if curtime == 0:
        ################################################
        # Setup the STAR-CCM+ Simulation               #
        ################################################
        # Simply submits STAR-CCM+ submission script to server
        run_STAR1 = "qsub STARTop_Job.sh"
        run_STAR2 = "qsub STARBot_Job.sh"
```

```
    os.system(run_STAR1)
    os.system(run_STAR2)
if curtime > 0:
    # Write SERPENTDone.txt file indicating that the current loop has been completed and data extracted
    file_out = open('./SerpentDone.txt','w')
    file_out.write('Done')
    file_out.close
    time.sleep(60)
    os.remove('SerpentDone.txt')

# check to see if STAR is done executing
STARBot = r'./STARBotDone.txt'
STARTop = r'./STARTopDone.txt'
time_to_wait = 1000000
time_counter = 0
while not os.path.exists(STARTop):
    time.sleep(1)
    time_counter += 1
    if time_counter > time_to_wait:break
while not os.path.exists(STARBot):
    time.sleep(1)
    time_counter += 1
    if time_counter > time_to_wait:break
############################
# Update Top interface     #
############################
os.remove('HE3TOP.ifc')
file_out = open('HE3TOP.ifc', 'w')

file_out.write('2  He3Top 0\n')

# Write the mesh type
```

```
    file_out.write('1\n')

    # Write the mesh data (NX XMIN XMAX NY YMIN YMAX NZ ZMIN ZMAX)

    file_out.write('1 -100 100 1 0 100 500 -60.48375 60.48375\n')

    # Check to see if STAR-CCM_ .csv file has been created and wait for a predetermined amount of time if it hasn't
    time_to_wait = 1000000
    time_counter = 0
    filename = r'./ExtractedData/He3Data_table_'+str(STAR_STEP)+'.csv'
    while not os.path.exists(filename):
        time.sleep(1)
        time_counter += 1
        if time_counter > time_to_wait:break
    if os.path.isfile(filename):
        # Take the data from STAR-CCM+ .csv file and write to ifc
        columns = ['Position in Cartesian 1[X] (cm)', 'Density(g/cm^3) (kg/m^3)', 'Temperature (K)']
        df = pd.read_csv(filename, usecols=columns)
        # Position = df['Position in Cartesian 1[X] (cm)']
        # Density =  df['Density(g/cm^3) (kg/m^3)']
        # Temp = df['Temperature (K)']

        # Sorts data from STAR-CCM+ in ascending order by position
        df_convert = df.to_numpy()
        numpy_array = df_convert[df_convert[:, 0].argsort()]

        # Cross Section Data for He-3 in SERPENT2 only exists for 300 K or above, this will only happen early on in the transient and temp
        # should never fall that far below 300K so setting this arbitrary constraint should have minimal change in results
        for i in range(STAR_Points):
            if numpy_array[i,2] < 300.0:
                numpy_array[i,2] = 300.0
        denstemp = numpy_array[:,[1,2]]
        np.savetxt(file_out,denstemp, fmt = "%1.6f")
```

```
    else:
        raise ValueError("%s has not been created or could not be read" % filename)
    # Close interface file

    file_out.close()
    ############################
    # Update Bottom interface  #
    ############################
    os.remove('HE3BOT.ifc')
    file_out = open('HE3BOT.ifc', 'w')

    file_out.write('2  He3Bot 0\n')

    # Write the mesh type

    file_out.write('1\n')

    # Write the mesh data (NX XMIN XMAX NY YMIN YMAX NZ ZMIN ZMAX)

    file_out.write('1 -100 100 1 0 -100 500 -60.48375 60.48375\n')

    # Check to see if STAR-CCM_ .csv file has been created and wait for a predetermined amount of time if it hasn't
    time_to_wait = 1000000
    time_counter = 0
    filename = r'./ExtractedBotData/He3Data_table_'+str(STAR_STEP)+'.csv'
    while not os.path.exists(filename):
        time.sleep(1)
        time_counter += 1
        if time_counter > time_to_wait:break
    if os.path.isfile(filename):
        # Take the data from STAR-CCM+ .csv file and write to ifc
        columns = ['Position in Cartesian 1[X] (cm)', 'Density(g/cm^3) (kg/m^3)', 'Temperature (K)']
        df = pd.read_csv(filename, usecols=columns)
```

```
    # Position = df['Position in Cartesian 1[X] (cm)']
    # Density =  df['Density(g/cm^3) (kg/m^3)']
    # Temp = df['Temperature (K)']

    # Sorts data from STAR-CCM+ in ascending order by position
    df_convert = df.to_numpy()
    numpy_array = df_convert[df_convert[:,0].argsort()]

    # Cross Section Data for He-3 in SERPENT2 only exists for 300 K or above, this will only happen early on in the transient and temp
    # should never fall that far below 300K so setting this arbitrary constraint should have minimal change in results
    for i in range(STAR_Points):
        if numpy_array[i,2] < 300.0:
            numpy_array[i,2] = 300.0
    denstemp = numpy_array[:,[1,2]]
    np.savetxt(file_out,denstemp, fmt = "%1.6f")
else:
    raise ValueError("%s has not been created or could not be read" % filename)
# Close interface file

file_out.close()
# Update STAR_STEP by number of steps that STAR takes before updating data
STAR_STEP += step_length

############################
# Calculate TH-solution for fuel   #
############################

# Fuel specific heat capacity

cp = 998  # J/(kg*K)

# Calculate EOI temperatures at (nz) axial nodes
# No heat transfer, just deposition
```

```
for i in range(10):
    # Calculate EOI temperature based on BOI temperature
    # and energy deposition during current time interval

    # Calculate mass of this node (in kg) (z-slice of reactor)
    #   Area of Active Fuel (cm^2)  * Length of Node (cm) * density (g/cm^3)  * conversion (g to kg)
    #   314 Fuel Assemblies (93.16 cm^2 each)
    #   20 Control Rod Assemblies ( Fuel Assembly Area - Control Rod Area) (65.65 cm^2 each)
    #   4 HENRI Assemblies (Fuel Assembly Area - HENRI Area) (73.53 cm^2)
    #   Density of Fuel assumed to be 1.72 g/cm^3 (mostly Graphite)
    #   z-slice (10 slices / total length of reactor (121 cm))

    m = ((93.16*314)+(65.65*20)+(75.53*4)) * 12.1 * 1.72 * 1e-3

    # Calculate initial heat in this axial node

    Q = TBOI[i] * (cp * m)

    # The interface output is Joules in case of time dependent
    # simulation, no need to multiply with time step

    dQ = data_fuelpass[i]

    # Calculate new temperature based on new amount of heat

    TEOI[i] = (Q + dQ) / (cp * m)

############################
# Update interface       #
############################

file_out = open('./fuel.ifc', 'w')
```

```
# Write the header line (TYPE MAT OUT)

file_out.write('2 fuel 0\n')

# Write the mesh type

file_out.write('1\n')

# Write the mesh size (NX XMIN XMAX NY YMIN YMAX NZ ZMIN ZMAX)

file_out.write('1 -500 500 1 -500 500 10 -60.48375 60.48375\n')

# Write updated fuel temperatures

for i in range(10):
   # Use the base density throughout the simulation
   # Write density and temperature at this layer

   file_out.write('-1.72 {}\n'.format(TEOI[i]))

file_out.close()

#############################################################
# Tell code to move to next timestep #
#############################################################
file_out = open('com.in','w')
file_out.write(str(signal.SIGUSR2))
file_out.close()

#############################################################
# Archive Files                          #####
#############################################################
```

```
copyfile('HE3TOP.ifc','Archive/HE3TOP.ifc'+str(curtime))
copyfile('HE3BOT.ifc','Archive/HE3BOT.ifc'+str(curtime))
copyfile('fuel.ifc', 'Archive/fuel.ifc' + str(curtime))
copyfile('STAR_HeatTop.csv','Archive/Star_HeatTop'+str(curtime)+'.csv')
copyfile('STAR_HeatBot.csv','Archive/Star_HeatBot'+str(curtime)+'.csv')
if curtime >= 2:
   copyfile('coupledTreat_res.m','Archive/coupledTreat_res'+str(curtime)+'.m')
# Delete Files that are not needed between iterations
os.remove(STARTop)
os.remove(STARBot)
os.remove(textfile)
# Increment time step
curtime += 1

# Copy EOI temperatures to BOI vector

for i in range(10):
   TBOI[i] = TEOI[i]
####################################
 # Check if simulation has finished #
####################################
if (simulating == 0):
      break

time.sleep(5)
file_out = open('com.in','w')
file_out.write(str(signal.SIGUSR2))
file_out.close()
```