A FUEL BURNUP MOLTEN SALT REACTOR MULTIPHYSICS BENCHMARK

A Thesis

by

ANDREW MICHAEL HERMOSILLO

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Jean C. Ragusa
Committee Members,	Mauricio Tano Retamales
	Pavel V. Tsvetkov
	Michael B. Pate
Head of Department,	Michael Nastasi

August 2021

Major Subject: Nuclear Engineering

ABSTRACT

Molten Salt Reactors (MSRs) are being researched as possible reactors to be built in the next generation of nuclear reactors. The interest in building these reactors has required improvement in current numerical methods for modeling the physics of these reactors. Current codes will need to update their features to include the physics of molten salt reactors or new codes will need to be developed in order to account for these physics. A numerical benchmark for modeling MSRs has been created for the modeling of these physics, however this benchmark could be updated to include the physics of isotope depletion (burnup).

The work presented in this thesis entails using the finite volume method in an open-source code, FiPy, to solve the partial differential equations required for the modeling of the generic MSR design in the original benchmark. Python scripts were then utilized to visualize the results produced by FiPy and compare them to the values obtained in the original MSR benchmark. Once the model in FiPy is verified against the original MSR data, the benchmark could then be extended to include the additional physics modeling of isotope depletion. The Monte Carlo code, Serpent, was utilized to generate burnup cross sections based on the original MSR benchmark parameters and these cross sections were inserted into the python code to extend the original benchmark to include burnup.

The results from FiPy matched closely with the original benchmark results for the steady-state problems with little to no discrepancy between them. The burnup cross sections were generated to simulate a six month operating period and were input into the FiPy simulations to generate the change in eigenvalue for FiPy over the original fully coupled problem for an MSR. Results were obtained for these burunp calculations and suggest FiPy was capable of handling newly-generated cross sections to model the physics of isotope depletion in an MSR.

ACKNOWLEDGMENTS

I would like to thank Dr. Jean Ragusa, my advisor, for supporting me both in undergraduate and graduate school, offering me advice and mentorship throughout my academic career. Thank you for helping me grow as both a student and professional in this field. Dr. Mauricio Tano, thank you for the kind words of encouragement and always setting aside time to help me whenever I ran into any problems.

Thank you to my committee members, Dr. Pavel Tsvetkov and Dr. Michael Pate, for their feedback and advice in helping me complete this work.

Thank you to my friends and office-mates, Nolan MacDonald, Begoña Aranguren, Tarek Ghaddar, Robert Turner, and Peter German for keeping my life fun and interesting throughout graduate school.

To Lindsey Randolph, thank you for reminding me to enjoy the finer things in life.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a thesis consisting of advisor Professor Jean Ragusa, Professor Pavel Tsvetkov, Professor Mauricio Tano of the Department of Nuclear Engineering and Professor Michael Pate of the Department of Mechanical Engineering.

The original python script for modeling molten salt reactors was provided by Dr. Mauricio Tano of Texas A&M before modifying it to perform the benchmark steps.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This material is based upon work supported under an NEUP-IRP Award of the U.S. Department of Energy, Office of Nuclear Energy (contract reference DE-NE0008651).

NOMENCLATURE

LWR	Light Water Reactor
MSR	Molten Salt Reactor
GIF	Generation-IV International Forum
CNRS	National Centre for Scientific Research
MSRE	Molten Salt Reactor Experiment
MSBR	Molten Salt Breeder Reactor
ORNL	Oak Ridge National Laboratory
MSFR	Molten Salt Fast Reactor
MCFR	Molten Chloride Fast Reactor
DNP	Delayed Neutron Precursor
PDE	Partial Differential Equation
P_N	Spherical Harmonics
SP_N	Simplified Spherical Harmonics
BCs	Boundary Conditions
FV	Finite Volume
SIMPLE	Semi-Implicit Method for Pressure Linked Equations
LHS	Left Hand Side
RHS	Right Hand Side
CRAM	Chebyshev Rational Approximation Method

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

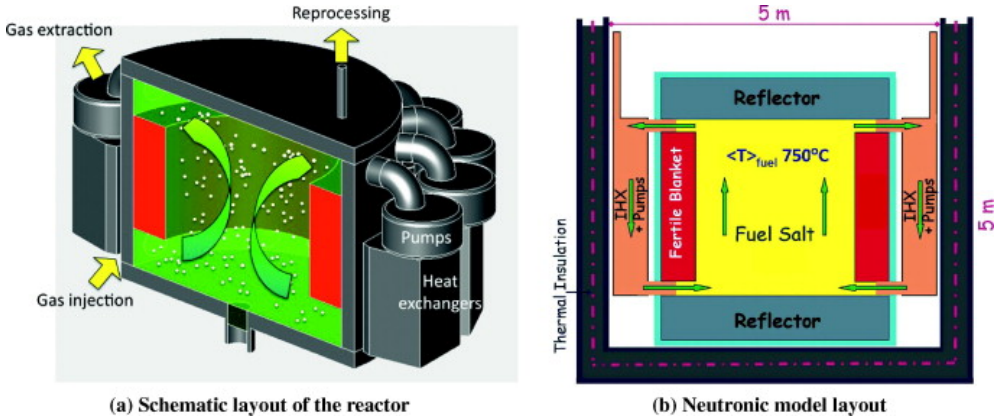
LIST OF TABLES

TABLE Page

1. INTRODUCTION

In order to ensure a cost-effective manner for simulating the operation of a nuclear reactor, computational methods are used in reactor development. Current computational codes are designed to model and simulate the physics of light water reactors (LWRs) and ensure the safe operation of these reactors. However, with the current push to develop newer and safer reactor technologies from the Generation-IV International Forum (GIF), reactor design has strayed from the conventional LWR design to include features that promote sustainability, safety and reliability, economic feasibility, and proliferation resistance. From GIF, six reactor designs emerged as a focus for research and development: The gas-cooled fast reactor, lead-cooled fast reactor, the molten salt reactor (MSR), the sodium-cooled fast reactor, the supercritical water-cooled reactor, and the very high-temperature reactor[1]. Of these the MSR is the only reactor in which the fuel moves in a fluid motion, requiring updates to current coding frameworks to allow for the movement of fuel in a simulation instead of using a static fuel structure. This moving fuel also creates issues in that delayed neutron precursors (DNPs) can be lost and move out of the core, requiring a method for modeling the loss of DNPs in the core as they move throughout. Lastly, the MSR also requires advances in modeling of the physio-chemical processes that occur within reactor due to the use of salt fuels that may interact with the surrounding materials.

The original Molten Salt Reactor Experiment (MSRE) was performed at Oak Ridge National Laboratory (ORNL) with original criticality in 1965 and ran until 1969 [2]. The objective of the original experiment was to demonstrate the capabilities of using a molten salt as a reactor fuel and the practicality of building such a system. The original design was a graphite moderated thermal reactor however modern reactors typically employ a fast neutron spectrum. Later, the Molten Salt Breeder Reactor (MSBR) was created with the intention of breeding thorium to create fissionable U-233. This reactor also contained a unique safety design of a freeze plug which would allow for draining the molten salt system should an accident occur and the temperature increases to unsafe levels. This design was eventually abandoned due to lack of eventual funding but its ideas lived on



(a) Schematic layout of the reactor (b) Neutronic model layout

Figure 1.1: A schematic design showing the layout of the MSFR along with a 2D axial slice of the reference MSFR design.

for future modern reactors. These modern reactors that followed the MSBR design included the Molten Salt Fast Reactor (MSFR) in Europe[3] and the Molten Chloride Fast Reactor (MCFR) in the United States.

The MSFR project arose from funding from the Safety Assessment of the Molten Salt Fast Reactor (SAMOFAR) project with the goal of coming up with a design of a reactor concept that can be built and run safely[4]. The reference MSFR design contains a cylindrical core with reflectors on the above and below it. The outer periphery of the core contains a fertile fuel blanket to improve the breeding in the reactor. For the thermal hydraulics, the fuel salt is fed from the bottom of the core and exits out the top through 16 different pumps and heat exchangers in the core [5]. A schematic design of the MSFR can be seen in Figure 1.1.

As part of SAMOFAR's objectives, the need for updating or developing new tools for modeling MSRs has become a priority and benchmarking is an important step of this. The current MSFR design utilizes the safety plug design of the MSBR, includes methods for online fuel recycling through the thorium fuel cycle, and uses a fast spectrum for its neutron energy. The use of these features require updates in core modeling and simulation for showing the transmutation of nuclides into fissionable material in a molten salt system, physio-chemical modeling of the movement of the salt through the freeze plug to allow adequate draining in a timely manner, and neutronics

simulations to capture the expected neutron spectrum and its effects on the thermal hydraulics in a MSR.

These reactors improve current reactor safety by using molten salt fuels that have higher thermal conductivities than light water, allowing for heat to be moved out of the core in a more efficient manner and flattening the thermal distribution. The molten salt also increases proliferation resistance since the liquid form is much harder to filter out plutonium while also burning off plutonium produced during normal operation. The reactor meets sustainability goals by burning off transuranics during operation and having a closed fuel cycle that allows for actinide recycling by chemically reprocessing the fuel. Online reprocessing also improves the economics of the reactor by reducing shutdown times for the reactor and removing the need for fuel fabrication through chemical reprocessing. These improvements in reactor design, however, create the need to more accurately model the physics of flowing fuel, online reprocessing, and actinide burnup.

Current conventional codes model the fuel as a static structured lattice of fuel pins but the introduction of a molten salt as the fuel creates the need to model the fuel as a moving environment within the reactor. This creates a more difficult task within the modeling of the reactor where the fuel fluid motion must be coupled to the nuclear physics aspects of the reactor. This multiphysics coupling creates a problem where the solution is not clearly defined with experiments and creates difficulty in verifying if a code created for modeling a MSR is functioning correctly. The National Centre for Scientific Research (CNRS) benchmark [6], based off of the MSFR design, was created to address these issues by allowing a code-to-code comparison of MSR multiphysics simulations and verify that the solutions obtained by each research code converged to a reasonable solution with each other. Previous benchmarks for MSRs only focused on single physics and used experimental data from the original MSRE to benchmark against[7] [8]. The CNRS benchmark contains 3 phases:

1. Steady-state single physics modeling

2. Steady-state multiphysics modeling

3. Transient multiphysics modeling

Moving further up each phase adds an additional amount of complexity to the simulation but allows for it to become closer to modeling reality. The first phase deals with modeling the single physics of fluid movement, neutronics, and the energy equations as singular equations in a square geometry. The second phase begins coupling fluid movement and neutronics through precursor drift, temperature feedback effects on neutronics, and buoyancy effects on neutronics through the Boussinesq approximation before finally coupling all of these physics together in a single steady-state simulation. The last phase then focuses on performing a transient simulation and measuring the phase shift and gain over the course of the transient. Matching the results of the benchmark would help verify that the code is reproducing the physics correctly and serves as a stepping stone for testing more complicated versions of MSRs. This benchmark, however, could be improved to include the physics of fuel depletion, reprocessing, and more complicated geometries on the nuclear physics and thermal hydraulics aspects of reactor operation. Current codes rely on the finite volume (FV) method and the finite element method (FEM) in order to solve the physics of nuclear reactors. Testing and verification of these codes is needed to ensure that the physics coupling capabilities of these codes are functional and that the discretization schemes are efficient in solving for the quantities of interest in a nuclear reactor. Typical open source finite volume and finite element codes have routines available to solve single physics simulations and these can be extended to include multiphysics coupling as well as coupling to other codes to allow different codes to solve for different physics. Some open-source FV discretizations can be found in codes such as FiPy [9] and be used for solving multiphysics benchmarks as well as extruding them to work with other codes to include additional physics aspects. One method of extruding would be to couple a FV code to a burnup code such as Serpent to add the physics of fuel depletion into the multiphysics equations. Including these features would bring current codes one step closer to modeling the full physics of a MSR.

1.1 Objective

A new benchmark containing fuel burnup and its effect on MSR operation is proposed with the goal of serving as a benchmark for other codes to compare their burnup simulations to. The addition of this feature would allow for the creation of a more realistic simulation for the operation of a MSR and serve as another building stage towards the eventual construction of MSRs. Investigation into the optimal burnup step and its effects on the k-eigenvalue solution will be studied in order to ensure a viable solution point for other codes to benchmark towards. Performing these steps would demonstrate the capabilities of FiPy in solving partial differential equations (PDEs) using the finite volume method and its ability to couple with burnup codes such as Serpent for performing reactor physics operations.

The results presented in this work are intended to verify the abilities of the code in modeling the multiphysics aspects and are not necessarily intended to model completely realistic reactor situations.

2. GOVERNING LAWS

The governing laws defining operation of MSRs are the neutron transport equation, the Navier-Stokes equations, and the energy equation.

2.1 Neutronics Equations

2.1.1 The Neutron Transport Equation

The steady-state neutron transport equation is used for describing the movement of neutrons in a medium and is given by:

$$\vec{\Omega}\cdot\vec{\nabla}\psi(\vec{r},E,\vec{\Omega})+\Sigma_t(\vec{r},E)\psi(\vec{r},E,\vec{\Omega}) = \int_0^\infty dE' \int_{4\pi} d\Omega'\Sigma_s(\vec{r},E'\rightarrow E,\vec{\Omega}'\rightarrow\vec{\Omega})\psi(\vec{r},E,\vec{\Omega})+Q(\vec{r},E,\vec{\Omega}) \tag{2.1}$$

The terms of the neutron transport equation are as follows:

- $\vec{\Omega}\cdot\vec{\nabla}\psi(\vec{r},E,\vec{\Omega})$ describes the streaming or advection term and gives the net leakage of neutrons per unit volume, per unit energy, per unit angle, per unit time out of the control volume V.

- $\Sigma_t(\vec{r},E)\psi(\vec{r},E\ \vec{\Omega})$ is the total interaction term and describes the number of neutron collisions per unit volume, per unit energy, per unit angle, per unit time within the control volume. The interaction term contains neutron collisions involving both scattering and absorption.

- $\int_0^\infty dE' \int_{4\pi} d\Omega'\Sigma_s(\vec{r},E'\rightarrow E,\vec{\Omega}'\rightarrow\vec{\Omega})\psi(\vec{r},E\ \vec{\Omega})$ is the total inscattering source term and defines the number of neutrons per unit volume, energy, angle, and time that scatter from an angle and energy outside of the current phase space into the current phase space volume. The resolution of this integral term is described later in the multi-group section.

- $Q(\vec{r},E,\vec{\Omega})$ is the external source term and describes the number of neutrons produced per unit volume, energy, angle, and time through fission, neutron emission, and other extraneous sources.

The neutron transport equation is a balance equation of neutrons in space, energy, angle, and time and when solved yields the angular flux $\psi(\vec{r}, E, \vec{\Omega})$. The angular flux is used for describing the number of neutrons at position $\vec{r}$ with energy E and direction $\vec{\Omega}$ per unit area, per unit energy, per unit time, per unit steradian in $\frac{neutrons}{cm^2-MeV-ster-s}$. This quantity is important because it is a intermediary for generating the scalar flux, which is simply the angular flux integrated out over the solid angle of the unit sphere:

$$\phi(\vec{r}, E) = \int_{4\pi} \psi(\vec{r}, E, \vec{\Omega}) d\Omega \tag{2.2}$$

The scalar flux is a function that describes the number of neutrons at position $\vec{r}$ with energy E per unit energy per unit area per unit time in $\frac{neutrons}{cm^2-MeV-s}$. This fundamental quantity is useful for determining the reactor power, reaction rates, and fuel burnup. Difficulties, however, lie within solving the steady-state neutron transport equation due to the nature of it requiring six independent variables in phase space: three in real space, one in energy, and two in angle. The solution methods for solving this equation fall into two categories: stochastic and deterministic.

In stochastic methods, a Monte Carlo method is used for tracking the life of a neutron from its birth to death while neutron cross sections describe the neutron's probability of survival during an interaction with the surrounding media. In Monte Carlo, a neutron is "born" through physical processes such as fission or a neutron source and can "die" through absorption into the medium or by leaking outside of the problem boundary. The Monte Carlo method tracks the neutron's distance to collision and will sample if the neutron "died" or "survived" the collision where it will sample its survival probability through a random number generator based on the cross section of the material the neutron collided with. If the neutron survives the collision, a new scattering angle will be determined and a new distance to collision will be computed and the process will repeat until the neutron's death. This method gives very accurate results, however, can be computationally time consuming due to the nature of having to track each particle and requiring a large number of particles being sampled in order to obtain good statistics for error reporting.

For deterministic transport, a discretization scheme is utilized to create a system of linear equa-

tions in the form of $Ax = b$ with N discrete values. The matrix A is a square $N \times N$ matrix containing the coefficients of the unknowns in the transport while x contains an N-sized solution vector of the unknowns and b contains the source terms in a column vector. For small systems, this linear equation can be inverted to solve to yield $x = A^{-1}b$ to give the angular flux solution to the transport equation. However, for most realistic transport systems, inverting the matrix is not computationally feasible and iteration techniques must be utilized to obtain the solution.

2.1.2 The P_N and SP_N Methods

Due to the difficulties in solving the full transport equation, some approximations can be made to ease the computational burden [10]. Namely, the simplified spherical harmonics (SP_N) equations can be used in reducing the complexity of the solution method for the full transport equation [11] [12]. The SP_N equations start from the 1D slab-geometry transport equation with isotropic source:

$$\mu \frac{\partial \psi(x,\mu)}{\partial x} + \sigma_t(x)\psi(x,\mu) = \int_{-1}^{1} \sigma_s(x,\mu_0)\psi(x,\mu')d\mu' + \frac{Q(x)}{2} \tag{2.3}$$

From the 1D transport equation, a trial space of Legendre polynomials are assumed for the angular flux:

$$\psi(x,\mu) = \sum_{n=0}^{N} \frac{2n+1}{2}\phi_n(x)P_n(\mu). \tag{2.4}$$

The differential scattering cross-section is given as:

$$\sigma_s(x,\mu_0) = \sum_{n=0}^{\infty} \frac{2n+1}{2}\sigma_{s,n}(x)P_n(\mu_0) \tag{2.5}$$

$$\sigma_{s,n} = \int_{-1}^{1} d\mu_0 \sigma_s(x,\mu_0)P_n(\mu_0) \tag{2.6}$$

For the source term, the source moments are defined as:

$$Q(x,\mu) = \sum_{n=0}^{\infty} \frac{2n+1}{2} q_n(x) P_n(\mu) \tag{2.7}$$

$$q_n(x) = \int_{-1}^{1} d\mu Q(x,\mu) P_n(\mu) \tag{2.8}$$

Inserting in these definitions for the source moments and the expansion for the angular flux will give:

$$\mu \sum_{n=0}^{\infty} \frac{2n+1}{2} \frac{\partial \phi_n(x)}{\partial x} P_n(\mu) + \sigma_t \sum_{n=0}^{\infty} \frac{2n+1}{2} \phi_n(x) P_n(\mu) = \sum_{n=0}^{\infty} \frac{2n+1}{2} \sigma_{s,n} \phi_n(x) P_n(\mu) + \sum_{n=0}^{\infty} \frac{2n+1}{2} q_n P_n(\mu) \tag{2.9}$$

Taking the moments of the Legendre polynomials in μ and utilizing the Legendre recursive relationship:

$$(2n+1)\mu P_n(\mu) = (n+1)P_{n+1}(\mu) + nP_{n-1}(\mu), \tag{2.10}$$

while multiplying through by $P_k(\mu)$ and integrating over all directions will yield the spherical harmonics (P_N) equations for a 1D slab geometry. The P_N equations with an isotopic source in 1D are thus given by:

$$\frac{n}{2n+1} \frac{\partial \phi_{n-1}}{\partial x} + \frac{n+1}{2n+1} \frac{\partial \phi_{n+1}}{\partial x} + (\sigma_t - \sigma_{s,n})\phi_n = q_n \delta_{n,0} \quad 0 \leq n \leq N. \tag{2.11}$$

To simplify the equation further, we can define:

$$\sigma_n = \sigma_t - \sigma_{s,n} \tag{2.12}$$

where for the $0'th$ moment, $\sigma_0 = \sigma_t - \sigma_{s,0} = \sigma_a$. The P_N equations will give $N+1$ equations

and unknowns assuming that the $N+1$ moment of the flux is 0 for closure. The P_N equations, however, can be difficult to solve in multi-dimensions due to the expansion of the angular variable in spherical harmonics and require more simplifications to allow for multi-dimensional solving and reduce the number of equations. Solving for the flux for $n > 0$ will yield:

$$\phi_n(x) = -\frac{1}{\sigma_n}\Big(\frac{n}{2n+1}\frac{\partial\phi_{n-1}}{\partial x} + \frac{n+1}{2n+1}\frac{\partial\phi_{n+1}}{\partial x}\Big) \tag{2.13}$$

Assuming that this definition is used for odd values of n, this value of the flux can be substituted into equations with even values of n. This will yield equations of the form:

$$\begin{aligned} &-\frac{\partial}{\partial x}\Big(\frac{1}{\sigma_{n-1}}(\frac{n^2-n}{4n^2-1}\frac{\partial\phi_{n-2}}{\partial x} + \frac{n^2}{4n^2-1}\frac{\partial\phi_n}{\partial x})\Big) \\ -\frac{\partial}{\partial x}\Big(\frac{1}{\sigma_{n+1}}(\frac{(n+1)^2}{4n^2+8n+3}\frac{\partial\phi_n}{\partial x} + \frac{n^2+3n+2}{4n^2+8n+3}\frac{\partial\phi_{n+2}}{\partial x})\Big) &+ \sigma_t\phi_n = 0 \end{aligned} \tag{2.14}$$

for $0 \leq n \leq N-1$ with even values of n. For closure the $\phi_{n+2} = 0$ at $n = N-1$ to match the assumption of $\phi_{N+1} = 0$. For the case of $n = 0$, $\phi_2 = 0$ for closure and the 1-D P_N equation is given as:

$$\frac{\partial\phi_1}{\partial x} + \sigma_t\phi_0 = q_0 \tag{2.15}$$

$$\phi_1 = -\frac{1}{\sigma_1}\frac{1}{3}\frac{\partial\phi_0}{\partial x}. \tag{2.16}$$

Inserting ϕ_1 back into Eqn. 2.15 will yield:

$$-\frac{\partial}{\partial x}(\frac{1}{\sigma_1}\frac{1}{3}\frac{\partial\phi_0}{\partial x}) + \sigma_t\phi_0 = q_0 \tag{2.17}$$

This is similar to inserting $n = 0$ into Eqn. 2.14, however the $n = 0$ case will contain a source term, q_0 in its equation while the others will return q_n = 0 due to the isotropic source condition.

Using the methodology described in Eqn. 2.14 will reduce the number of equations and unknowns to $\frac{2N+1}{2}$ and creates 1-D diffusion-like equations in the form of $-\frac{\partial}{\partial x}(\frac{1}{\sigma_n}\frac{\partial \phi}{\partial x})$. The SP_N equations then extend the P_N to 3-D by changing any $\frac{\partial}{\partial x}$ terms to ∇ terms to create a 3-D diffusion operator $\nabla \cdot \frac{1}{\sigma_n}\nabla\phi$. Truncating the Legendre expansion to $N = 3$ will create the SP_3 equations. The SP_3 equations are given as:

$$-\nabla \cdot \frac{1}{3\sigma_1}\nabla\phi_0 - \nabla \cdot \frac{2}{3\sigma_0}\nabla\phi_2 + \sigma_a\phi_0 = q_0 \tag{2.18}$$

$$-\nabla \cdot \frac{2}{15\sigma_1}\nabla\phi_0 - \nabla \cdot (\frac{4}{15\sigma_1} + \frac{9}{35\sigma_3})\nabla\phi_2 + \sigma_2\phi_2 = 0 \tag{2.19}$$

To simplify the SP_3 equations even further, composite moments for the fluxes can be defined:

$$\varphi_1 = \phi_0 + 2\phi_2 \tag{2.20}$$

$$\varphi_2 = \phi_2 \tag{2.21}$$

The SP_3 equations can thus be defined as:

$$-\nabla \cdot \frac{1}{3\sigma_1}\nabla\varphi_1 + \sigma_a\varphi_1 = q_0 + 2\varphi_2 \tag{2.22}$$

$$-\nabla \cdot \frac{2}{15\sigma_1}\nabla\varphi_1 - \nabla \cdot \frac{9}{35\sigma_3}\nabla\varphi_2 + \sigma_2\varphi_2 = 0 \tag{2.23}$$

To remove the φ_1 in Eqn 2.23, the $-\nabla \cdot \frac{1}{3\sigma_1}\nabla\varphi_1$ term can be solved for in Eqn 2.22 and inserted back into Eqn 2.23 to have it in terms of φ_2. In this form, coefficients can be defined to create a coupled system of diffusion-like equations:

$$D_0 = \frac{1}{3\sigma_1} \tag{2.24}$$

$$D_1 = \frac{3}{7\sigma_3} \tag{2.25}$$

$$\alpha = \frac{5}{3}\sigma_2 + \frac{4}{3}\sigma_a. \tag{2.26}$$

Inserting these definitions into the SP_3 equations will yield a compact form:

$$-D_0\varphi_1'' + \sigma_a\varphi_1 = q_0 + 2\sigma_a\varphi_2 \tag{2.27}$$

$$-D_1\varphi_2'' + \alpha\varphi_2 = -\frac{2}{3}q_0 + \frac{2}{3}\sigma_a\varphi_1 \tag{2.28}$$

In the case of $N = 1$, the SP_1 equations will simply return the standard diffusion equation:

$$-\nabla \cdot \frac{1}{3\sigma_1}\nabla\phi_0 + \sigma_a\phi_0 = q_0 \tag{2.29}$$

2.1.3 Boundary Conditions

For the boundary conditions (BCs) of the SP_N equations, they are similar to the 1D Marshak BCs used for the P_N equations and extended to 3D by changing $\frac{\partial}{\partial x}$ terms to $\hat{n} \cdot \nabla$ terms. The 1-D Marshak boundary conditions are given as:

$$\int_{-1}^{0} P_k(\mu)\psi(X,\mu)d\mu = \int_{-1}^{0} P_k(\mu)S(X,\mu) + \int_{-1}^{0} P_k(\mu)\psi(X,-\mu)d\mu \tag{2.30}$$

The cavity used in the MSR benchmark utilizes vacuum boundary conditions. For vacuum boundary conditions without an external source or incident flux the equation simplifies to:

$$\int_{-1}^{0} P_k(\mu)\psi(X,\mu)d\mu = 0, \tag{2.31}$$

for odd values of k. Expanding the moments of the angular flux using $\psi(x,\mu) = \sum_{n=0}^{N} \frac{2n+1}{2}\phi_n(x)P_n(\mu)$ and inserting it into the BC will give $\frac{N+1}{2}$ BCs:

$$\int_{-1}^{0} P_k(\mu)\psi(X,\mu)d\mu = \sum_{n=0}^{N} \frac{2n+1}{2}\phi_n(x)\int_{-1}^{0} P_k(\mu)P_n(\mu)d\mu = 0 \tag{2.32}$$

In the case of N = 1, the SP_1 vacuum BCs are given as:

$$\sum_{n=0}^{N} \frac{2n+1}{2}\phi_n(x)\int_{-1}^{0} P_k(\mu)P_n(\mu)d\mu = \frac{1}{2}\phi_0\int_{-1}^{0} P_0(\mu)P_1(\mu)d\mu + \frac{3}{2}\phi_1\int_{-1}^{0} P_1(\mu)P_1(\mu)d\mu = 0 \tag{2.33}$$

Evaluating the values for the integrals of the Legendre polynomials on the half range:

$$\frac{1}{2}\phi_0(-\frac{1}{2}) + \frac{3}{2}\phi_1(\frac{1}{3}) = -\frac{1}{4}\phi_0 + \frac{1}{2}\phi_1, \tag{2.34}$$

and substituting in $\phi_1 = -\frac{1}{3\sigma_1}\hat{n}\cdot\nabla\phi_0$ and simplifying will yield:

$$\frac{1}{2}\phi_0 + \frac{1}{3\sigma_1}\hat{n}\cdot\nabla\phi_0 = 0. \tag{2.35}$$

For $N = 3$, the SP_3 vacuum BCs are:

$$\frac{1}{2}\phi_0\int_{-1}^{0} P_0(\mu)P_1(\mu)d\mu + \frac{3}{2}\phi_1\int_{-1}^{0} P_1(\mu)P_1(\mu)d\mu + \frac{5}{2}\phi_2\int_{-1}^{0} P_2(\mu)P_1(\mu)d\mu \tag{2.36}$$

$$\frac{1}{2}\phi_0\int_{-1}^{0} P_0(\mu)P_3(\mu)d\mu + \frac{5}{2}\phi_2\int_{-1}^{0} P_2(\mu)P_1(\mu)d\mu + \frac{7}{2}\phi_3\int_{-1}^{0} P_3(\mu)P_3(\mu)d\mu. \tag{2.37}$$

After evaluating the integrals and simplifying, the two vacuum BCs for the SP_3 equations are:

$$\frac{1}{2}\phi_0 + \frac{1}{3\sigma_1}\hat{n}\cdot\nabla\phi_0 + \frac{5}{8}\phi_2 + \frac{2}{3\sigma_1}\hat{n}\cdot\nabla\phi_2 = 0 \tag{2.38}$$

$$-\frac{1}{8}\phi_0 + \frac{5}{8}\phi_2 + \frac{3}{7\sigma_3}\hat{n}\cdot\nabla\phi_2 = 0 \tag{2.39}$$

Replacing the flux values with the composite fluxes changes the boundary conditions for SP_3

to:

$$\frac{1}{2}\varphi_1 + \frac{1}{3\sigma_1}\hat{n}\cdot\nabla\varphi_1 - \frac{3}{8}\varphi_2 = 0 \tag{2.40}$$

$$-\frac{1}{8}\varphi_1 + \frac{7}{8}\varphi_2\frac{3}{8\sigma_3}\hat{n}\cdot\nabla\varphi_2 = 0 \tag{2.41}$$

The SP_1 BCs would change to:

$$\frac{1}{2}\varphi_1 + \frac{1}{3\sigma_1}\hat{n}\cdot\nabla\varphi_1 = 0 \tag{2.42}$$

for the composite flux form with φ_2 equal to 0.

2.1.4 SP_N in Multi-group

The multi-group method is used to discretize the energy dependence of the neutron transport equation. In multi-group, cross sections are divided into G groups based on their energy ranges where cross sections over a certain energy range will retain a single value. In the multi-group form the number of energy groups is problem-specific, where simpler problems require fewer groups while more complicated problems require more refinements in energy and thus more groups. In the multi-group method, the group angular flux and group scalar flux are defined as:

$$\psi^g(\vec{r},\Omega) \equiv \int_{\Delta E_g} \psi(\vec{r},E,\Omega)dE \tag{2.43}$$

$$\phi^g(\vec{r}) \equiv \int_{\Delta E_g} \phi(\vec{r},E)dE \tag{2.44}$$

Inserting the definition of the group scalar flux into the compact SP_3 equations (Eqn. 2.27) will yield:

$$-D_0\varphi_1'' + \sigma_a\varphi_1 = q_0 + 2\sigma_a\varphi_2 \tag{2.45}$$

$$-D_1\varphi_2'' + \alpha\varphi_2 = -\frac{2}{3}q_0 + \frac{2}{3}\sigma_a\varphi_1 \tag{2.46}$$

In this form, the composite fluxes and sources are stored as multi-group vectors:

$$\varphi_n = \begin{bmatrix} \varphi_n^1 \\ \varphi_n^2 \\ \vdots \\ \varphi_n^G \end{bmatrix}, q_0 = \begin{bmatrix} q_0^1 \\ q_0^2 \\ \vdots \\ q_0^G \end{bmatrix} \tag{2.47}$$

where the subscript n describes the Legendre moment and the superscript g denotes the energy group. The "diffusion coefficients" are stored in a diagonal matrix while cross sections are stored in a full matrix:

$$D_n = \begin{bmatrix} D_n^1 & 0 & \dots & 0 \\ 0 & D_n^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & D_n^G \end{bmatrix}, \sigma_n = \begin{bmatrix} \sigma_n^1 & -\sigma_{s,n}^{2\to1} & \dots & -\sigma_{s,n}^{G\to1} \\ -\sigma_{s,n}^{1\to2} & \sigma_n^2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -\sigma_{s,n}^{G\to G-1} \\ -\sigma_{s,n}^{1\to G} & \dots & -\sigma_{s,n}^{G-1\to G} & \sigma_n^G \end{bmatrix} \tag{2.48}$$

with n and g retaining their definitions as the Legendre moments and energy group respectively. For the cross section, the diagonal retains the definition of $\sigma_n^g = \sigma_t^g - \sigma_{s,n}^g$ while the off-diagonals contain the up and down-scattering components. In cases without up-scattering the above-diagonal terms are defined to be 0.

2.1.5 SP_N with Fission

In reactor problems, the generic source q is typically replaced by a fission source to simulate the production of neutrons from fission and an in-scatter source to simulate the phenomena of neutrons scattering from one energy group into another energy group for multi-group problems with fission. Expanding the generic source q to account for fission and in-scattering terms will give:

$$q_0 = (1-\beta)\chi_p^g \sum_{g'=1}^{G} \nu\sigma_f^{g'}\phi_0^{g'} + \chi_d^g \sum_{i=1}^{I} \lambda_i C_i + \sum_{g'=1}^{G} \sigma_{s,0}^{g'\to g}\phi_0^{g'} \tag{2.49}$$

where β describes the effective fraction of neutrons that have a delayed release, χ_p and χ_d describe the prompt and delayed fission spectra, $\nu\sigma_f$ is the fission cross section, λ_i is the decay constant of the i'th precursor group, C_i is the concentration of the $i'th$ precursor, and $\sigma_{s,0}^{g'\to g}$ is the scattering cross section from group g' to group g. The equation contains three main parts: neutrons produced from prompt fission, neutrons produced from delayed fission, and neutrons that scatter from other groups, g', into the current energy group, g. The prompt fission kernel describes neutrons that are immediate produced after fission and can contribute immediately to the chain reaction process. The delayed fission kernel, requires that DNPs decay into isotopes that release neutrons and thus have a wait time before the neutrons can be added. The delayed fission kernel requires an additional auxiliary equation for determining the concentration of the $i'th$ precursor. The steady-state version of this equation can be given as:

$$C_i = \frac{\beta}{\lambda_i} \sum_{g'=1}^{G} \nu\sigma_f^{g'} \phi_0^{g'} \tag{2.50}$$

It can be noted that expanding this equation out and for cases in which the delayed contribution is small, it can be merged with the prompt fission kernel to yield the total fission kernel: $\chi \sum_{g'=1}^{G} \nu\sigma_f^{g'} \phi_0^{g'}$, however for MSRs this is typically not the case.

The isotropic source assumption still holds for SP_N with fission and will return $\nu\sigma_{f,n}^{g'} = \sigma_{s,n}^{g'\to g} = 0$ due to scattering from other groups being isotropic for $n > 0$ and $g' \neq g$, thus defining fission only for the 0'th source moment. Inserting this definition of the source into the multi-group

SP_3 equations (Eqn. 2.45) will give:

$$-\nabla \cdot D_0 \nabla \varphi_1 + \sigma_a \varphi_1 = 2\sigma_a \varphi_2 + (1-\beta)\chi_p^g \sum_{g'=1}^{G} \nu\sigma_f^{g'}(\varphi_1^{g'} - 2\varphi_2^{g'}) + \chi_d^g \sum_{i=1}^{I} \lambda_i C_i + \sum_{\substack{g'\neq g \\ g'=1}}^{G} \sigma_{s,0}^{g'\to g}(\varphi_1^{g'} - 2\varphi_2^{g'}) \tag{2.51}$$

$$-\nabla \cdot D_1 \nabla \varphi_2 + \alpha\varphi_2 = \frac{2}{3}\sigma_a \varphi_1 + \frac{2}{3}\Big((1-\beta)\chi_p^g \sum_{g'=1}^{G} \nu\sigma_f^{g'}(\varphi_1^{g'} - 2\varphi_2^{g'}) + \chi_d^g \sum_{i=1}^{I} \lambda_i C_i + \sum_{\substack{g'\neq g \\ g'=1}}^{G} \sigma_{s,0}^{g'\to g}(\varphi_1^{g'} - 2\varphi_2^{g'})\Big) \tag{2.52}$$

When solving these equations, they will return φ_1 and φ_2 and the scalar flux can be recovered using the relationship:

$$\phi_0 = \varphi_1 - 2\varphi_2 \tag{2.53}$$

For the SP_1 equations, the multi-group diffusion equation is returned:

$$-\nabla \cdot D_0 \nabla \phi_0 + \sigma_a \phi_0 = (1-\beta)\chi_p^g \sum_{g'=1}^{G} \nu_f \sigma_f^{g'} \phi_0^{g'} + \chi_d^g \sum_{i=1}^{I} \lambda_i C_i + \sum_{\substack{g'\neq g \\ g'=1}}^{G} \sigma_{s,0}^{g'\to g} \phi_0^{g'} \tag{2.54}$$

With the scalar flux recovered form these equations, it can be used for determining certain reactor parameters. In particular, the reactor power and reaction rates can be determined in the core. To determine the reactor power from the scalar flux, the following equation defines the reactor power over the volume of the core:

$$P = \int_V \sum_{g=1}^{G} \kappa_f \Sigma_f^g \phi_0^g dV, \tag{2.55}$$

where κ_f is the energy per fission, $\Sigma_{f,g}$ is the fission cross section, and ϕ_0^g is the multi-group scalar

flux. For determining reaction rates, the expression:

$$RR = \int_V \sum_{g=1}^{G} \Sigma_x^g \phi_0^g dV, \tag{2.56}$$

is used where Σ_x^g denotes a cross section of type x and in group g, ϕ_0^g is the group scalar flux, and V is the volume of the core.

2.1.6 Extension to MSR Applications

In a MSR, neutronics coupling with the thermal hydraulics is necessary for determining the effects of the of the heat produced in the reactor on the velocity field as well as the temperature feedback effects on the cross sections. The use of molten fuel requires modeling of the velocity field and its effects on the neutronics since some materials may fission in a different spot compared to where they were born, creating changes in the power distribution of the reactor.

Of note in the multi-group equations with fission is the presence of a C_i term in the delayed fission kernel. Fuel movement causes the displacement of DNPs to where their production may occur in another physical location or outside of the core where they are lost. In the case of DNP drift, the steady-state DNP equation changes to include a convection term:

$$\lambda_i C_i + \vec{u} \cdot \vec{\nabla} C_i = \beta \sum_{g=1}^{G} \nu \Sigma_{f,g} \phi_g. \tag{2.57}$$

In this form, a velocity field, $\vec{u}$, is required for modeling the movement of the DNPs in the MSR core and requires extensive modeling of the thermal hydraulics in the core to obtain a representative field for the movement of the precursors.

Temperature feedback is simplified in the benchmark and only studies the effect of temperature on the cross sections through density feedback rather than through Doppler broadening. For this benchmark, this assumption is acceptable due to the MSR using a fast system and the Doppler

effect being negligible in this case. The temperature effect on cross sections is modeled as:

$$\Sigma_x(T) = \frac{\rho(T)}{\rho_0}\Sigma_x(T_0) \tag{2.58}$$

$$D(T) = \frac{\rho_0}{\rho(T)}D(T_0), \tag{2.59}$$

where ρ_0 is the initial fluid density, $\rho(T)$ is the fluid density at the prescribed temperature T, and $\Sigma_x(T_0)$ and $D(T_0)$ are the cross section of type x and diffusion coefficient values at the initial reference temperature. To determine the value for the temperature-dependent density, the thermal expansion equation is used:

$$\rho(T) = \rho_0(1 - \beta(T - T_0)) \tag{2.60}$$

where β is the value of the working fluid's thermal expansion coefficient, T is the current temperature, and T_0 is the initial temperature. With the need for temperature and velocity in the neutronics equations, the focus is shifted to thermal hydraulics equations to determine the values for these parameters.

2.2 Thermal Hydraulics

Thermal hydraulics focuses on modeling the temperature distribution and movement of a fluid in a geometry. Thermal hydraulics consists of two main components: velocity modeling through the Navier-Stokes equation and temperature modeling through the conservation of energy equation.

2.2.1 The Navier-Stokes Equations

The Navier-Stokes equation is used for describing the flows of the molten salt in an MSR. The Navier-Stokes equations focus on solving two parameters, fluid velocity and pressure. These parameters are modeled through the use of a conservation of mass equation and a conservation of momentum equation. In the case of a constant density, the flow field is considered incompressible and allows for removing density coupling with velocity. The Incompressible Navier-Stokes equations describe the motion of viscous fluids and is given by:

$$\nabla \cdot \vec{u} = 0 \tag{2.61}$$

$$\rho(\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u}) = \mu \nabla^2 \vec{u} - \nabla p \tag{2.62}$$

with boundary conditions:

$$\vec{u} = \vec{v} \in \Gamma_D, \tag{2.63}$$

where ρ is the fluid density, $\vec{u}$ is the fluid velocity, μ is the fluid kinematic viscosity, and p is the fluid pressure. For Dirichlet boundaries, Γ_D, the velocity is imposed as a user-specified value, $\vec{v}$. The equations are split into conservation of mass equation 2.61 called the continuity equation and a conservation of momentum equation 2.62. A common convention in solving Navier-Stokes problems is to divide the momentum equation through by the fluid density, ρ, and instead solve:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = \nu \nabla^2 \vec{u} - \nabla \frac{p}{\rho} \tag{2.64}$$

$$\tag{2.65}$$

In this form of the equation, the kinematic viscosity, μ is converted into the dynamic viscosity, ν while the pressure being solved for becomes the kinematic pressure, $\frac{p}{\rho}$. The left hand side of the equation contains the convective terms, where $\frac{\partial \vec{u}}{\partial t}$ describes the time rate of change of the velocity field while $\vec{u} \cdot \nabla \vec{u}$ describes the fluid's motion. The right hand side contains a pressure gradient, ∇p and a viscous term $\nu \nabla^2 \vec{u}$ that describes the diffusion of the fluid. The discretization technique used for solving this PDE is described in Chapter 3.

For modeling the addition of buoyancy effects on the velocity field, the Boussinesq approximation is employed. This approximation describes the buoyancy forces exhibited by the fluid as its temperature changes using the thermal expansion equation. The Boussinesq physics kernel is

given by:

$$\rho(1 - \beta_T(T - T_{ref}))\vec{g}, \tag{2.66}$$

where ρ is the initial fluid density, β_T is the thermal expansion coefficient, T is the current temperature, T_{ref} is the initial fluid temperature, and $\vec{g}$ is the acceleration due to gravity. Adding the Boussinesq approximation to the momentum equation of Eqn 2.64 will give:

$$\frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} = \nu \nabla^2 \vec{u} - \nabla p + (1 - \beta(T - T_{ref}))\vec{g} \tag{2.67}$$

$$\tag{2.68}$$

Adding this effect will allow for most of the fluid physics to be modeled for MSRs and requires only one more piece of physics to be modeled in temperature effects. The solution of this equation will return a velocity field that is used in solving the DNP equation for neutronics and heat convection in the conservation of energy equation.

2.2.2 The Conservation of Energy Equation

The conservation of energy equation can be used to describe the change in temperature within the system and is given as:

$$\rho C_p(\frac{\partial T}{\partial t} + \nabla \cdot (\vec{u}T)) = \nabla \cdot (k \nabla T) + Q_{fission} - \gamma(T - T_{ref}) \tag{2.69}$$

where ρ is the fluid density, C_p is the specific heat capacity of the fluid, $\vec{u}$ is the velocity solution from the Navier-Stokes equation, T is the fluid temperature, T_{ref} is the initial fluid temperature, k is the thermal conductivity, $Q_{fission}$ is the heat source from fission, and γ is the volumetric heat transfer coefficient. A common convention in the conservation of energy equation is to divide both sides through by ρC_p to leave the time derivative and convection term without constants in front.

This will yield:

$$\frac{\partial T}{\partial t} + \nabla \cdot (\vec{u}T) = \nabla \cdot (\alpha \nabla T) + \frac{Q_{fission}}{\rho C_p} - \frac{\gamma}{\rho C_p}(T - T_{ref}) \tag{2.70}$$

where α is defined as $\alpha \equiv \frac{k}{\rho C_p}$. The conservation of energy equation is quite similar to the Navier-Stokes equation setup where the left hand side contains the convection terms and the right hand side contains the diffusive terms and body forces. The $\frac{\partial T}{\partial t}$ term describes the time rate of change of the fluid temperature while the $\nabla \cdot (\vec{u}T)$ describes the temperature advection, or the movement of the heat within along the fluid's velocity field. For incompressible flows, this term can be changed to $\vec{u} \cdot \nabla T$ similar to the DNP equation where the velocity is pulled out of the dependent variable. On the right hand side, the $\nabla \cdot (k \nabla T)$ term describes the thermal diffusivity where the value of the thermal conductivity determines the rate at which the heat diffuses through the fluid. For a constant thermal conductivity, the diffusive term $\nabla \cdot (k \nabla T)$ term can be described using a Laplacian, $k \nabla^2 T$. The last term in the equation, $\gamma(T - T_{ref})$ describes the heat sink where heat is absorbed by the working fluid. The constants used in both the Navier-Stokes and energy equations are described in Table 2.1.

Table 2.1: Molten salt thermodynamic and transport properties.

Property	Units	Value
Density	kg m^{-3}	2.0 $\times 10^{3}$
Kinematic Viscosity	m^2 s^{-1}	2.5 $\times 10^{-2}$
Volumetric heat capacity	J m^{-3} K^{-1}	6.15 $\times 10^{6}$
Thermal expansion coefficient	K^{-1}	2.0 $\times 10^{-4}$
Prandtl number	-	3.075 $\times 10^{5}$
Schmidt number	-	2.0 $\times 10^{8}$
Specific heat capacity	J kg^{-1} K^{-1}	3.075 $\times 10^{-3}$
Thermal Conductivity	W m^{-1} K^{-1}	5.0 $\times 10^{-1}$
α	m^2 s^{-1}	8.13 $\times 10^{-8}$

2.2.3 Thermal Hydraulics Boundary Conditions

For the thermal hydraulics equations, Dirichlet boundary conditions are applied to the Navier-Stokes equations while Neumann boundary conditions are given to the conservation of energy equation. The no-slip condition is applied to the bottom, left, and right walls of the cavity used for the MSR benchmark, meaning that the velocity on these walls is set to 0. On the top wall, a lid-velocity is used $u_x = 0.5$ m/s, $u_y = 0$ m/s to create a lid-cavity problem for the Navier-Stokes equations. The BCs can be given as:

$$\vec{u} = \vec{0} \in \Gamma_D, x = 0, x = L, y = 0 \tag{2.71}$$

$$\vec{u} = \vec{u_{lid}} \in \Gamma_D, y = L \tag{2.72}$$

On the top corners where the nodes conflict, the velocity is set to 0 and the next node on the top will contain the lid velocity. For the temperature equation, adiabatic boundary conditions are applied, setting the heat flux or temperature derivative on the wall boundaries to 0. This is given as:

$$-k\nabla T \cdot \hat{n} = 0 \in \Gamma_N, \tag{2.73}$$

where k is the thermal conductivity, ∇ is the gradient operator, and $\hat{n}$ is the unit normal to the surface. This boundary condition is applied to all walls and ensures no there is no heat flux on the boundary for the lid-cavity problem.

3. DISCRETIZATION AND SOLUTION TECHNIQUES

This chapter describes the finite volume discretization used for all of the physics described in Chapter 2 then follows it up with a sections describing the different solution techniques used for some of the individual physics and lastly, a section covering the overall multiphysics coupling.

3.1 Finite Volume Method

The main method used for solving the governing laws defining neutronics and thermal hydraulics is a computational method called the finite volume (FV) method. The FV method starts by defining a mesh and breaking up the spatial domain into small cells. In each cell, a conservation equation is attached to it and discretized in space using the boundaries of the cell. Interface conditions are applied at each cell face and the conservation equation is integrated over the volume of the cell. In essence, the discretized integral form of the conservation equation is solved in each cell and gives a fully conservative estimate to the conservation equation. For estimating derivatives in the integral forms of conservation equations, a finite difference method is employed within each cell.

The FV method solves for the average of the quantity of interest in each cell and tracks the fluxes into and out of the faces of the cell using the cell-centered values of the quantities of interest from the adjacent cells. Solving for these flux values ensures that the FV method is conservative since any particles leaking out of the cell are picked up by the adjacent cell so that no particles are lost during a solve. The FV method described will follow the methods used in the FiPy open-source code for solving the equations of interest.

The discretization of the SP_1 equations in the finite volume method would begin by integrating over volume:

$$\int_V -\nabla \cdot D_0 \nabla \phi_0 + \sigma_a \phi_0 dV = \int_V q_0 dV \tag{3.1}$$

Next the equation is divided up into individual control volumes on the mesh and the divergence

theorem can be used to break up volume integrals into surface integrals:

$$-\int_S D_0 \vec{n} \cdot \nabla \phi_0 dS + \int_V \sigma_a \phi_0 = \int_V q_0 dV \tag{3.2}$$

Using finite difference schemes to handle derivatives and performing the integration over each control volume will yield:

$$\Big(\sum_f^{N_{faces}} -D_{0,f} \frac{\phi_{0,C}^{i+1} - \phi_{0,C}^{i}}{d_{cell}} A_f \Big) + \sigma_a \phi_{0,C} V_{cell} = q_{0,C} V_{cell} \tag{3.3}$$

where f subscripts denote values on the face, C subscripts denotes values at the cell center, D_0 is the diffusion coefficient, ϕ_0^{i+1} denotes the flux of the adjacent cell center, ϕ_0^i denotes the flux on the current cell center, d_{cell} is the distance between the cell centers, A_f is the area of the face, $q_{0,C}$ is the value of the source at the cell center, and V_{cell} is the volume of the control volume. Similarly, for SP_3 equations the finite volume discretization is given as:

$$\Big(\sum_f^{N_{faces}} -D_{0,f} \frac{\varphi_{1,C}^{i+1} - \varphi_{1,C}^{i}}{d_{cell}} A_f \Big) + \sigma_{a,C} \varphi_{1,C} V_{cell} = 2\sigma_{a,C} \varphi_{2,C} V_{cell} + q_{0,C} V_{cell} \tag{3.4}$$

$$\Big(\sum_f^{N_{faces}} -D_{1,f} \frac{\varphi_{2,C}^{i+1} - \varphi_{2,C}^{i}}{d_{cell}} A_f \Big) + \alpha_C \varphi_{2,C} V_{cell} = \frac{2}{3} \sigma_{a,C} \varphi_{1,C} V_{cell} + \frac{2}{3} q_{0,C} V_{cell} \tag{3.5}$$

with the same cell and face syntax defined previously for the SP_1 equations. For a 2D problem, the number of faces in each interior cell is 4, creating a loop over the top, bottom, left, and right faces for creating the divergence terms. The q_0 term contains the fission terms and will need the fluxes on the cell centers in order to compute the total source term. Another term needed would be the precursor concentrations on the cell centers and those can be solved with the finite volume method using:

$$\int_V \lambda_i C_i + \vec{u} \cdot \vec{\nabla} C_i dV = \int_V \beta \sum_{g=1}^{G} \nu \Sigma_{f,g} (\varphi_1 - 2\varphi_2) dV \tag{3.6}$$

and applying the divergence theorem to the convection term:

$$\int_V \lambda_i C_i dV + \int_S (\vec{n} \cdot \vec{u}) C_i dS = \int_V \beta \sum_{g=1}^{G} \nu \Sigma_{f,g} (\varphi_1 - 2\varphi_2) dV. \tag{3.7}$$

Performing the integrals over the control volumes will give the final equation used in the FV method for the DNPs:

$$\lambda_{i,C} C_{i,C} V_{cell} + \sum_{f}^{N_{faces}} (\vec{n} \cdot \vec{u})_f C_{i,f} A_f = \beta_C \sum_{g=1}^{G} \nu \Sigma_{f,g,C} (\varphi_{1,C} - 2\varphi_{2,C}) V_{cell}, \tag{3.8}$$

where subscripted C terms indicate values on the cell centers and f terms represent the values on cell faces, $\lambda_{i,C}$ is the decay constant of the $i'th$ precursors, C_i is the concentration of the precursor, $\vec{u}$ is the velocity at the face, A_f is the area of the face, β is the delayed neutron fraction, ν is the number of neutrons per fission, Σ_f is the fission cross section, φ represents the composite flux moment, and V_{cell} is the volume of the cell. These equations conclude the FV notation for the neutronics terms however thermal hydraulics are also solved using the FV method.

For the Navier-Stokes equation, the FV method can applied as:

$$\int_V \frac{\partial \vec{u}}{\partial t} + \vec{u} \cdot \nabla \vec{u} dV = \int_V \nu \nabla^2 \vec{u} - \nabla \frac{p}{\rho} + (1 - \beta(T - 900)) \vec{g} dV. \tag{3.9}$$

Using the divergence theorem as seen in the neutronics equations and applying time indices to the momentum equation:

$$\frac{u_C^{\ell+1} - u_C^{\ell}}{\Delta t} V_{cell} + \sum_{f}^{N_{faces}} (\vec{n} \cdot \vec{u_C^{\ell}}) \vec{u_C^{\ell+1}} A_f =$$

$$\sum_{f}^{N_{faces}} \nu_f \vec{n} \cdot \frac{u_C^{i+1,\ell+1} - u_C^{i,\ell+1}}{d_{cell}} A_f - \frac{p_C^{i+1}/\rho_C^{i+1} - p_C^{i}/\rho_C^{i}}{d_{cell}} V_{cell} + (1 - \beta_{T,C}(T_C - T_{ref,C})) \vec{g_C} V_{cell} \tag{3.10}$$

where $\ell + 1$ indicates the current time step, ℓ represents the old time step value, Δt is the time step

size, $\vec{u}$ is the velocity, ν is the kinematic viscosity, p is the pressure, ρ is the density, T is the fluid temperature, β_T is the thermal expansion coefficient, T_{ref} is the initial fluid temperature, and $\vec{g}$ is the acceleration due to gravity in each cell. The energy equation is also defined similarly to the Navier-Stokes equation in the FV method:

$$\int_V \frac{\partial T}{\partial t} + \nabla \cdot (\vec{u}T)dV = \int_V \nabla \cdot (\alpha \nabla T) + \frac{Q_{fission}}{\rho C_p} - \frac{\gamma}{\rho C_p}(T - T_{ref}). \tag{3.11}$$

Applying the divergence theorem and integrating over each control volume:

$$\frac{T_C^{\ell+1} - T_C^{\ell}}{\Delta t} V_{cell} + \sum_f^{N_{faces}} (\vec{n} \cdot \vec{u_C}) T_C^{\ell+1} A_f = \sum_f^{N_{faces}} \alpha_f \vec{n} \cdot \frac{T_C^{i+1,\ell+1} - T_C^{i,\ell+1}}{d_{cell}} A_f + \frac{Q_{fission,C}}{\rho_C C_{p,C}} V_{cell} - \frac{\gamma_C}{\rho_C C_{p,C}}(T_C^{\ell+1} - T_{ref,C}) \tag{3.12}$$

In the FV method, the updated values at $\ell + 1$ are solved for and moved to the LHS while the old values at $t = \ell$ are moved to the RHS to update at each time step.

3.2 Power Iteration

For solving the eigenvalue problem, a technique called power iteration is used [13]. In power iteration, an equation in the form of an ordinary eigenvalue problem

$$A\vec{x} = \lambda \vec{x} \tag{3.13}$$

can be solved by lagging the left hand side (LHS) equation. With the LHS lagged, the flux on the right hand side (RHS) of the equation can be updated using the operation of A on the previous flux value and eigenvalue. Adding indices to demonstrate this technique shows the power iteration scheme as:

$$x^{i+1} = \frac{1}{\lambda^i} A x^i \tag{3.14}$$

where x^{i+1} is the current iterate of the flux, λ_i is the previous eigenvalue, and x^i is the previous flux value. Once the flux value is updated, the eigenvalue can be updated using the Rayleigh quotient:

$$\lambda^{i+1} = \lambda^i \frac{x^{(i)T} Q x^{i+1}}{x^{(i)T} Q x^i}, \tag{3.15}$$

where Q is the source term. Relating this to the SP_3 equations, we can set up a matrix form by first defining a loss operator:

$$Ax = \begin{bmatrix} -\nabla \cdot D_0 \nabla + \sigma_a - \sum_{\substack{g'=1 \\ g' \neq g}}^{G} & -2\sigma_a + 2\sum_{\substack{g'=1 \\ g' \neq g}}^{G} & -\chi_d \sum_{i=1}^{I} \lambda_i \\ -\frac{2}{3}\sigma_a - \frac{2}{3}\sum_{\substack{g'=1 \\ g' \neq g}}^{G} & -\nabla \cdot D_1 \nabla + \alpha + \frac{4}{3}\sum_{\substack{g'=1 \\ g' \neq g}}^{G} & -\frac{2}{3}\chi_d \sum_{i=1}^{I} \lambda_i \\ 0 & 0 & \lambda_i + \vec{u} \cdot \nabla \end{bmatrix} \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ C_i \end{pmatrix} \tag{3.16}$$

and a production operator for the right hand side:

$$Bx = \begin{bmatrix} (1-\beta)\chi_p \sum_{g'=1}^{G} \nu\sigma_f^{g'} & -2(1-\beta)\chi_p \sum_{g'=1}^{G} \nu\sigma_f^{g'} & 0 \\ \frac{2}{3}(1-\beta)\chi_p \sum_{g'=1}^{G} \nu\sigma_f^{g'} & -\frac{4}{3}(1-\beta)\chi_p \sum_{g'=1}^{G} \nu\sigma_f^{g'} & 0 \\ \beta\chi_d \sum_{g'=1}^{G} \nu\sigma_f^{g'} & -2\beta\chi_d \sum_{g'=1}^{G} \nu\sigma_f^{g'} & 0 \end{bmatrix} \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ C_i \end{pmatrix} \tag{3.17}$$

For the eigenvalue form of the SP_3 equations, a k-eigenvalue is added in front of the production terms. This creates equations of the form:

$$Ax = \frac{1}{k} Bx \tag{3.18}$$

The matrix A can be inverted onto the eigenvalue equation and the source fluxes and precursors can be lagged to update the LHS

$$x^{i+1} = \frac{1}{k^i} A^{-1} B x^i. \tag{3.19}$$

The eigenvalue update is thus given as:

$$k^{i+1} = k^i \frac{((1-\beta)\chi_p^g \sum_{g'=1}^{G} \nu\sigma_f^{g'}(\varphi_1^{g',i+1} - 2\varphi_2^{g',i+1}) + \chi_d^g \sum_{j=1}^{J} \lambda_j C_j^i)V}{((1-\beta)\chi_p^g \sum_{g'=1}^{G} \nu\sigma_f^{g'}(\varphi_1^{g',i} - 2\varphi_2^{g',i}) + \chi_d^g \sum_{j=1}^{J} \lambda_j C_j^i)V}. \tag{3.20}$$

This eigenvalue will be used for updating the delayed neutron source in the DNP equation,

$$\lambda_i C_i + \vec{u} \cdot \vec{\nabla} C_i = \frac{1}{k}\beta \sum_{g=1}^{G} \nu\Sigma_{f,g}\phi_g, \tag{3.21}$$

as well as the fission source terms in the SP_N equations. The power iteration method works to find the largest eigenvalue, which is the eigenvalue associated with the fundamental mode flux of the nuclear reactor and describes the ratio of the production of neutrons to their losses. This eigenvalue is known as k-effective and in general is a useful quantity for determining neutron production rates in the reactor and ensuring it is operating in a critical state where the neutron losses are equal to the neutron production (k-effective = 1). With the power iteration method, both quantities of interest (eigenvalue and scalar fluxes) are obtained for the simulation.

3.3 SIMPLE Algorithm

The Navier-Stokes portion of the MSR benchmark is solved using the Semi-Implicit Method for Pressure Linked Equations (SIMPLE) algorithm [14]. It is an iterative algorithm that uses a predictor-corrector method in order to solve for velocity and update the pressure at each step until convergence. The SIMPLE algorithm starts by first linearizing the Navier-Stokes equation by lagging one of the velocities on the nonlinear velocity term and using an initial guess for the pressure:

$$\frac{\partial \vec{u^*}}{\partial t} + \vec{u^n}\nabla\vec{u^*} - \nu\nabla^2\vec{u^*} = -\frac{\nabla p^*}{\rho} \tag{3.22}$$

where u^n is the lagged velocity, $\vec{u^*}$ is the predictor velocity, and p^* is the lagged pressure field value. The next step of the SIMPLE algorithm is to discretize the equation for the derivatives and

set it up in matrix form as:

$$MU = -\frac{\nabla p^*}{\rho} \tag{3.23}$$

where the matrix M contains the discretized velocity time derivative, diffusion term, and convection term, U contains the velocity unknowns, and the RHS contains the lagged pressure field. After the discretization phase, the matrix M is split into a diagonal and off-diagonal term:

$$MU = AU - H \tag{3.24}$$

where A contains the diagonal terms and H contains the off-diagonal terms. This is done to allow for easy inversion of the matrix A and reduce the computational footprint on solving the Naiver-Stokes equations. With the matrix separated into different components, the predictor step for velocity can be solved:

$$U = A^{-1}H - A^{-1}\frac{\nabla p}{\rho}. \tag{3.25}$$

With the velocity expression solved for, it can be reinserted into the continuity equation to derive an equation for pressure:

$$\nabla \cdot \vec{u} = 0 \tag{3.26}$$

$$\nabla \cdot (A^{-1}H - A^{-1}\frac{\nabla p}{\rho}) = 0 \tag{3.27}$$

Rearranging the continuity equation with the predictor velocity will give a Poisson equation for pressure:

$$\nabla \cdot (A^{-1}\frac{\nabla p}{\rho}) = \nabla \cdot (A^{-1}H) \tag{3.28}$$

This Poisson equation is the corrector step of the SIMPLE algorithm where the pressure in this equation can be reinserted into the predictor velocity equation to correct for the initial guess of the pressure. This process is repeated until convergence for velocity and pressure to complete the SIMPLE algorithm.

3.3.1 Rhie-Chow Interpolation

The implementation of the SIMPLE algorithm on a collocated grid can create issues with odd-even decoupling that leads to a checkerboarding effect for the pressure where a low and high value of pressure can exist a cell apart from each other and create instability for the velocity solution as well due to their decoupling. Odd-even decoupling occurs due to the pressure gradient depending on non-consecutive and alternating cells, which only enforce a consistent gradient value across the non-consecutive faces and can numerically oscillate between consecutive faces. The Rhie-Chow interpolation [15] can help prevent this checkerboarding effect by adding a dissipation term to the linear interpolation used for calculating cell face velocities. The dissipation term helps with recoupling the pressure and velocity by taking the estimates of cell pressure gradients across consecutive cell faces to ensure a consistent pressure across cells. The Rhie-Chow interpolation is given as:

$$\vec{u_f} = \bar{\vec{u_f}} - \bar{D_f^u}(\nabla p_f - \bar{\nabla p_f}) \tag{3.29}$$

where the subscript f denotes a face value, D is the flow diameter across the face, and bar terms denote average values across the face of interpolation.

For its relation to the SIMPLE algorithms, the Rhie-Chow interpolation is performed after the velocity predictor step to recouple the pressure to the velocity after solving of the momentum equation and allow for a coupled solve of the pressure Poisson equation for pressure instead of allowing for decoupling to occur. The steps of the SIMPLE algorithm with the Rhie-Chow approximation can thus be given as:

1. Initialize pressure and velocity values with initial guess

2. Solve the predictor equation for velocity using the previous guesses for velocity and pressure

3. Update the velocities according to the Rhie-Chow approximation

4. Solve the pressure Poisson equation to get the corrector value for the pressure

5. Repeat the process after Step 2 until convergence of pressure and velocities

3.4 Multiphysics Coupling

In solving for the temperature and velocity, they are coupled to the neutronics equations in that velocity is used for determining the DNP drift while temperature is used for cross section feedback. The neutronics equation, however, couples to the conservation of energy equation in that the heat released from fission is coupled to the temperature contained in the energy equation. This requires performing Picard iterations where the problem must be resolved with the most up to date values of temperature, velocity, and power until all variables converge to a single value within the specified tolerance before incrementing in time or ending the simulation. Although the benchmark is steady-state, a pseudo-transient may be used to help with providing numerical stability when solving the problem.

The coupled equations form a nonlinear system of equations in the form:

$$F(x) = 0 \tag{3.30}$$

where $F(x)$ can be written as:

$$F(x) = A(x)x - b = 0 \tag{3.31}$$

In order to solve the nonlinear system of equations, the $A(x)$ matrix can be lagged with the previous guess for the quantity of interest x and solved to give an updated value for x. The algorithm for

this would look like:

$$A(x^{\ell})x^{\ell+1} = b \tag{3.32}$$

where ℓ indicates the iteration index. In essence this creates a linear solve that allows for the updated values of $x^{\ell+1}$ to be found by inverting the lagged $A(x^{\ell})$ matrix. In following through with this fixed point iteration, the update for x is simply:

$$x^{\ell+1} = A^{-1}(x^{\ell})b \tag{3.33}$$

This process is continued until ℓ reaches the maximum number of iterations or the residual for $F(x) < tol$, which will approximate $F(x) \approx 0$. For a pseudo-transient, the term ℓ is replaced with t and a march forward in time is used until the simulation reaches steady state. The process for this iteration procedure is shown in Figure 3.1.

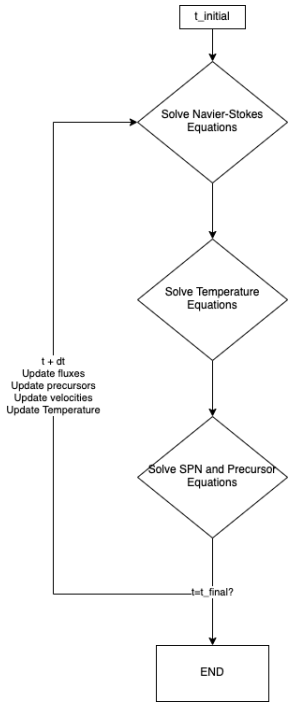


Figure 3.1: Display of the iteration process used in FiPy for solving coupled MSR benchmark problems.

4. ISOTOPE DEPLETION

4.1 Burnup

During reactor operation, fissionable material will absorb neutrons and break apart to form other elements as part of the fission process. The fissionable material will deplete over time, reducing the amount available and requiring the reactor to eventually need refueling to replenish the spent fuel. In order to measure the amount of material that was burnt, the burnup equation is used:

$$BU = \frac{PT}{m_{HM}} \tag{4.1}$$

where BU is the total burnup, P is the reactor power, T is the total time, and m_{HM} is the mass of heavy metal in the reactor. The typical units of burnup are in $\frac{MWd}{kgU}$ for uranium-fueled reactors.

Non-fissionable material will also absorb neutrons during reactor operation or change energy levels during a scattering event but instead will go through radioactive decay in order to change elements or isotopes as part of a decay chain. The change of both fissionable and non-fissionable material into different nuclides affects the cross sections for the overall reactor, which in turn will affect the overall scalar fluxes and k-eigenvalue for the reactor. The changes in material composition as a function of time for any nuclide, i, can be described using the Bateman equations for radioactive decay:

$$\frac{dN_i}{dt} = \sum_j N_j \sigma_{a,j\to i}\phi - \sum_j N_i \sigma_{a,i\to j}\phi + \sum_j \gamma_j N_j \lambda_j - N_i \lambda_i \tag{4.2}$$

where N_i is the atom density of nuclide i, N_j is the atomic density of a precursor or daughter product of i, $\sigma_{a,j\to i}$ is the absorption cross section for transmuting isotope j into i, ϕ is the scalar flux, σ_a is the absorption cross section of isotope i, γ_j is the branching ratio of the $j'th$ precursor, and λ is the decay constant of the isotope.

As these isotopes transmute, the atomic densities of them will change and thus affect the macroscopic cross sections due to their dependence on the atom density:

$$\Sigma = N\sigma \tag{4.3}$$

where Σ is the macroscopic cross section, N is the atomic density, and σ is the microscopic cross section. In order to extend the MSR benchmark to include isotope depletion as another physics component, cross section generation as a function of burnup will be needed. Assumptions made for producing cross sections were:

- No online fuel reprocessing
- Burnup is constant everywhere
- Constant power of 1 GW

For a simple benchmark extension, using the finite volume solver with cross sections at each burnup step should be sufficient for testing the multi-physics capabilities of the solver without complicating the problem further by introducing online fuel reprocessing and non-uniform burnup. While the flux distribution can be non-uniform, the movement of the fuel and mixing allows for the fuel to burn evenly and let a uniform burnup assumption to hold. This allows for constant cross sections at each burnup step with only density feedback from thermal expansion affecting the value of the cross sections. Online fuel reprocessing removes some of the fission products from the fuel before reinserting the fuel into the core and increase the amount of usable fuel without having as many actinides to absorb neutrons. While fuel reprocessing does occur in an MSR, the burnup cross sections were produced without reprocessing to create a simple case to test the abilities of the code to model burnup cross sections.

4.2 Serpent

For this work, the burnup routines and group constant generation will be used from Serpent 2 to generate the cross sections used for simulating the changes in the MSR as a function of burnup

Table 4.1: Salt isotopic composition used for the Serpent cross section generation simulations.

Isotope	Atomic Fraction (%)
Li-6	2.11488
Li-7	25.0836
Be-9	14.0992
F-19	56.39687
U-235	1.30545

Table 4.2: Energy groups used for the multi-group method and cross section generation.

Group, g	Upper Energy Bound (MeV)
1	2.000×10^{1}
2	2.231×10^{0}
3	4.979×10^{-1}
4	2.479×10^{-2}
5	5.531×10^{-3}
6	7.485×10^{-4}

[16]. Material properties for molten salt were entered following the parameters given within the original MSR benchmark and are described in Table 4.1. The geometry was set as a cuboid that was 2 meters long, 2 meters wide, and 1 meter tall. Vacuum boundary conditions were set for the sides while the top and bottom faces contained reflective boundary conditions. The neutron population was set to 1 million neutrons with 10,000 active cycles and 100 inactive cycles. Cross sections were set using the JEFF-3.1 libraries. The energy group layout followed the groups described in Table 4.2. The burnup mode was set to the using the Chebyshev Rational Approximation Method (CRAM) with a Quadratic Interpolation and Linear Extrapolation for the time integration method at each predictor-corrector step [17].

The original cross sections were provided by the CNRS benchmark but were regenerated very closely at the 0 burnup step as seen in Table A.1 contained in Appendix A. The differences at zero-burnup between the original cross sections and those generated as part of the isotope depletion could be attributed to differences in seeding and the nature of random particle generation. Cross sections were then generated at multiple burnup steps over an approximately 6-month period to

Table 4.3: Burnup steps used for generating the cross sections as a function of isotope depletion.

Burnup [MWd/kgU]	Time Elapsed [days]
0.0	0.0
1.0	1.424
10.0	14.24
20.0	28.48
50.0	71.20
100.0	142.4

simulate a single cycle in the MSFR configuration defined in the benchmark. For delayed neutron data, it matched the original DNP data described in the benchmark. When inserted into the fully-coupled problem, the cross sections also adopt a temperature dependence through density feedback as described previously in Chapter 3.

5. RESULTS

In this chapter, the results obtained from FiPy will be compared against the benchmark participants to determine the viability of using FiPy for performing MSR modeling and simulation. FiPy will perform steps 0.1, 0.2, 0.3, 1.1, 1.2, 1.3, and part of step 1.4 to verify it's capabilities for modeling MSFRs in a steady-state manner. Ones these steps have been completed, a burnup phase will be included where cross sections generated from Serpent will be inserted into the FiPy input file. The general guide for the workflow of the benchmark is show in Figure 5.1. The FiPy input file will then generate the k-effective value obtained at each burnup step to add its contribution to the MSR benchmark. Results in FiPy were generated using 200×200 uniform meshes for all steps.

The FiPy results are compared to the different benchmark participants from the original CNRS Benchmark paper. The participants are referred to as CNRS-SP_1 and CNRS-SP_3 for the spherical harmonics solver from the Center for Scientific Research [18], PSI for the GeN-Foam solver used by the Paul Sherrer Institute (PSI) [19], PoliMi for the Politecnico di Milano solver [20] in OpenFOAM [21], and TUD for the Delft University of Technology (TUD) Phantom-S_N solver[22].

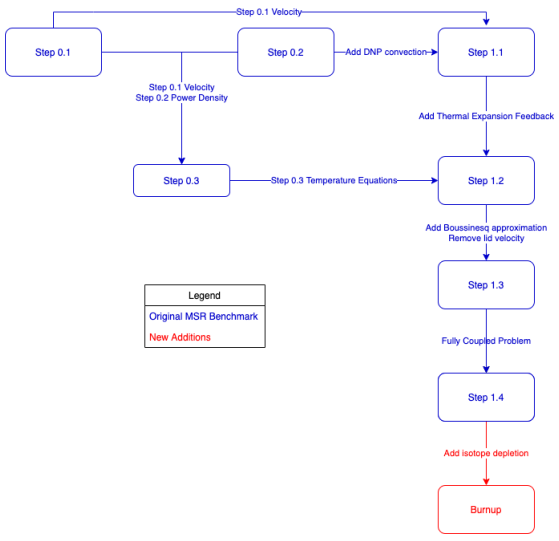


Figure 5.1: General flowchart of the different benchmark steps and the additions made to the original MSR benchmark.

5.1 Step 0.1

Step 0.1 of the MSR benchmark focuses on solving the Navier-Stokes equations as a simple lid-velocity problem without neutronics or temperature effects. The quantities of interest for this portion of the benchmark were the x-velocity and y-velocity along the horizontal and vertical mid-planes, AA and BB respectively. The Navier-Stokes equations were solved using the SIMPLE algorithm as described previously in section 3 with Rhie-Chow interpolation to help prevent instabilities. Velocity and pressure relaxation were also used, with values of 0.5 and 0.8 respectively. The x-velocity solutions can be found in Figure 5.2 while the y-velocity solutions are contained in Figure 5.3. The overall 2D velocity solutions are shown in Figure 5.4.

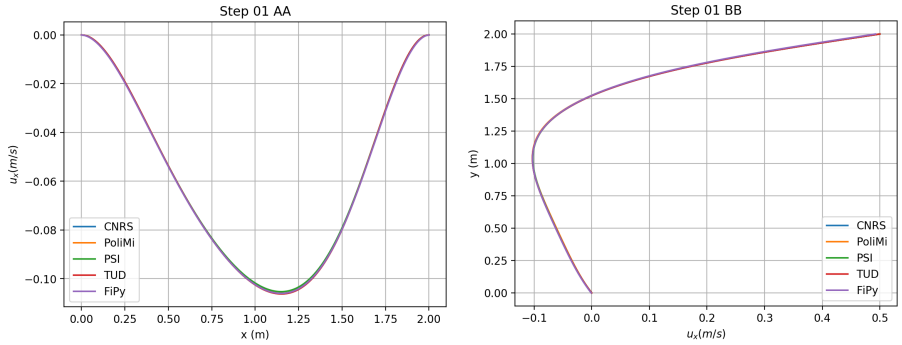


Figure 5.2: Velocities in the x-direction along the horizontal and vertical midplanes.

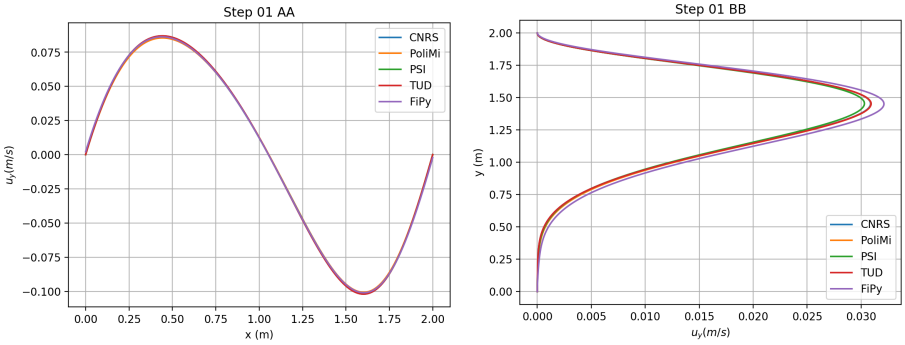


Figure 5.3: Velocities in the y-direction along the horizontal and vertical midplanes.

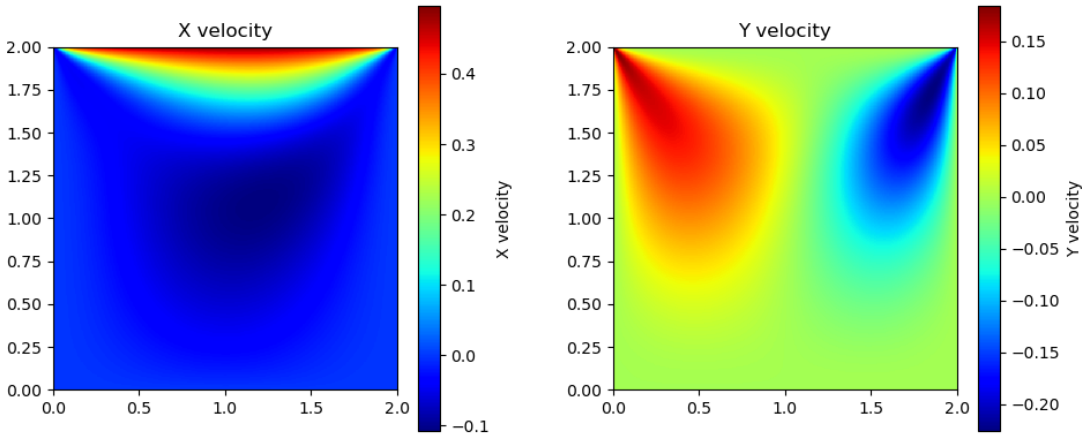


Figure 5.4: 2D velocity plots across the entire domain generated in FiPy for a 200×200 uniform mesh.

Good agreement can be found between the benchmark participant velocities and FiPy for a 200×200 mesh, with some discrepancy at the maximum y-velocity along the vertical midplane, BB. The average participant maximum value was 0.307 m/s while the FiPy maximum value was 0.321 m/s, which gives an error of 4.5% at this peak. The small magnitude of these values suggest FiPy is in good agreement with the other codes and can be used for solving Navier-Stokes equations. The velocity field obtained from this step will be reused in Steps 0.3, 1.1, and 1.2.

5.2 Step 0.2

Step 0.2 of the MSR benchmark solves a simple neutronics problem with vacuum boundary conditions and power at 1 GW. The quantities of interest for this problem are the fission rate densities along the horizontal and vertical midplanes, AA and BB, respectively, along with the reactivity. The results along with their associated errors obtained for the diffusion simulation are displayed in Figures 5.5 and 5.6.

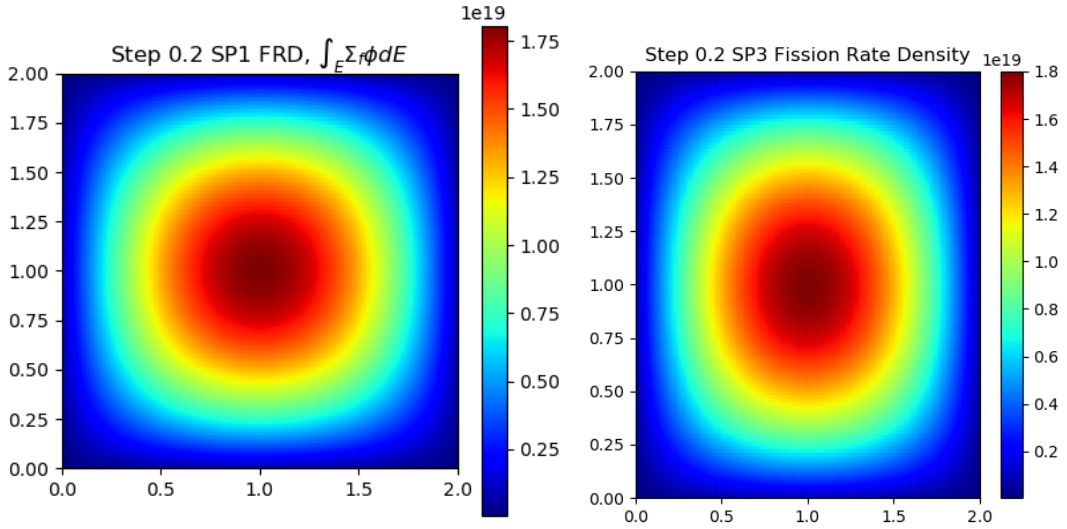


Figure 5.5: Overall fission rate density for FiPy using the diffusion and SP_3 methods.

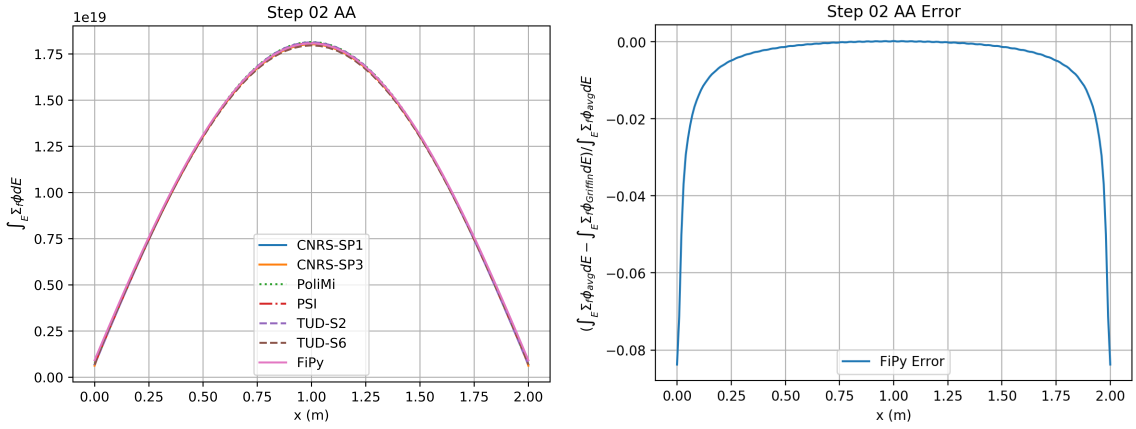


Figure 5.6: Fission rate densities in the x-direction along the horizontal midplane for FiPy using the diffusion method along with its associated error.

The fission rates obtained by FiPy match closely with the other benchmark values for both SP_3 and diffusion and are acceptable for a 200×200 mesh. The largest error was 8% and was at the boundaries while the average error was 0.5%. The error at the boundaries is due to the different approximation method for Robin boundary conditions used in FiPy to create a vacuum boundary

compared to the other benchmark participants' methods. The reactivities from FiPy and the other benchmark participants are shown in Table 5.1.

Table 5.1: All Step 0.2 reactivities.

Code	$\rho_{0.2}$ (pcm)
FiPy SP_1	421.2
FiPy SP_3	360.4
CNRS-SP_1	411.3
CNRS-SP_3	353.7
PoliMi	421.2
PSI	411.7
TUD-S_2	482.6
TUD-S_6	578.1

The FiPy results for diffusion are only a few pcm off of the other benchmark participants and follow closely with the diffusion solutions. For SP_3 the results were 10 pcm off of the CNRS SP_3 results, giving an error of 1.89%. The average reactivity was 443 pcm for all simulations while diffusion-only simulations had an average reactivity of 414.7 pcm. This would give FiPy a relative error of 4.94% and 1.57% to each average respectively, which is well within the standard deviation of the benchmark results. The diffusion code was deemed sufficient enough to continue using for the neutronics portions of the benchmark as well as having more participants to compare with and thus the SP_3 version was abandoned for the sake of time.

5.3 Step 0.3

Step 0.3 focuses on solving the energy equation using the velocity solution from Step 0.1 and the power density solution from Step 0.2 generated by the diffusion solver. The overall temperature

solution is found in Figure 5.7 while the midplane solutions are found in Figure 5.8.

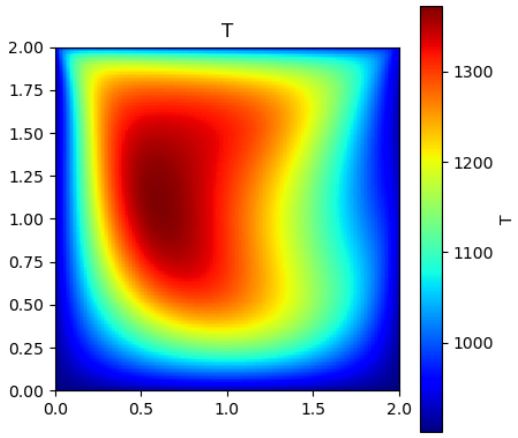


Figure 5.7: Overall temperature distribution for FiPy using the diffusion method.

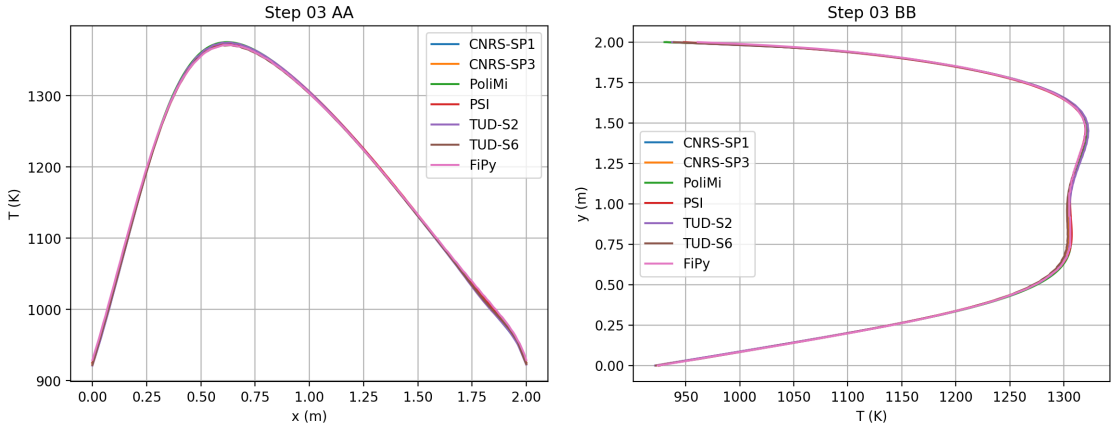


Figure 5.8: Temperatures along the horizontal and vertical midplanes for FiPy using the diffusion method compared to the other benchmark participants.

The temperature obtained from FiPy follows closely with the temperature distributions given by the benchmark participants as seen in the error plots in Figure 5.9. The average error was

0.16% for the horizontal midplane, AA, and 0.12% for the vertical midplane, BB. The low values for the error suggest FiPy is capable of solving the energy equations without issue and completes the verification for FiPy in performing single-physics modeling for MSRs.

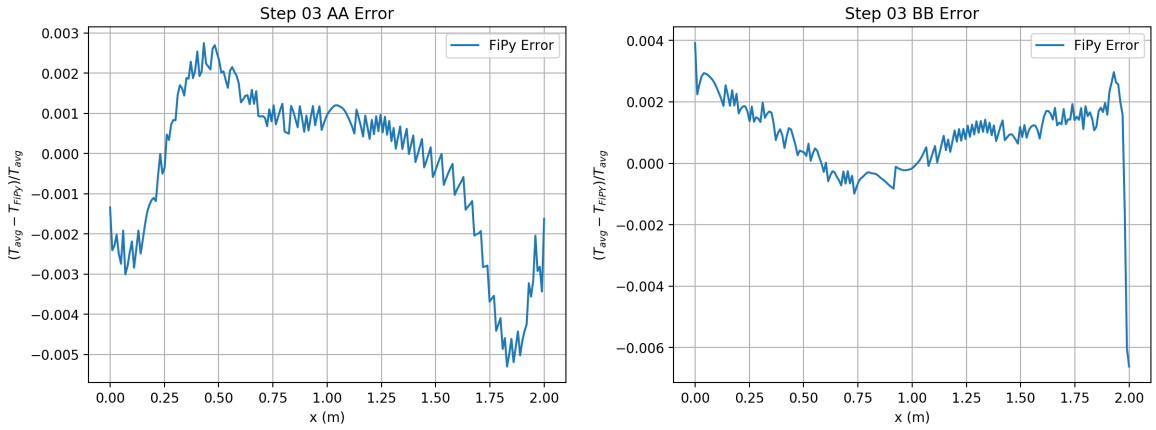


Figure 5.9: Error plots for temperature created by taking the difference in FiPy temperature and the average participant temperature normalized to the average temperature.

5.4 Step 1.1

Step 1.1 is the beginning of the multiphysics coupling steps and aims to couple the velocity solution of Step 0.1 to neutronics from Step 0.2 to model the DNP drift within the core. The quantities of interest for Step 1.1 are the DNP source, $\sum_i^I \lambda_i C_i$, and the change in reactivity from Step 0.2, $\rho_{1.1-0.2}$. Results are from a 200×200 uniform mesh using the diffusion solver in FiPy. The overall solution for the DNP source is given in Figure 5.10

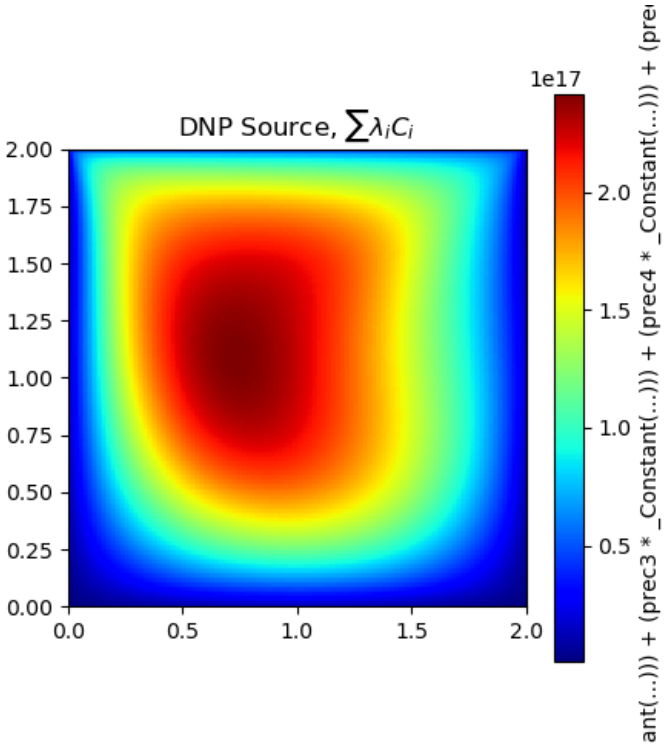


Figure 5.10: Overall DNP source distribution for FiPy using the diffusion method.

The overall shape for the DNP source distribution matches the shapes given by the benchmark participants. There are no instabilities in the simulation and the lowest values for the DNP source were on the boundary, which is expected for a vacuum boundary. For quantitative comparisons, the values of the DNP source along the horizontal and vertical midplanes were used for analyzing against the benchmarks as seen in Figures 5.11 and 5.12.

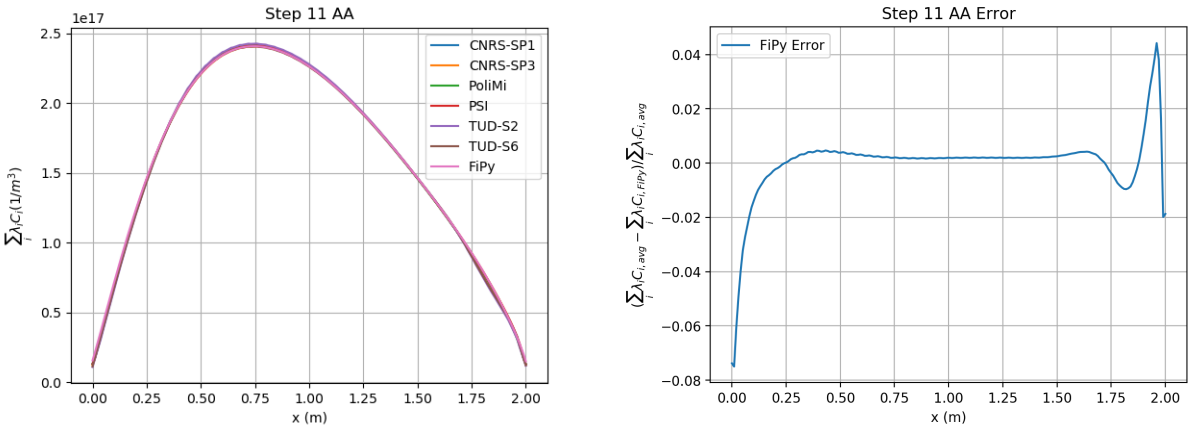


Figure 5.11: DNP source along the horizontal midplane AA for FiPy using the diffusion method compared to the other benchmark participants along with its associated error plot.

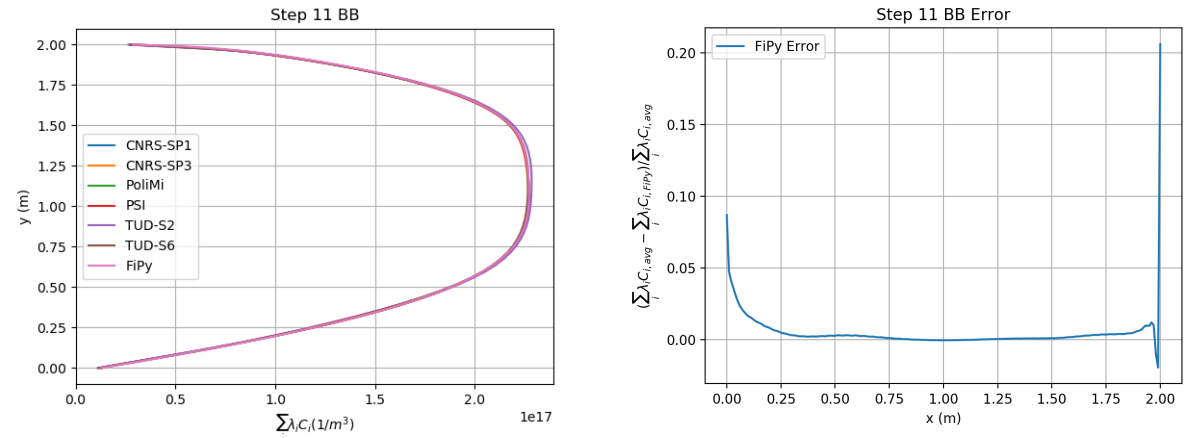


Figure 5.12: DNP source along the vertical midplane BB for FiPy using the diffusion method compared to the other benchmark participants along with its associated error plot.

The DNP source obtained in FiPy matches closely with the sources produced by other participants along the midplanes. The average error for the simulation along AA was 0.01% with a maximum of 8% at the boundary. For BB, the average error was 0.05% with a maximum error of 22% also at the boundary. This could be due to the data being provided as face elements versus

nodal elements at the boundary as well as different boundary implementations among the codes. The only notable difference is with TUD, however they used a finite element method for their discretization while FiPy uses finite volume. The differences in discretization methods and use of transport over diffusion are the major reasons for these minor discrepancies and can be considered an acceptable results for this simulation. The other quantity of interest for this benchmark was the change in reactivity from Step 0.2, $\rho_{1.1-0.2}$. The change in reactivity for FiPy from Step 0.2 along with the participants is shown in the Table 5.2.

Table 5.2: All Step 1.1 reactivity changes.

Code	$\rho_{1.1-0.2}$ (pcm)
FiPy SP_1	-64.5
CNRS-SP_1	-62.5
CNRS-SP_3	-62.6
PoliMi	-62.0
PSI	-63.0
TUD-S_2	-62.0
TUD-S_6	-60.7

The average benchmark participant reactivity was -62.1 pcm while FiPy was 3.4 pcm below the average. This is an acceptable result as it is less than 4% off of the average and follows closely with the other participant results. This suggests FiPy is capable of handling DNP convection problems where the diffusion equation is solved in the presence of a velocity field.

5.5 Step 1.2

Step 1.2 incorporates temperature feedback from the energy equation to the neutronics equation in the presence of the velocity field from Step 0.1. The velocity field from Step 0.1 is used

within the convection terms in the temperature and neutronics equations however no feedback from temperature or neutronics are imposed on the velocity equation. This simulation performs only two-way coupling between the temperature equation and the diffusion equation in the form of density feedback on the cross sections and power feedback from neutronics on the temperature equation. The quantities of interest for this simulation are the temperature along the midplanes and the change in fission rate density from Step 0.2.

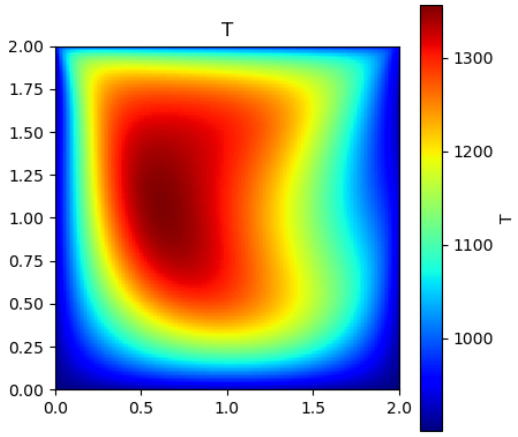


Figure 5.13: Overall temperature distribution in Kelvin for FiPy using the diffusion method.

The temperature solution in Figure 5.13 follows the overall shape of the other benchmark participants and is similar to the temperature distribution seen in Step 0.3. For the midplanes, the FiPy temperature and fission rate density solutions also follow closely with the results of the other benchmark participants as seen in 5.14.

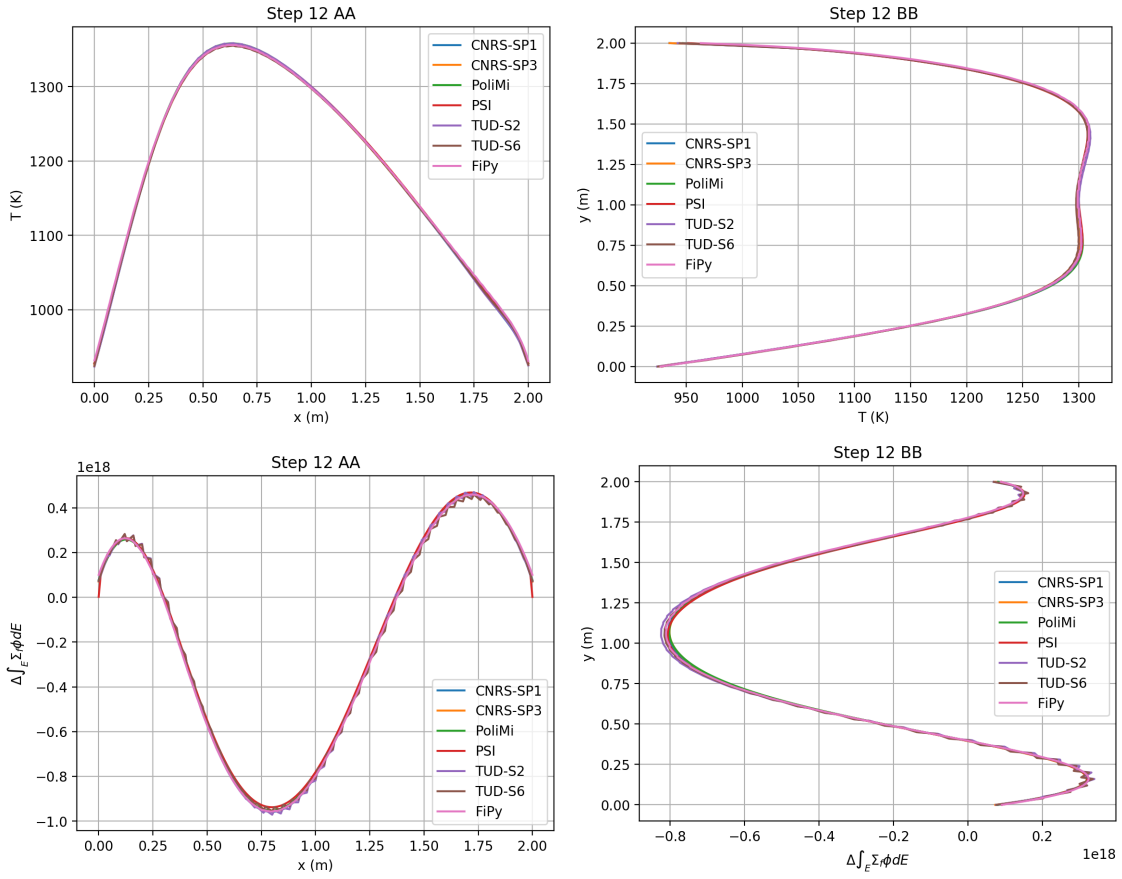


Figure 5.14: Temperature along the horizontal and vertical midplanes for FiPy using the SP_1 method compared to the other benchmark participants.

The average error for the temperatures were 0.1% and 0.08% for midplanes BB and AA respectively. For the change in fission rate density from Step 1.1 to 1.2, the average error was 0.6% and 0.7% respectively. The error was calculated by taking the difference in the FiPy value against the average benchmark and normalizing to the average benchmark value. These average errors suggest excellent agreement between FiPy and the benchmarks for performing temperature feedback effects on neutronics. The last quantity of interest for this problem is the reactivity, which is displayed in the Table 5.3.

Table 5.3: Display of the reactivity change from Step 1.1 to 1.2 recorded for each benchmark participant.

Code	$\rho_{1.2-1.1}$
FiPy SP_1	-1151.1
CNRS-SP_1	-1152.0
CNRS-SP_3	-1152.7
PoliMi	-1161.0
PSI	-1154.8
TUD-S_2	-1145.2
TUD-S_6	-1122.0

FiPy has excellent agreement with the other benchmarks in predicting the reactivity for a coupled temperature and neutronics case, having an error of 0.2% off of the average reactivity.

5.6 Step 1.3

Step 1.3 solves the fully coupled problem without the use of a lid velocity. This includes the Boussinesq approximation for buoyancy. The quantities of interest for this simulation were the x- and y-velocity, temperature, and the DNP source values along the centerlines along with the eigenvalue. Results were produced using a 200×200 uniform mesh and the SP_1 method. Overall results are shown in Figure 5.15 while the AA and BB midplane results are shown in Figures 5.16 and 5.17 respectively.

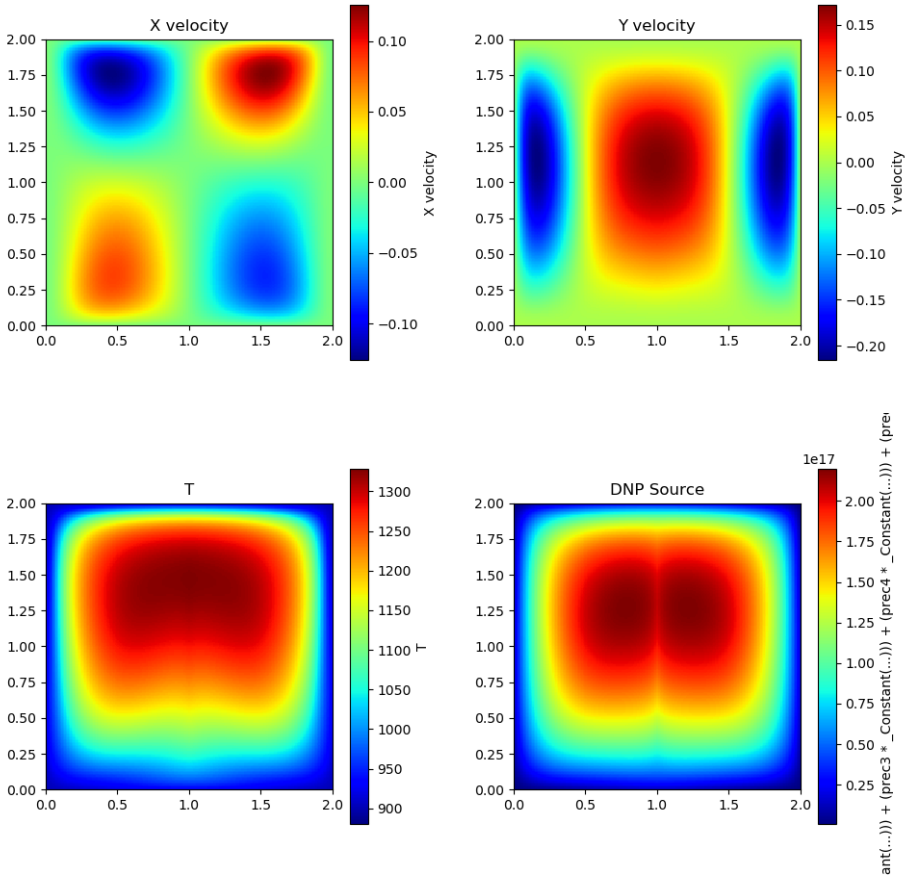


Figure 5.15: Velocity, temperature, and DNP source solutions for FiPy along the entire domain using the SP_1 method.

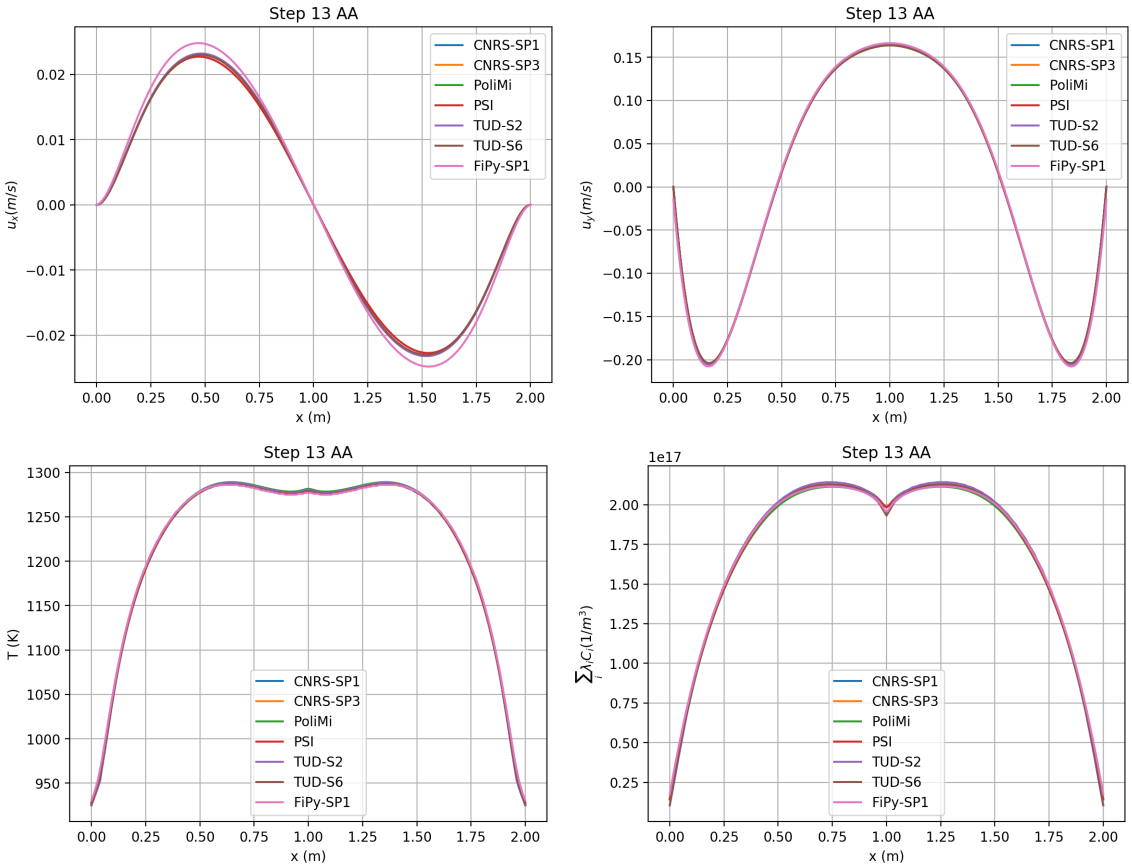


Figure 5.16: Velocity, temperature, and DNP source solution along the horizontal midplane AA for benchmark participants and FiPy SP_1 method.

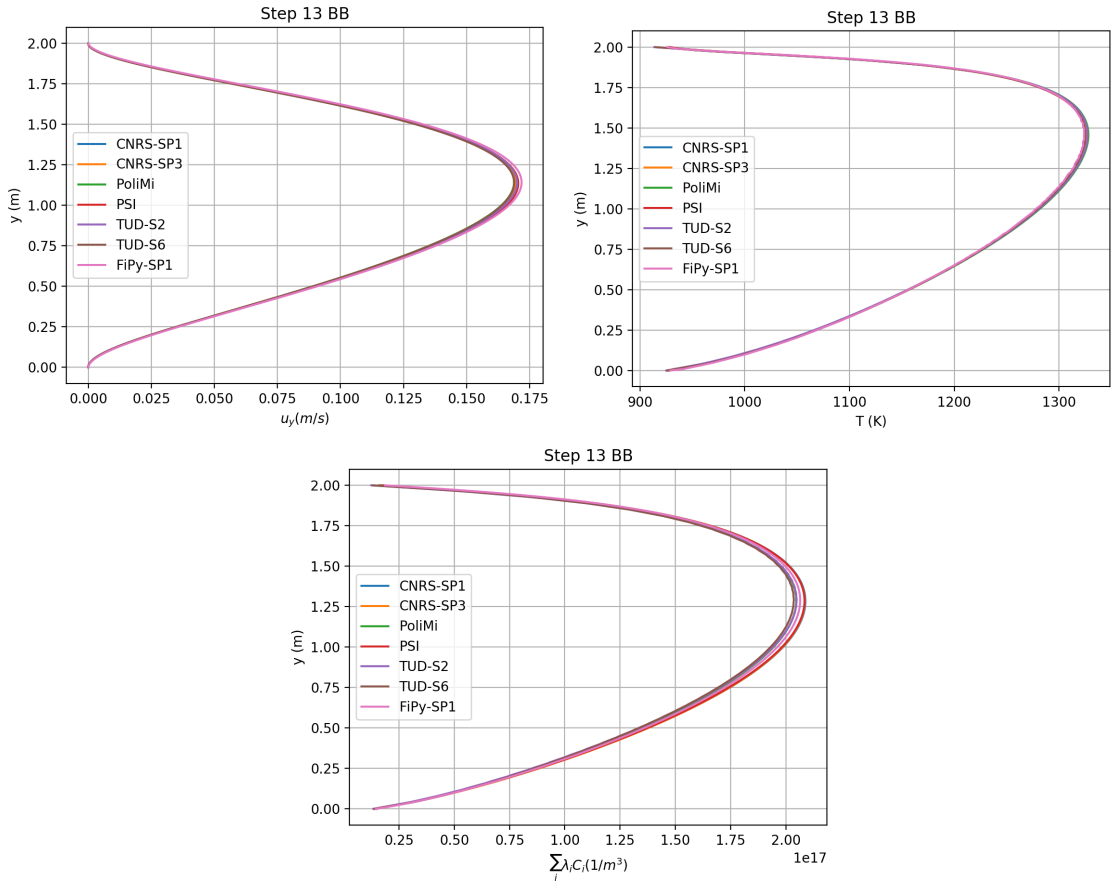


Figure 5.17: Velocity, temperature, and DNP source solution along the vertical midplane BB for benchmark participants and FiPy SP_1 method.

The maximum average error was 1.6% for the y-velocity along BB while all other average errors were below this value. This gives very good agreement with the average value of the benchmark participants for all the quantities of interest in this step. A small discrepancy of note is in the plot for the x-velocity along AA, where the maximum and minimum velocity does not completely align with the other benchmarks. This discrepancy is caused by a difference in the application of boundary conditions for FiPy in solving the buoyancy problem. The average benchmark value at these extrema were $\pm$ 0.023039 m/s while for FiPy the value was $\pm$ 0.024798 m/s. This is an error of 7.6% at the extrema for FiPy off of the benchmark values. Considering the small quantity for the velocity at these locations, this error value is acceptable and should not have a large effect

on the overall solution. As a measure of the overall solution, the eigenvalue can characterize the multiplication of neutrons within the reactor and also served as a quantity of interest, with results shown in Table 5.4

Table 5.4: Display of the reactivity change from Step 0.2 to 1.3 recorded for each benchmark participant.

Code	$\rho_{1.3-0.2}$
FiPy SP_1	-1219.2
CNRS-SP_1	-1220.5
CNRS-SP_3	-1220.7
PoliMi	-1227.0
PSI	-1219.6
TUD-S_2	-1208.5
TUD-S_6	-1184.4

The eigenvalue (converted to reactivity) for FiPy matched closely with the other benchmark participants and conveys that the overall solution for FiPy is accurate. The value of -1219.2 pcm was different from the average eigenvalue of -1213.45 by less than 6 pcm. This is an error of 0.47%. FiPy has performed well for the multiphysics coupling stage of the benchmark and can continue forward to do a full multiphysics problem. The previous reactivity values obtained in FiPy for a 200×200 mesh are summarized in Table 5.5 against the benchmark participant's reactivities.

Table 5.5: Display of the reactivities recorded for each step of the MSR Benchmark.

	Reactivity change, ρ (pcm)			
Code	$\rho_{0.2}$	$\rho_{1.1-0.2}$	$\rho_{1.2-1.1}$	$\rho_{1.3-0.2}$
FiPy SP_1	421.2	-64.5	-1151.1	-1219.
CNRS-SP_1	411.3	-62.5	-1152.0	-1220.5
CNRS-SP_3	353.7	-62.6	-1152.7	-1220.7
PoliMi	421.2	-62.0	-1161.0	-1227.0
PSI	411.7	-63.0	-1154.8	-1219.6
TUD-S_2	482.6	-62.0	-1145.2	-1208.5
TUD-S_6	578.1	-60.7	-1122.0	-1184.4

FiPy has proven itself capable of performing steady-state multiphysics problems thus far.

5.7 Step 1.4

Step 1.4 is a fully-coupled steady-state simulation that varies power and lid-velocity to record the changes in the eigenvalue as the parameters are varied. This was done in order to detect any systematic biases in the different codes. The reactivities were recorded for powers of 0.2 GW and 1 GW and lid velocities of 0.0 and 0.5 m/s in FiPy to compare against the other benchmark values as seen in Table 5.6.

Table 5.6: Reactivities obtained by FiPy compared to the other benchmark participant reactivities for Step 1.4.

Code		$\rho - \rho_{s_{0.2}}$ (pcm)	
		P (GW)	
	U_{lid} (m s^{-1})	0.2	1.0
FiPy	0.0	-266.3	-1219.2
CNRS-SP_1		-264.5	-1220.5
CNRS-SP_3		-265.8	-1220.7
PoliMi		-266.0	-1227.0
PSI		-268.0	-1215.1
TUD-S_2		-263.7	-1208.5
TUD-S_6		-258.0	-1184.4
FiPy	0.5	-276.6	-1203.8
CNRS-SP_1		-276.5	-1204.8
CNRS-SP_3		-276.8	-1205.2
PoliMi		-284.0	-1214.0
PSI		-278.1	-1199.8
TUD-S_2		-273.1	-1193.0
TUD-S_6		-267.6	-1169.7

FiPy shows excellent agreement with the other benchmark codes for fully coupled steady state problems in handling different reactor powers and velocities. At lower power, the effects of temperature on the eigenvalue are not as strong due to less feedback on the cross sections and movement of the precursors. At higher power, the reactivity decreases by a noticeable amount from the effect of power feedback on the temperature equation, which in turn increases the buoyancy and density feedback on velocity and neutronics respectively. The introduction of a lid velocity seems to have

smaller effects on the eigenvalue compared to the reactor power as it seems DNP drift does not affect the eigenvalue as harshly as power feedback does.

The differences between FiPy and the other codes is always limited to under 35 pcm compared to other benchmarks and shows that FiPy is capable of handling full multiphysics coupling. With FiPy verified for performing steady-state multiphysics coupling problems in both obtaining the eigenvalues and reactor parameters, the code can be extended to include burnup as a feedback mechanism on the eigenvalue.

5.8 Burnup Results

As described previously, cross sections were generated in Serpent for the MSR cavity at total burnup times of 0.0, 1.0, 10.0, 20.0, 50.0, and 100.0 MWd/kg. At each of these burnup values, the cross sections from group constant generation in Serpent were read into FiPy in order to do multiphysics coupled steady-state problems. Each steady state problem would yield the eigenvalue at the burnup and display the decrease in reactivity for the MSR as its fuel becomes depleted of fissionable material.

The thermal hydraulics parameters for the burnup simulation matched those of the 1 GW and 0.5 m/s lid-velocity case performed in Step 1.4 while cross sections were changed to match each burnup step. The k-effective values from FiPy were recorded in Table 5.7 along with its conversion to reactivity in terms of pcm and deviance from the reactivity recorded in Step 1.4. A display of the k-effective results for the burnup simulations are shown in Figure 5.18.

Table 5.7: K-effective and reactivity as a function of burnup recorded from the FiPy simulations.

Burnup (MWd/kg)	k-effective	ρ	$\rho_{bu-1.4}$ (pcm)
0.0	0.992235	-782.6	0.0
1.0	0.985080	-1514.6	-732.0
10.0	0.979577	-2084.9	-1302.3
20.0	0.973192	-2754.7	-1972.1
50.0	0.953647	-4860.6	-4078.0
100.0	0.918806	-8836.9	-8054.3

The fuel depletes quickly for the 6 month period set for the burnup calculation however this makes sense given the higher power density used for this benchmark compared to the actual MSFR design. In keeping with the spirit of the benchmark, the power was set to 1 GW in a 4 m^3 domain, giving a power density of 250 $\frac{MW}{m^3}$. The proposed design parameters for the MSFR uses a power of 3 GW in an 18 m^3 fuel cavity, thus changing the power density to a nominal 166.67 $\frac{MW}{m^3}$. The goal of this work, however, is to test the capabilities of using an open source FV code for performing both multiphysics coupling and including burnup calculations so this is an acceptable trade-off in testing the code rather than modeling a real reactor.

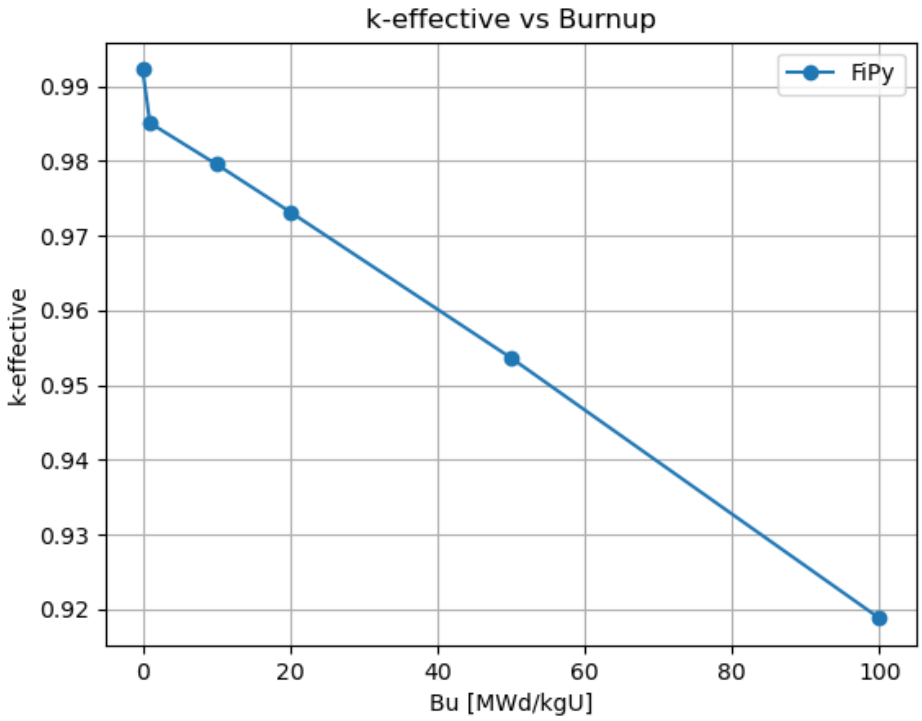


Figure 5.18: Display of the k-effective value produced as a function of steady-state burnup calculations.

6. CONCLUSIONS

In the work presented, the feasibility of using open-source finite volume codes such as FiPy was demonstrated in performing the modeling of multiphysics present in MSRs. The FiPy code performed very well in matching the reactor parameter outputs given by the CNRS benchmarks and disparity was very small between the different codes. FiPy proved itself capable of handling the different physics seen in each step of the benchmarks and could be set up to extend its methods to include other reactor physics present in MSFRs. Furthermore, FiPy showed the ability to couple itself to the Monte Carlo code, Serpent in utilizing the cross sections generated from a burnup input to create a study on the effect of cross section changes on the thermal hydraulic and neutronics parameters.

Together, with burnup calculations, the ability to model the full physics of an MSR is one step closer to becoming a reality. In adding burnup, the ability to predict the k-effective value of the reactor during its operational period gives insight into how long a fuel cycle can last in an MSR, albeit at a higher power density than the actual design contains. Introducing burnup also allows the effects of isotope depletion on the thermal hydraulics aspects of the code to be studied and see how the reduction in fissionable material affects these quantities of interest.

The main goal of this work was to create a multiphysics model for MSFR reactor physics that was based on a FV discretization. In this work the FiPy FV code was verified against the international benchmark data provided by the CNRS benchmark. This model was created in FiPy to demonstrate the capabilities of open-source software in performing the modeling of MSFR physics of the original benchmark while also extending it to include isotope depletion.

The FiPy model did have some drawbacks, however, in time it takes to model a simple benchmark simulation as some of the benchmark steps took many hours or even days before converging. The FiPy code also had some differences in approximations it used for applying boundary conditions that could be improved to match with the methods used by the benchmark participants. The use of python over C/C++ based codes showed some performance deficiencies but also allowed for

easier coding and could be used as a low level script for introductory reactor physics work before moving on to more complicated problems. Despite its drawbacks, FiPy still proved capable of solving the problems it was tasked with and displays the feasibility of using open-source codes for reactor physics problems.

Overall, the FiPy code proved functional for modeling MSFRs as well as introducing other important reactor physics necessary for the next steps in creating a simulation that captures the phenomena that occur in MSFRs. The work in this study could prove to be an important step in eventually building next-generation reactors.

6.1 Further Study

Any future study for this work would include reorganizing the benchmark to include different geometries closer to that of the actual MSFR and obtain results that would provide more insight into the real-world designs used in MSRs. Currently, the benchmark exists in a simplistic state for verifying the multiphysics portions and next would be including more realistic scenarios that MSRs would undergo during operation. Another item of interest would be including online fuel reprocessing to improve the accuracy of the cross sections as a function of burnup however in this work the addition of online reprocessing was neglected in order to test the capabilities of a simple burnup case. Online fuel reprocessing has already been implemented in Serpent [23] before and could possibly be used to extend this work.

REFERENCES

[1] "A technology roadmap for generation iv nuclear energy systems executive summary," March 2003.

[2] P. N. Haubenreich and J. R. Engel, "Experience with the molten-salt reactor experiment," *Nuclear Applications and Technology*, vol. 8, no. 2, pp. 118–136, 1970.

[3] E. Merle-Lucotte, M. Allibert, M. Brovchenko, D. Heuer, V. Ghetta, A. Laureau, and P. Rubiolo, "Introduction to the physics of thorium molten salt fast reactor (msfr) concepts," in *Thorium Energy for the World* (J.-P. Revol, M. Bourquin, Y. Kadi, E. Lillestol, J.-C. de Mestral, and K. Samec, eds.), (Cham), pp. 223–231, Springer International Publishing, 2016.

[4] T. Dolan, *Molten Salt Reactors and Thorium Energy*. Elsevier Science, 2017.

[5] H. Rouch, O. Geoffroy, P. Rubiolo, A. Laureau, M. Brovchenko, D. Heuer, and E. Merle-Lucotte, "Preliminary thermal–hydraulic core design of the molten salt fast reactor (msfr)," *Annals of Nuclear Energy*, vol. 64, pp. 449–456, 2014.

[6] M. Tiberga, R. G. G. de Oliveria, E. Cervi, J. A. Blanco, S. Lorenzi, M. Aufiero, D. Lathouwers, and P. Rubiolo, "Results from a multi-physics numerical benchmark for codes dedicated to molten salt fast reactors," *Annals of Nuclear Energy*, vol. 142, July 2020.

[7] M. Fratoni, D. Shen, J. Powers, and G. Ilas, "Molten salt reactor experiment benchmark evaluation," March 2020.

[8] M. Delpech, S. Dulla, C. Garzenne, J. Kophazi, J. Krepel, C. Lebrun, D. Lecarpentier, F. Mattioda, P. Ravetto, A. Rineiski, *et al.*, "Benchmark of dynamic simulation tools for molten salt reactors," in *Proceedings of the international conference GLOBAL*, pp. 2182–2187, 2003.

[9] J. E. Guyer, D. Wheeler, and J. A. Warren, "FiPy: Partial differential equations in Python," *Computing in Science and Engineering*, vol. 11, no. 3, pp. 6–15, 2009.

[10] Y. Zhang, J. C. Ragusa, and J. E. Morel, "Model error estimation for the simplified pn radiation transport equations," *Nuclear Science and Engineering*, vol. 194, no. 10, pp. 903–926, 2020.

[11] R. G. McClarren, "Theoretical aspects of the simplified pn equations," *Transport Theory and Statistical Physics*, vol. 39, no. 2-4, pp. 73–109, 2010.

[12] E. M. Gelbard, "Simplified spherical harmonics equations and their use in shielding problems," 2 1961.

[13] A. Vidal-Ferràndiz, S. González-Pintor, D. Ginestar, A. Carreño, and G. Verdú, "Optimized eigenvalue solvers for the neutron transport equation," in *Computational Science – ICCS 2018* (Y. Shi, H. Fu, Y. Tian, V. V. Krzhizhanovskaya, M. H. Lees, J. Dongarra, and P. M. A. Sloot, eds.), (Cham), pp. 823–832, Springer International Publishing, 2018.

[14] S. Patankar and D. Spalding, "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows," *International Journal of Heat and Mass Transfer*, vol. 15, no. 10, pp. 1787–1806, 1972.

[15] C. M. Rhie and W. L. Chow, "Numerical study of the turbulent flow past an airfoil with trailing edge separation," *AIAA Journal*, vol. 21, no. 11, pp. 1525–1532, 1983.

[16] J. Leppänen, M. Pusa, T. Viitanen, V. Valtavirta, and T. Kaltiaisenaho, "The serpent monte carlo code: Status, development and applications in 2013," *Annals of Nuclear Energy*, vol. 82, pp. 142–150, 2015.

[17] M. Pusa, "Rational approximations to the matrix exponential in burnup calculations," *Nuclear Science and Engineering*, vol. 169, no. 2, pp. 155–167, 2011.

[18] E. Cervi, S. Lorenzi, A. Cammi, and L. Luzzi, "Development of an sp3 neutron transport solver for the analysis of the molten salt fast reactor," *Nuclear Engineering and Design*, vol. 346, pp. 209–219, 2019.

[19] C. Fiorina, I. Clifford, M. Aufiero, and K. Mikityuk, "Gen-foam: a novel openfoam® based multi-physics solver for 2d/3d transient analysis of nuclear reactors," *Nuclear Engineering and Design*, vol. 294, pp. 24–37, 2015.

[20] M. Aufiero, A. Cammi, O. Geoffroy, M. Losa, L. Luzzi, M. E. Ricotti, and H. Rouch, "Development of an openfoam model for the molten salt fast reactor transient analysis," *Chemical Engineering Science*, vol. 111, pp. 390–401, 2014.

[21] OpenCFD, *OpenFOAM - The Open Source CFD Toolbox - User's Guide*. OpenCFD Ltd., United Kingdom, 1.4 ed., 11 Apr. 2007.

[22] M. Tiberga, D. Lathouwers, and J. Kloosterman, "A discontinuous galerkin fem multi-physics solver for the molten salt fast reactor," 08 2019.

[23] M. Aufiero, A. Cammi, C. Fiorina, J. Leppänen, L. Luzzi, and M. Ricotti, "An extended version of the serpent-2 code to investigate fuel burn-up and core material evolution of the molten salt fast reactor," *Journal of Nuclear Materials*, vol. 441, no. 1, pp. 473–486, 2013.

APPENDIX A

FIRST APPENDIX

Group, g	Σ_t (cm^{-1})	Σ_f (cm^{-1})
1	1.65512 $\times 10^{-1}$	1.11309 $\times 10^{-3}$
2	2.17253 $\times 10^{-1}$	1.08682 $\times 10^{-3}$
3	3.18009 $\times 10^{-1}$	1.52219 $\times 10^{-3}$
4	2.42093 $\times 10^{-1}$	2.58190 $\times 10^{-3}$
5	2.50351 $\times 10^{-1}$	5.36326 $\times 10^{-3}$
6	2.72159 $\times 10^{-1}$	1.44917 $\times 10^{-2}$

(a) Total and fission cross sections.

Group, g	Σ_t (cm^{-1})	Σ_f (cm^{-1})
1	1.65514 $\times 10^{-1}$	1.11307 $\times 10^{-3}$
2	2.17252 $\times 10^{-1}$	1.08683 $\times 10^{-3}$
3	3.18015 $\times 10^{-1}$	1.52216 $\times 10^{-3}$
4	2.42095 $\times 10^{-1}$	2.58339 $\times 10^{-3}$
5	2.50357 $\times 10^{-1}$	5.36790 $\times 10^{-3}$
6	2.72161 $\times 10^{-1}$	1.44925 $\times 10^{-2}$

(b) Burnup total and fission cross sections.

$\Sigma_{s,0,g'\to g}$ (cm^{-1})						
Group	g'					
g	1	2	3	4	5	6
1	1.08476e-1	0.0	0.0	0.0	0.0	0.0
2	5.23316e-2	1.83666e-1	0.0	0.0	0.0	0.0
3	4.01805e-3	3.19138e-2	2.98293e-1	0.0	0.0	0.0
4	1.09869e-4	2.34218e-5	1.63470e-2	2.17472e-1	0.0	0.0
5	2.53290e-5	2.25259e-6	1.70575e-5	1.90243e-2	2.27173e-1	0.0
6	3.78334e-6	2.00405e-7	1.24625e-6	1.36858e-8	1.05885e-2	2.37826e-1

(c) P_0 scattering cross sections.

$\Sigma_{s,0,g'\to g}$ (cm^{-1})						
Group	g'					
g	1	2	3	4	5	6
1	1.08247e-1	0.0	0.0	0.0	0.0	0.0
2	5.23440-2	1.83666e-1	0.0	0.0	0.0	0.0
3	4.01665e-3	3.19118e-2	2.98301e-1	0.0	0.0	0.0
4	1.10323e-4	2.32994e-5	1.63440e-2	2.17478e-1	0.0	0.0
5	2.53950e-5	2.21894e-6	1.70439e-5	1.90201e-2	2.27176e-1	0.0
6	3.78991e-6	2.01365e-7	1.25785e-6	1.49488e-8	1.05837e-2	2.37839e-1

(d) Burnup P_0 scattering cross sections.

$\Sigma_{s,1,g'\to g}$ (cm^{-1})						
Group	g'					
g	1	2	3	4	5	6
1	5.02479e-2	0.0	0.0	0.0	0.0	0.0
2	-5.31822e-3	3.78784e-2	0.0	0.0	0.0	0.0
3	4.77967e-4	-6.77085e-3	2.42287e-2	0.0	0.0	0.0
4	3.40533e-5	-1.92591e-6	-5.13491e-3	1.75032e-2	0.0	0.0
5	8.93761e-6	-6.99121e-8	8.27643e-7	-5.64363e-3	1.49908e-2	0.0
6	1.34789e-6	1.59216e-8	3.25237e-8	-5.05232e-10	-3.22691e-3	1.17041e-2

(e) P_1 scattering cross sections.

$\Sigma_{s,1,g'\to g}$ (cm^{-1})						
Group	g'					
g	1	2	3	4	5	6
1	5.02355e-2	0.0	0.0	0.0	0.0	0.0
2	-5.32354e-3	3.78716e-2	0.0	0.0	0.0	0.0
3	4.77603e-4	-6.76658e-3	2.42270e-2	0.0	0.0	0.0
4	3.43542e-5	-1.88876e-6	-5.13556e-3	1.75085e-2	0.0	0.0
5	8.73258e-6	-9.09299e-8	8.70552e-7	-5.64172e-3	1.49887e-2	0.0
6	1.34975e-6	1.51846e-8	2.39081e-8	-3.22659e-10	-3.22570e-3	1.17154e-2

(f) Burnup P_1 scattering cross sections.

$\Sigma_{s,2,g' \to g}$ (cm^{-1})						
Group	g'					
g	1	2	3	4	5	6
1	2.92625e-2	0.0	0.0	0.0	0.0	0.0
2	1.42924e-3	1.34479e-2	0.0	0.0	0.0	0.0
3	8.34387e-5	-1.08011e-4	3.26562e-3	0.0	0.0	0.0
4	1.81137e-5	-1.22275e-7	-2.07083e-4	7.74424e-4	0.0	0.0
5	4.60830e-6	-1.36245e-8	2.24708e-7	-4.07402e-4	5.04809e-4	0.0
6	7.72548e-7	5.37827e-9	-5.79136e-8	-2.06625e-9	-1.82238e-4	3.21414e-4

(g) P_2 scattering cross sections.

$\Sigma_{s,2,g' \to g}$ (cm^{-1})						
Group	g'					
g	1	2	3	4	5	6
1	2.92639e-2	0.0	0.0	0.0	0.0	0.0
2	1.43035e-3	1.34404e-2	0.0	0.0	0.0	0.0
3	8.26441e-5	-1.07563e-4	3.27403e-3	0.0	0.0	0.0
4	1.81076e-5	-1.19965e-7	-2.05486e-4	7.74762e-4	0.0	0.0
5	4.55102e-6	-1.12013e-8	2.69513e-7	-4.06660e-4	5.07266e-4	0.0
6	7.06042e-7	3.81142e-9	-6.91259e-8	-1.02791e-10	-1.82967e-4	3.20851e-4

(h) Burnup P_2 scattering cross sections.

$\Sigma_{s,3,g' \to g}$ (cm^{-1})						
Group	g'					
g	1	2	3	4	5	6
1	1.43002e-2	0.0	0.0	0.0	0.0	0.0
2	-1.47138e-3	2.82616e-3	0.0	0.0	0.0	0.0
3	1.41136e-5	-1.27263e-4	2.81143e-4	0.0	0.0	0.0
4	4.86683e-6	-2.35861e-8	1.40428e-4	1.23373e-5	0.0	0.0
5	1.18151e-6	2.87008e-10	4.64989e-8	-8.90726e-6	3.24220e-6	0.0
6	1.98042e-7	7.05060e-9	-6.27160e-8	-3.97151e-10	6.27731e-7	8.75552e-6

(i) P_3 scattering cross sections.

$\Sigma_{s,3,g' \to g}$ (cm^{-1})						
Group	g'					
g	1	2	3	4	5	6
1	1.42979e-2	0.0	0.0	0.0	0.0	0.0
2	-1.47612e-3	2.82539e-3	0.0	0.0	0.0	0.0
3	1.28886e-5	-1.27571e-4	2.83753e-4	0.0	0.0	0.0
4	4.84952e-6	-1.36724e-8	1.39874e-4	1.48040e-5	0.0	0.0
5	1.16816e-6	3.08588e-9	8.77201e-8	-9.39920e-6	1.66775e-6	0.0
6	1.84472e-7	2.10530e-9	-6.08477e-8	-1.50847e-10	8.50214e-7	1.67820e-6

(j) Burnup P_3 scattering cross sections.

Group, g	D_g (cm)	$1/v_g$ ($s\ cm^{-1}$)
1	2.80064	4.00367 $\times 10^{-10}$
2	1.84021	7.39846 $\times 10^{-10}$
3	1.13110	2.61748 $\times 10^{-9}$
4	1.44786	6.69270 $\times 10^{-9}$
5	1.39750	1.55845 $\times 10^{-8}$
6	1.28252	4.24462 $\times 10^{-8}$

(k) Diffusion coefficients and inverse neutron velocities.

Group, g	D_g (cm)	$1/v_g$ ($s\ cm^{-1}$)
1	2.80023	4.00376 $\times 10^{-10}$
2	1.84020	7.39830 $\times 10^{-10}$
3	1.13107	2.61734 $\times 10^{-9}$
4	1.44789	6.69250 $\times 10^{-9}$
5	1.39745	1.55834 $\times 10^{-8}$
6	1.28257	4.24490 $\times 10^{-8}$

(l) Burnup diffusion coefficients and inverse neutron velocities.

Group, g	$\nu_{tot,g}$ (-)	$\chi_{p,g}$ (-)	$\chi_{d,g}$ (-)	E_{fiss} (J)
1	2.85517	3.53812 $\times 10^{-1}$	4.30325 $\times 10^{-3}$	3.240722 $\times 10^{-11}$
2	2.54532	5.23642 $\times 10^{-1}$	3.87734 $\times 10^{-1}$	3.240722 $\times 10^{-11}$
3	2.43328	1.21033 $\times 10^{-1}$	5.81848 $\times 10^{-1}$	3.240722 $\times 10^{-11}$
4	2.43127	1.35457 $\times 10^{-3}$	2.27947 $\times 10^{-2}$	3.240722 $\times 10^{-11}$
5	2.43330	1.51226 $\times 10^{-4}$	2.89130 $\times 10^{-3}$	3.240722 $\times 10^{-11}$
6	2.43330	7.37236 $\times 10^{-6}$	4.28935 $\times 10^{-4}$	3.240722 $\times 10^{-11}$

(m) Delayed and prompt neutron spectra..

Group, g	$\nu_{tot,g}$ (-)	$\chi_{p,g}$ (-)	$\chi_{d,g}$ (-)	E_{fiss} (J)
1	2.85511	3.53869 $\times 10^{-1}$	4.23079 $\times 10^{-3}$	3.240722 $\times 10^{-11}$
2	2.54532	5.23599 $\times 10^{-1}$	3.87924 $\times 10^{-1}$	3.240722 $\times 10^{-11}$
3	2.43329	1.21017 $\times 10^{-1}$	5.81918 $\times 10^{-1}$	3.240722 $\times 10^{-11}$
4	2.43127	1.35368 $\times 10^{-3}$	2.27044 $\times 10^{-2}$	3.240722 $\times 10^{-11}$
5	2.43330	1.52377 $\times 10^{-4}$	2.78685 $\times 10^{-3}$	3.240722 $\times 10^{-11}$
6	2.43330	7.86800 $\times 10^{-6}$	4.35542 $\times 10^{-4}$	3.240722 $\times 10^{-11}$

(n) Burnup delayed and prompt neutron spectra.

Family, i	λ_i (s^{-1})	β_i (-)
1	1.24667 $\times 10^{-2}$	2.33102 $\times 10^{-4}$
2	2.82917 $\times 10^{-2}$	1.03262 $\times 10^{-3}$
3	4.25244 $\times 10^{-2}$	6.81878 $\times 10^{-4}$
4	1.33042 $\times 10^{-1}$	1.37726 $\times 10^{-3}$
5	2.92467 $\times 10^{-1}$	2.14493 $\times 10^{-3}$
6	6.66488 $\times 10^{-1}$	6.40917 $\times 10^{-4}$
7	1.63478 $\times 10^{0}$	6.05805 $\times 10^{-4}$
8	3.55460 $\times 10^{0}$	1.66016 $\times 10^{-4}$

(o) Fraction and decay constant for each family of delayed neutron precursors.

Table A.1: Cross sections used for benchmark.