ABSTRACT

ELLIS, JOHN AUSTIN. Performant Hybrid and Parallel Domain Decomposed Monte Carlo Methods for Radiation Transport. (Under the direction of C.T. Kelley).

This document further develops performant algorithms for massively parallel, hybrid Monte Carlo methods for radiation transport. New algorithms are implemented in the Shift Monte Carlo code developed at Oak Ridge National Laboratory. We provide the governing neutron transport equation and long-standing solution techniques. First, we explore the convergence of two acceleration techniques when the fixed point maps are corrupted with stochastic noise: Anderson Acceleration and Nonlinear Diffusion Acceleration. Next, we consider the Monte Carlo algorithm for solving the transport equation. It simulates a finite number of particle *histories* from known probability distribution functions inside a given domain. It avoids discretization error associated with deterministic algorithms, but requires a significant number of random samples. The Shift code is targeted at leadership class high performance computing platforms, like the Titan and Summit supercomputers at the Oak Ridge Leadership Computing Facility.

Domain decomposition is necessary when the problem's memory footprint grows too large to fit on a single high performance compute node. We diagnose a load imbalance problem that typically arises with domain decomposed calculations and propose as solution a nonuniform allocation of processors across subdomains. We optimize the allocation with runtime diagnostics collected during a calibration step, then complete the full calculation. We demonstrate the effectiveness and robustness of nonuniform processor allocation and its optimization on three 3D radiation transport applications, including a simulation of the Watts Bar Nuclear 1 initial start up reactor. We conclude with discussion on the future of high performance computing within the Department of Energy, particularly continued emphasis on GPU computing, through the highly-heterogenous Summit machine and the soon-to-come Frontier machine. We also provide potential interesting optimization problems that may arise in a GPU implementation of domain decomposed Monte Carlo algorithms.

Performant Hybrid and Parallel Domain Decomposed Monte Carlo Methods for Radiation Transport

by
John Austin Ellis

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2018

APPROVED BY:

Dmitriy Anistratov	Steven P. Hamilton
Ralph C. Smith	C.T. Kelley Chair of Advisory Committee

DEDICATION

To human achievement and Lee Mellon.

BIOGRAPHY

The author was born in the small mountain town of Asheville, North Carolina . . .

ACKNOWLEDGEMENTS

First, I want to thank all the math and science and humanities teachers I have had throughout my education. Each school has been quite unique and added to my view of nature and the world. For my degree, I was fortunate to work with the Exnihilo development team, an outstanding team at Oak Ridge National Laboratory. I especially want to thank Steven Hamilton for his generosity with his time while I was at Oak Ridge. It was always a great pleasure working together through the trials and tribulations of research. Most of all, I would like to thank my advisor, C.T. Kelley, for his great help. Now, time for me to get to work!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

Our research builds on the past successes in hybrid numerical techniques for solving the neutron transport equation, while making efficient use of state-of-the-art computing platforms available to researchers. The development of algorithmic techniques that maximize usage of new computational power will allow us to solve problems at scales that had no viable solutions ten years ago. The Department of Energy (DOE) and Oak Ridge National Laboratory (ORNL) have funded research through the Consortium for Advanced Simulation of Light-Water Reactors (CASL) and the Exascale Computing Project (ECP) to improve capabilities and performance of the massively parallel Shift Monte Carlo radiation transport code developed at ORNL. The Shift code is used for solving problems in a variety of nuclear applications, including reactor physics, shielding, irradiation target design, and threat detection [46]. The Shift development team is investigating algorithmic performance improvements on the leadership class platforms including advanced variance reduction techniques, efficient GPU implementations, and load balancing domain decomposed Monte Carlo calculations. The focus of this thesis is on our load balancing successes for domain decomposition that have led to robust increases in parallel efficiency with little additional overhead required.

In terms of tangible world impact, high-fidelity solutions in reactor simulations serve as benchmarks for industry partners. Due to the nature of the nuclear reactors, access to experimental data may be severely limited. This makes simulation one of the only viable options for test verification. Daily calculations are made in industry using more approximate methods and smaller computer clusters. It is CASL's mission to build a new reactor core simulator, The Virtual Environment for Reactor Applications (VERA), that is sensitive to industry computer resource and time constraints. In numerical analysis, comparing results to the known "true" solution is a basic principle. If the benchmark solutions are low-grade, then matching those solutions provides no useful information and, in worse circumstances, information that may be inadequate or even harmful. Higher fidelity benchmarks may be attainable if the

full potential of the DOE's leadership class supercomputers and future computing platforms are realized on high precision reactor simulations.

1.2 Neutron Transport Equation

A perennial goal of numerical nuclear physics is the solution to the linear integro-differential transport equation presented in [4, 13, 37]

$$\frac{1}{v(E)}\frac{\partial \psi}{\partial t} + \hat{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\hat{\Omega},E,t) + \Sigma_t(\vec{r},E)\,\psi(\vec{r},\hat{\Omega},E,t) = \int_0^\infty dE' \int_{4\pi} d\hat{\Omega}'\ \Sigma_s(\vec{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)\,\psi(\vec{r},\hat{\Omega}',E',t) + \frac{\chi(\vec{r},E,t)}{4\pi}\int_0^\infty dE' \int_{4\pi} d\hat{\Omega}'\ \nu(\vec{r},E',t)\,\Sigma_f(\vec{r},E',t)\,\psi(\vec{r},\hat{\Omega}',E',t) + Q(\vec{r},\hat{\Omega},E,t), \quad (1.1)$$

where the quantity of interest is the angular neutron flux ψ which depends on a phase space of the spatial vector as $\vec{r}$ within a defined domain volume $V \in \mathbb{R}^3$, the unit direction vector of particle movement $\hat{\Omega} \in \mathbb{R}^3$, particle energy $E \in [0,\infty)$, and time $t \in [0,\infty)$. The velocity of a particle is represented by a direction and an energy. We aim to solve (2.43) for ψ. If we define $d\ell = v(E)dt$ as the differential unit of track length, then

Definition 1.2.1. *$\psi(\vec{r},\hat{\Omega},E,t)\ d^3\vec{r}\ d\hat{\Omega}\ dE$ is the amount of neutron track length with velocity $v(E)$ in a differential volume $d^3\vec{r}$ about $\vec{r}$, traveling in $d\hat{\Omega}$ about $\hat{\Omega}$, having energies in dE around E, and at time t. ψ has units neutrons per cm^2 per seconds per steradians per MeV.*

The linear transport equation is a simplified form of the nonlinear Boltzmann equation for gas dynamics [37]. The collision kernel in the transport equation is linear, whereas for the Boltzmann equation it is nonlinear. In the nonlinear form, particles of interest may interact with themselves resulting in a quadratic term. In the linear form, particles of interest interact only with medium nuclei. We discuss further model assumptions in Chapter 2.

Each element in the model equation governs a different physical phenomenon for a neutron. Respective to (2.43), they are temporal change, particle streaming, total collision, scattering production, fission production, and external source. *Cross sections* are the coefficients that depend on a phase space of location, angle, energy, and time with each term. They are predetermined material dependent constants that appropriately weight the probability of events occurring in the model equation.

The *mean free path* (mfp) is the average distance a particle travels between interactions. The $mfp = 1/\Sigma_t(\vec{r},E)$, where the total cross section $\Sigma_t(\vec{r},E)$ describes the probability of a neutron undergoing any type of "collision" at the location $\vec{r}$ with neutron energy E. The scattering cross section $\Sigma_s(\vec{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)$ describes the probability of a neutron undergoing a state-change from an initial energy state E' to the final state E, noted by $E'\to E$ and from initial angle to final angle, $\hat{\Omega}'\to\hat{\Omega}$.

There are three types of energy state-change.

Definition 1.2.2. Up-scattering *occurs when a neutron initial energy E transitions to a higher final energy state E'.*

Definition 1.2.3. Down-scattering *occurs when a neutron initial energy E transitions to a lower final energy state E'.*

Definition 1.2.4. In-scattering *occurs when a neutron initial energy E remains in the same energy state, or $E = E'$.*

The fission cross section Σ_f describes the probability of a neutron spawning additional neutrons during a portion of absorption processes in fissile material. The absorption cross section Σ_a describes the probability of a neutron leaving the system by undergoing absorption from a nuclide during a collision. Absorption of neutrons by a nuclei is implicitly described in the model equation as $\Sigma_a = \Sigma_t - \Sigma_s$.

The probabilistic energy spectrum occurring from fission χ and ν is the average number of neutrons birthed per fission. The function Q is a known external source depending on the same phase space as the angular flux ψ. When integrating ψ over angle $\hat{\Omega}$, we use dimensionless unit *steradians*. First, the components of $\hat{\Omega} = \{\Omega_x, \Omega_y, \Omega_z\}$ are

$$\Omega_x = \sin\theta\cos\varphi; \quad \Omega_y = \sin\theta\sin\varphi; \quad \Omega_z = \cos\theta \tag{1.2}$$

where θ is the polar angle and φ is the azimuthal angle. This two dimensional angle in $\mathbb{R}^3$ space is a solid angle and its integration is

$$\int_{4\pi} \cdot \, d\hat{\Omega} \equiv \int_0^{2\pi} \int_0^{\pi} \cdot \, \sin(\theta)\, d\theta d\varphi. \tag{1.3}$$

From solid angle integration, we find the scalar flux, ϕ.

Definition 1.2.5. *ϕ is the scalar flux. The scalar flux is the angle independent quantity that describes the total amount of neutron track length moving with velocity $v(E)$ in volume $d^3\vec{r}$ about $\vec{r}$, having energies in dE around E, and at time t.*

The scalar flux is

$$\phi(\vec{r}, E, t) = \int_{4\pi} \psi(\vec{r}, \hat{\Omega}, E, t)\, d\hat{\Omega}. \tag{1.4}$$

The partial differential equation has one spatial and one temporal derivative, so it requires a boundary condition and an initial condition. We define ψ on the boundary ∂A as

$$\psi(\vec{r}_b, \hat{\Omega}, E, t) = \psi_b(\vec{r}_b, \hat{\Omega}, E, t), \ \vec{r}_b \in \partial A; \ \hat{\Omega} \cdot \hat{e}_n < 0 \tag{1.5}$$

where $\hat{e}_n$ is the outward unit normal of the domain surface. We assume all boundaries are *non-reentrant*, or that no particles return to the domain once exiting.

The initial condition is

$$\psi(\vec{r}, \hat{\Omega}, E, 0) = \psi_0(\vec{r}, \hat{\Omega}, E). \tag{1.6}$$

We define the loss and production operators as

$$\mathcal{L}\psi = \hat{\Omega} \cdot \vec{\nabla}\psi(\vec{r}, \hat{\Omega}, E, t) + \Sigma_t(\vec{r}, E)\psi(\vec{r}, \hat{\Omega}, E, t) \tag{1.7}$$

and

$$\mathcal{P}\psi = \int_0^\infty dE' \int_{4\pi} d\hat{\Omega}' \ \Sigma_s(\vec{r}, \hat{\Omega}' \to \hat{\Omega}, E' \to E)\psi(\vec{r}, \hat{\Omega}', E', t) + \frac{\chi(\vec{r}, E, t)}{4\pi} \int_0^\infty dE' \int_{4\pi} d\hat{\Omega}' \ \nu(\vec{r}, E', t)\Sigma_f(\vec{r}, E', t')\psi(\vec{r}, \hat{\Omega}', E', t). \tag{1.8}$$

We take

$$\psi = \psi(\vec{r}, \hat{\Omega}, E, t)$$

and

$$Q = Q(\vec{r}, \hat{\Omega}, E, t),$$

to obtain the transport equation in compact form

$$\frac{1}{v}\frac{\partial \psi}{\partial t} + \mathcal{L}\psi = \mathcal{P}\psi + Q. \tag{1.9}$$

Chapter 2

The Neutron Transport Equation Model

2.1 Model Assumptions

The linear Boltzmann transport equation is used to model the statistical behavior of various particles being transported through a medium. It is important to understand its derivation and the grounding assumptions before we continue. The Boltzmann equation is only a model of reality and with each model comes certain conditional assumptions. We discuss these long-standing and well-defined assumptions all in the context of neutron transport [4, 12, 13], but other particles may also be applicable.

Assumption 2.1.1. Particle represented as point particle:
The type of particles transported must be considered point particles, and so must be fully described by position $\vec{r}$ and velocity $\vec{v}$.

Assumption 2.1.2. Size of particle small relative to size of system:
The physical width of the particle must be far less than the domain width.

For example, the width of a neutron $\lambda\!\!\!^{-} \approx 4.55 \cdot 10^{-9}$cm, while a standard Westinghouse $17x17$ fuel assembly is approximately 21.4 cm [5].

Assumption 2.1.3. Size of particle small relative to distance between collisions:
The physical width of the particle must be far less than the distance between successive collisions, the mean free path. It is $1/\Sigma_t(\vec{r}, E)$, a positive length that is dependent on domain media and particle energy.

For example, the mean free path of a neutron with energy 10^6eV in a uniform water medium is approximately 2.42cm, which again is far larger than than neutron width $\lambda\!\!\!^{-}$. We calculated these values using the Nemesis package inside the Scale code developed by Oak Ridge National Lab [1].

Assumption 2.1.4. Particles only collide with nuclides of medium:
Particle-particle collisions are assumed to be negligible because of similar size and low-probability arguments.

We assume no nuclide-nuclide interactions occur. This assumption allows us to reduce the nonlinear Boltzmann equation to the linear transport equation.

Assumption 2.1.5. System must contain a large quantity of particles:
The system must have enough particles so that deviations from the expected distribution are small and may be ignored.

Assumption 2.1.6. Medium in equilibrium state:
Particles do not influence the distribution of the moderating material.

If the model satisfies the assumptions, the model transport equation accurately reflects real world phenomena within a small level of uncertainty. In the case of the neutron transport equation, the necessary model assumptions are satisfied and make no further appearances in our future analyses.

2.2 Derivation of the Transport Equation

2.2.1 Conservation of Mass

The derivation of the transport equation [12] takes an arbitrary volume V and balances the loss $\mathcal{L}$ and production $\mathcal{P}$ of neutrons within V. Analogous to ψ, the neutron distribution function is $n(\vec{r}, \hat{\Omega}, E, t)$. The two are related by the equation $\psi(\vec{r}, \hat{\Omega}, E, t) = v(E)n(\vec{r}, \hat{\Omega}, E, t)$.

The number of neutrons at a specific time t in the arbitrary volume within the energy interval dE and the angle $d\hat{\Omega}$ is

$$\left(\int_V n(\vec{r}, \hat{\Omega}, E, t)d^3\vec{r}\right)d\hat{\Omega}\, dE, \tag{2.1}$$

which is an scalar quantity, because $d\hat{\Omega}$ and dE are both differentials and $n(\vec{r}, \hat{\Omega}, E, t)$ is a density depending on angle and energy. The change rate in the number of neutrons in the volume V and phase space differentials $d\hat{\Omega}, dE$ is

$$\frac{\partial}{\partial t}\left(\int_V n(\vec{r}, \hat{\Omega}, E, t)d^3\vec{r}\right)d\hat{\Omega}\, dE = \text{Neutron Production in } V - \text{Neutron Loss in } V. \tag{2.2}$$

We assume the volume has fixed boundaries and does not depend on time. We have the equality

$$\frac{\partial}{\partial t}\left(\int_V n(\vec{r}, \hat{\Omega}, E, t)d^3\vec{r}\right)d\hat{\Omega}\, dE = \left(\int_V \frac{\partial n}{\partial t}(\vec{r}, \hat{\Omega}, E, t)d^3\vec{r}\right)d\hat{\Omega}\, dE. \tag{2.3}$$

We must quantify *Neutron Production* and *Neutron Loss* in the righthand side of (2.2). The possible additions and subtractions of neutrons in volume V include:

Neutron Production Rate in Volume V:

1. *Neutron sources within the volume V from an external source Q.*
2. *Neutrons streaming into volume V through the surface S within angle* $d\hat{\Omega}$ *and within energy dE.*
3. *Neutrons scattering in volume V from incident angle* $\hat{\Omega}'$ *and original energy* E' *to values of interest* $\hat{\Omega}$ *and E. We write as* $(\hat{\Omega}' \rightarrow \hat{\Omega}, E' \rightarrow E)$.
4. *Neutron sources within the volume V due to fission.*

Neutron Loss Rate in Volume V:

5. *Neutrons streaming out of the volume V through the surface S with angle* $d\hat{\Omega}$ *and energy dE.*
6. *Neutrons undergoing collision in volume V. This term covers both absorption collision and scattering collisions from* $(\hat{\Omega} \rightarrow \hat{\Omega}', E \rightarrow E')$ *which are both considered losses.*

We provide expressions for each case that form the model equation.

1. **Neutron source rates within the volume** V**:**
The general source term is $Q(\vec{r}, \hat{\Omega}, E, t)\, d^3\vec{r}\, d\hat{\Omega}\, dE$.The source contribution in V is

$$\left(\int_V Q(\vec{r}, \hat{\Omega}, E, t)\, d^3\vec{r} \right) d\hat{\Omega}\, dE. \tag{2.4}$$

2&5. **Neutrons streaming into and out of volume** V**:**
The *leakage* is the amount of neutrons streaming into or out of the system. We define the angular current density as

$$J(\vec{r}, \hat{\Omega}, E, t) = v(E)\, \hat{\Omega}\, n(\vec{r}, \hat{\Omega}, E, t) \cdot dS. \tag{2.5}$$

The angular current density describes the rate of neutrons that leak out of the surface differential dS in angle $d\hat{\Omega}$ and energy dE. We integrate over the entire surface to obtain

$$\int_S dS \cdot v(E)\, \hat{\Omega}\, n(\vec{r}, \hat{\Omega}, E, t), \tag{2.6}$$

which is equivalent to

$$\int_V d^3\vec{r}\, \nabla \cdot \Big(v(E)\, \hat{\Omega}\, n(\vec{r}, \hat{\Omega}, E, t) \Big) \tag{2.7}$$

from the divergence theorem. Consider $v(E)$ and $\hat{\Omega}$ to be independent of location $\vec{r}$; therefore,

$$\nabla \cdot v(E)\, \hat{\Omega} = v(E)\, \hat{\Omega} \cdot \nabla. \tag{2.8}$$

Thus, we obtain the final streaming volume integral

$$\left(\int_S dS \cdot v(E)\,\hat{\Omega}\,n(\vec{r},\hat{\Omega},E,t)\right)\,d\hat{\Omega}\,dE = \left(\int_V d^3\vec{r}\,v(E)\,\hat{\Omega}\cdot\nabla n(\vec{r},\hat{\Omega},E,t)\right)\,d\hat{\Omega}\,dE. \tag{2.9}$$

3. **Neutrons scattering in volume V from $(\hat{\Omega}' \to \hat{\Omega}, E' \to E)$:**
Remember, the coefficient describing scattering Σ_s depends on the four parameters $(\hat{\Omega}', \hat{,}E', E)$ to determine the appropriate double-differential scattering cross section. The contribution from one incident angle and original energy is

$$\left(\int_V d^3\vec{r}\,v(E')\,\Sigma_s(\hat{\Omega}'\to\hat{\Omega},E'\to E)\,n(\vec{r},\hat{\Omega},E,t)\right)\,d\hat{\Omega}\,dE. \tag{2.10}$$

Contributions can come from any $(\hat{\Omega}', E')$, so integrating over all angles and energies then the total contribution is

$$\left(\int_V d^3\vec{r}\int_0^\infty dE'\int_{4\pi} d\hat{\Omega}' v(E')\,\Sigma_s(\hat{\Omega}'\to\hat{\Omega},E'\to E)\,n(\vec{r},\hat{\Omega},E,t)\,d^3\vec{r}\right)\,d\hat{\Omega}\,dE. \tag{2.11}$$

4. **Neutron source in volume V due to fission:**
The number of neutrons produced by fission is the reaction rate of one neutron scaled by the total neutron count. The reaction rate for one neutron is $v(E)\Sigma_f(\vec{r},E,t)$. The fission cross section Σ_f describes the number of interactions per unit of path length. Also recall, we defined the average number of neutrons produced by fission as $\nu(\vec{r},E,t)$.Then, the neutron total produced in angle $d\hat{\Omega}$ and energy dE is

$$v(E)\,\nu(\vec{r},E,t)\,\Sigma_f(\vec{r},E,t)\,n(\vec{r},\hat{\Omega},E,t)\,d^3\vec{r}\,d\hat{\Omega}\,dE. \tag{2.12}$$

Integrate this quantity over angle, energy, and volume V to obtain

$$\left(\int_V d^3\vec{r}\int_{4\pi} d\hat{\Omega}'\int_0^\infty dE'\,v(E')\,\nu(\vec{r},E',t)\,\Sigma_f(\vec{r},E',t)\,n(\vec{r},\hat{\Omega},E,t)\right)\,d\hat{\Omega}\,dE. \tag{2.13}$$

(2.13) does not take into account the direction or energy of neutrons that are produced from fission. We scale fission production by the distribution $\chi(\hat{\Omega},E)$ to obtain

$$\left(\int_V d^3\vec{r}\int_{4\pi} d\hat{\Omega}'\int_0^\infty dE'\,\chi(\hat{\Omega},E)\,v(E')\,\nu(\vec{r},E',t)\,\Sigma_f(\vec{r},E',t)\,n(\vec{r},\hat{\Omega},E,t)\right)\,d\hat{\Omega}\,dE. \tag{2.14}$$

Collisions that give no preference to any one direction $\hat{\Omega}$ are isotropically distributed. We integrate $\chi(\hat{\Omega},E)$ with respect to angle and obtain the coefficient $1/4\pi$, a normalizing constant from the solid angle integral for isotropic χ. In contrast, the energies of produced neutrons are not uniformly distributed. Therefore, the created neutrons are weighted by $\chi(E)$. The total contribution from

fission is

$$\left(\int_V d^3\vec{r}\,\frac{\chi(E)}{4\pi}\int_{4\pi}d\hat{\Omega}'\int_0^\infty dE'\;v(E')\,\nu(\vec{r},E',t)\,\Sigma_f(\vec{r},E',t)\,n(\vec{r},\hat{\Omega},E,t)\right)d\hat{\Omega}\,dE. \tag{2.15}$$

6. **Neutrons undergoing collision in volume V:**
   The total rate of collisions at location $\vec{r}$ is

$$f_t(\vec{r},\hat{\Omega},E,t) = v(E)\,\Sigma_t(\vec{r},E)\,n(\vec{r},\hat{\Omega},E,t). \tag{2.16}$$

   Integrate over volume and multiply by the differentials to find the loss term,

$$\left(\int_V v(E)\,\Sigma_t(\vec{r},E)\,n(\vec{r},\hat{\Omega},E,t)\,d^3\vec{r}\right)d\hat{\Omega}\,dE. \tag{2.17}$$

We have defined all production and loss terms. The resulting neutron balance is

$$\begin{aligned}\int_V d^3\vec{r}\,\Bigg[&\frac{\partial n}{\partial t}+v(E)\,\hat{\Omega}\cdot\nabla n+v(E)\,\Sigma_t\,n(\vec{r},\hat{\Omega},E,t)-\\&\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'v(E')\,\Sigma_s(\hat{\Omega}'\to\hat{\Omega},E'\to E)\,n(\vec{r},\hat{\Omega}',E',t)-\\&\frac{\chi(\vec{r},E,t)}{4\pi}\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\,v(E')\,\nu(\vec{r},E',t)\,\Sigma_f(\vec{r},E',t')\,n(\vec{r},\hat{\Omega}',E',t)+Q(\vec{r},\hat{\Omega},E,t)\Bigg]\,dE\,d\hat{\Omega}=0.\end{aligned} \tag{2.18}$$

V was chosen arbitrarily. An integral equivalent to zero for any V implies that the integrand is equivalently zero. Concluding, we obtain the final equation

$$\begin{aligned}\frac{\partial n}{\partial t}+v(E)\,\hat{\Omega}\cdot\nabla n+v(E)\,\Sigma_t\,n(\vec{r},\hat{\Omega},E,t)=&\\\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'v(E')\,\Sigma_s(\hat{\Omega}'\to\hat{\Omega},E'\to E)\,n(\vec{r},\hat{\Omega}',E',t)+&\\\frac{\chi(\vec{r},E,t)}{4\pi}\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\,v(E')\,\nu(\vec{r},E',t)\,\Sigma_f(\vec{r},E',t')\,n(\vec{r},\hat{\Omega}',E',t)+Q(\vec{r},\hat{\Omega},E,t)&\end{aligned} \tag{2.19}$$

where $\psi(\vec{r},\hat{\Omega},E,t) = v(E)\,n(\vec{r},\hat{\Omega},E,t)$ to obtain (2.43), the 3-D neutron transport equation.

2.2.2 Boundary and Initial Conditions

We introduce three different boundary conditions and an initial condition that appear in regular applications [37].

1. **Vacuum Boundary:**
   Vacuum boundaries do not allow neutrons to enter the system from the boundaries. Neutrons once outside the system do no return. This description is only for incoming directions $\hat{\Omega}$, so in a one dimensional problem, we have two half-conditions on the left and right boundary. We write angles for incoming directions in terms of the outward normal $\hat{n}$ as $\hat{n} \cdot \hat{\Omega} < 0$. The vacuum boundary condition is
   $$\psi(\vec{r}_b, \hat{\Omega}, E, t) = 0 \tag{2.20}$$
   with $\hat{n} \cdot \hat{\Omega} < 0$.

2. **Reflective Boundary:**
   Reflective boundaries do not allow neutrons to leave the system and instead neutrons are reflected when hitting a boundary. New particle direction is dependent on the incidental angle. If the new direction is $\hat{\Omega}'$, then we write
   $$\hat{n} \cdot \hat{\Omega} = -\hat{n} \cdot \hat{\Omega}'; \quad (\hat{\Omega} \times \hat{\Omega}') \cdot \hat{n} = 0 \tag{2.21}$$
   with half-angles $\hat{n} \cdot \hat{\Omega} < 0$. The reflective boundary condition is
   $$\psi(\vec{r}_b, \hat{\Omega}, E, t) = \psi(\vec{r}_b, \hat{\Omega}', E, t). \tag{2.22}$$
   Reflective boundary conditions are common in shielding applications.

3. **Fixed Source Boundary:**
   Fixed source boundary conditions explicitly set the angular flux at the boundary by a known function over the phase space. The boundary position and incoming angles of the source function are fixed, but the function may vary over energy. The fixed source boundary condition is
   $$\psi(\vec{r}_b, \hat{\Omega}, E, t) = \bar{\psi}(\vec{r}_b, \hat{\Omega}, E, t) \tag{2.23}$$
   with $\hat{n} \cdot \hat{\Omega} < 0$.

4. **Initial Condition:**
   The initial condition is the initial configuration of neutrons over space, angle, and energy at time $t = 0$. The initial condition is written as
   $$\psi(\vec{r}, \hat{\Omega}, E, t)\Big|_{t=0} = \tilde{\psi}(\vec{r}, \hat{\Omega}, E). \tag{2.24}$$

We have fully defined and derived the transport equation, its boundary conditions, and its initial conditions. The next step is to discretize the transport equation. The discretized space allows for numerical

computations when continuous solutions are unavailable.

2.3 Angular and Energy Discretization of the Transport Equation

First, we develop discretizations for the angle and energy spaces. Applications we consider are steady state problems; therefore, we need no time discretization. The spatial domain is discretized as needed depending on the application or theory being presented. The time-independent problem is

$$\begin{aligned}\hat{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\hat{\Omega},E)+\Sigma_t(\vec{r},E)\,\psi(\vec{r},\hat{\Omega},E)=&\\ \int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\;\Sigma_s(\vec{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)\,\psi(\vec{r},\hat{\Omega}',E')+&\\ \frac{\chi(\vec{r},E)}{4\pi}\int_0^\infty dE'\int_{4\pi}d\hat{\Omega}'\;\nu(\vec{r},E')\,\Sigma_f(\vec{r},E')\,\psi(\vec{r},\hat{\Omega}',E')+Q(\vec{r},\hat{\Omega},E),&\end{aligned}\tag{2.25}$$

with boundary conditions

$$\psi(\vec{r}_b,\hat{\Omega},E)=\psi_b(\vec{r}_b,\hat{\Omega},E),\;\vec{r}_b\in\partial A;\;\hat{\Omega}\cdot\hat{e}_n<0.\tag{2.26}$$

2.3.1 Angular Discretization

A basic angular discretization is the S_N method, or *discrete ordinates* method [4, 12, 37]. The angular discretization is a finite set $\hat{\Omega}_n=(\eta_n,\xi_n,\mu_n)^T$, each of unit length. Functions of $\hat{\Omega}$ are defined only on node locations:

$$f(\hat{\Omega})\to f(\hat{\Omega}_n)\equiv f_n,\quad n=1,\ldots,N.\tag{2.27}$$

We integrate over $\hat{\Omega}$ with quadrature weights w_n for each of the angular nodes, as

$$\int_{4\pi}f(\hat{\Omega}')\,d\hat{\Omega}'=\sum_{n=1}^{N}w_nf_n.\tag{2.28}$$

Using $(w_n,\hat{\Omega}_n)$, we can form the S_N equations for the n^{th} direction from (2.25) as

$$\begin{aligned}\hat{\Omega}_n\cdot\vec{\nabla}\psi_n(\vec{r},E)+\Sigma_t(\vec{r},E)\,\psi_n(\vec{r},E)=&\\ \int_0^\infty dE'\sum_{n'=1}^{N}\;\Sigma_{s,n'\to n}(\vec{r},E'\to E)\,w_{n'}\psi_{n'}(\vec{r},E')+&\\ \frac{\chi(\vec{r},E)}{4\pi}\int_0^\infty dE'\sum_{n'=1}^{N}\;\nu(\vec{r},E')\,\Sigma_f(\vec{r},E')\,w_{n'}\psi_{n'}(\vec{r},E')+Q_n(\vec{r},E),&\end{aligned}\tag{2.29}$$

where $\psi_n(\vec{r},E)=\psi(\vec{r},\Omega_n,E)$ and $\Sigma_{s,n'\to n}(\vec{r},E'\to E)=\Sigma_s(\vec{r},\Omega_{n'}\to\Omega_n,E'\to E)$.

We still must develop the energy mesh to have a fully discretized problem.

2.3.2 Energy Discretization

Each particle has energy that lies on the interval $[0, \infty)$. One approach to energy discretization is called the *multigroup* approximation. The multigroup approximation divides the interval into G continuous non-uniform intervals [4, 12, 37]. We write the intervals as $[E_G, E_{G-1}]$, $[E_{G-1}, E_{G-2}]$, $\ldots$, $[E_g, E_{g-1}]$, $\ldots$, $[E_2, E_1]$, $[E_1, E_0]$, where the lower value of g indicates a higher energy value and $E_G = 0$.

The first approximation is choosing a maximum energy value $E_0 < \infty$ such that all flux contributions from particles above that energy are reasonably ignored. Choosing an E_0 is application dependent and must be chosen deliberately. We divide $[0, E_0]$ so that neutrons live in distinct energy groups. A neutron is in group g if the energy of that neutron lies between $[E_g, E_{g-1}]$.

The energy grid provides new degrees of freedom where we solve the differential equation. The solutions to the multigroup equations are group angular fluxes, ψ_g. The group angular flux is

$$\psi_g(\vec{r}, \hat{\Omega}) = \int_{E_g}^{E_{g-1}} \psi(\vec{r}, \hat{\Omega}, E)\, dE \equiv \int_g \psi(\vec{r}, \hat{\Omega}, E)\, dE. \tag{2.30}$$

In the neutron transport equation (2.43), we must define a full integral over $[0, \infty)$ with the new finite energy grid. The intervals are all distinct so the total integral is equal to the summation of each distinct element; thus, the integral is

$$\int_0^\infty \cdot\, dE' = \sum_{g'=1}^{G} \int_{g'} \cdot\, dE'. \tag{2.31}$$

Using the multigroup integral approximation to replace the full energy integrals in (2.25) and integrating the whole equation over a single g, we obtain

$$\begin{aligned}\hat{\Omega} \cdot \vec{\nabla} \int_g dE\, \psi(\vec{r}, \hat{\Omega}, E) + \int_g dE\, \Sigma_t(\vec{r}, E)\, \psi(\vec{r}, \hat{\Omega}, E) = \\ \sum_{g'=1}^{G} \int_g dE \int_{g'} dE' \int_{4\pi} d\hat{\Omega}'\ \Sigma_s(\vec{r}, \hat{\Omega}' \to \hat{\Omega}, E' \to E)\, \psi(\vec{r}, \hat{\Omega}', E') + \\ \int_g dE\, \frac{\chi(\vec{r}, E)}{4\pi} \sum_{g'=1}^{G} \int_{g'} dE' \int_{4\pi} d\hat{\Omega}'\ \nu(\vec{r}, E')\, \Sigma_f(\vec{r}, E')\, \psi(\vec{r}, \hat{\Omega}', E') + \int_g dE\, Q(\vec{r}, \hat{\Omega}, E).\end{aligned} \tag{2.32}$$

We assume that the angular flux is separable in energy to use ψ_g. We require

$$\psi(\vec{r}, \hat{\Omega}, E) = \psi_g(\vec{r}, \hat{\Omega}) f(E), \quad E \in [E_g, E_{g+1}] \tag{2.33}$$

where $f(E)$ is some distribution and

$$\int_g dE\ f(E) = 1, \tag{2.34}$$

which is true for all group intervals. Using (2.33), we arrive at

$$\begin{aligned}\hat{\Omega}\cdot\vec{\nabla}\int_g dE\ f(E)\,\psi_g(\vec{r},\hat{\Omega}) + \int_g dE\ f(E)\,\Sigma_t(\vec{r},E)\,\psi_g(\vec{r},\hat{\Omega}) = \\ \sum_{g'=1}^{G}\int_g dE\int_{g'} dE'\int_{4\pi} d\hat{\Omega}'\ \Sigma_s(\vec{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)\,f(E')\,\psi_g(\vec{r},\hat{\Omega}') + \\ \int_g dE\,\frac{\chi(\vec{r},E)}{4\pi}\sum_{g'=1}^{G}\int_{g'} dE'\int_{4\pi} d\hat{\Omega}'\ \nu(\vec{r},E')\,\Sigma_f(\vec{r},E')\,f(E')\,\psi_g(\vec{r},\hat{\Omega}') + \int_g dE\ Q(\vec{r},\hat{\Omega},E).\end{aligned} \tag{2.35}$$

The coefficients are still continuous in energy E and not in terms of the group structure we formulated. So, the new multi-group cross sections, fission spectrum, and external source are

$$\Sigma_{t,g}(\vec{r}) = \int_g f(E)\,\Sigma_t(\vec{r},E)\,dE, \tag{2.36}$$

$$\nu_g(\vec{r})\Sigma_{f,g}(\vec{r}) = \int_g f(E')\,\nu(\vec{r},E')\,\Sigma_f(\vec{r},E')\,dE', \tag{2.37}$$

$$\Sigma_{s,g'\to g}(\vec{r},\hat{\Omega}'\to\hat{\Omega}) = \int_{g'}\int_g f(E')\,\Sigma_s(\vec{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)\,dE'\,dE, \tag{2.38}$$

$$\chi_g(\vec{r}) = \int_g \chi(\vec{r},E)\,dE, \tag{2.39}$$

$$Q_g(\vec{r},\hat{\Omega}) = \int_g Q_g(\vec{r},\hat{\Omega},E)\,dE. \tag{2.40}$$

Finally, the transport equation in multi-group form for a single group g is

$$\begin{aligned}\hat{\Omega}\cdot\vec{\nabla}\psi_g(\vec{r},\hat{\Omega}) + \Sigma_{t,g}(\vec{r})\,\psi_g(\vec{r},\hat{\Omega}) = \\ \sum_{g'=1}^{G}\int_{4\pi} d\hat{\Omega}'\ \Sigma_{s,g'\to g}(\vec{r},\hat{\Omega}'\to\hat{\Omega})\,\psi_g(\vec{r},\hat{\Omega}') + \\ \frac{\chi_g(\vec{r})}{4\pi}\sum_{g'=1}^{G}\int_{4\pi} d\hat{\Omega}'\ \nu_g(\vec{r})\,\Sigma_{f,g}(\vec{r})\,\psi_g(\vec{r},\hat{\Omega}') + Q_g(\vec{r},\hat{\Omega}).\end{aligned} \tag{2.41}$$

We combine the two discretizations, (2.29) and (2.41), to obtain

$$\hat{\Omega}_n \cdot \vec{\nabla}\psi_{n,g}(\vec{r}) + \Sigma_{t,g}(\vec{r})\,\psi_{n,g}(\vec{r}) = \sum_{g'=1}^{G}\sum_{n'=1}^{N} \Sigma_{s,n'\to n,g'\to g}(\vec{r})\, w_{n'}\psi_{n',g'}(\vec{r}) + \frac{\chi_g(\vec{r})}{4\pi}\sum_{g'=1}^{G}\sum_{n'=1}^{N} \nu_{g'}(\vec{r})\,\Sigma_{f,g'}(\vec{r})\, w_{n'}\psi_{n',g'}(\vec{r}) + Q_{n,g}(\vec{r}). \quad (2.42)$$

We discuss how to solve the full system of multi-group equations with up and down-scattering from Definition 1.2.2, 1.2.3 in Chapter 3.

2.4 The k-eigenvalue Problem

In nuclear reactor analysis, an important factor to determine is the criticality of the reactor. The criticality is defined as the multiplicative rate at which neutrons are being produced through fission in the system. The dominant eigenvalue of this system is k_{eff}.

1. *Subcritical ($k_{eff} < 1$)*: The total number of neutrons in the system is decreasing either from leakage or absorption.
2. *Critical ($k_{eff} = 1$)*: The total number of neutrons in the system remains constant.
3. *Supercritical ($k_{eff} > 1$)*: The total number of neutrons in the system is increasing either from fission or fixed sources.

The k-eigenvalue problem is

$$\hat{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\hat{\Omega},E) + \Sigma_t(\vec{r},E)\,\psi(\vec{r},\hat{\Omega},E) = \int_0^\infty dE' \int_{4\pi} d\hat{\Omega}'\ \Sigma_s(\vec{r},\hat{\Omega}'\to\hat{\Omega},E'\to E)\,\psi(\vec{r},\hat{\Omega}',E') + \frac{\chi(\vec{r},E)}{4\pi k_{\text{eff}}}\int_0^\infty dE' \int_{4\pi} d\hat{\Omega}'\ \nu(\vec{r},E')\,\Sigma_f(\vec{r},E')\,\psi(\vec{r},\hat{\Omega}',E'), \quad (2.43)$$

with boundary conditions

$$\psi(\vec{r}_b,\hat{\Omega},E) = \psi_b(\vec{r}_b,\hat{\Omega},E),\ \vec{r}_b\in\partial A;\ \hat{\Omega}\cdot\hat{e}_n < 0. \quad (2.44)$$

The solution to (2.43) is both ψ and k_{eff}, the eigenfunction and eigenvalue. We discuss solution techniques to the k-eigenvalue problem in multi-group form in Chapter 3.

2.5 One-Speed Transport Equation in Slab Geometry

To demonstrate basic methods, it is convenient to work with a simplified form of the transport equation. Many of the algorithms and solution techniques may be extended to the full form. A common simplification of the full transport equation is the one-speed steady-state neutron transport equation in slab geometry for one dimension. Further simplifications include *isotropic* scattering and source, meaning there is no preference given to different angles $\hat{\Omega}$.

2.5.1 Slab Geometry

Slab geometry describes a material slab that is infinite in both the y and z dimensions, where solution from any horizontal cross-section at height z would be identical to any other cross-section at a different height z'. We define the new angular variable $\mu = \cos\theta$, seen in Figure 2.1, where θ describes the polar angle out of the cross-sectional plane; thus, $\mu \in [-1, 1]$.

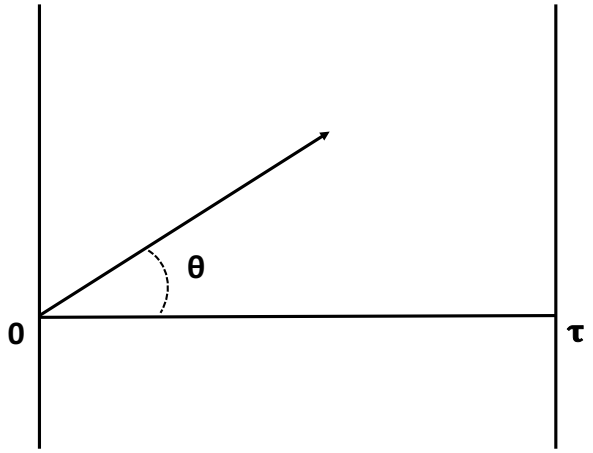


Figure 2.1: Angular variable $\mu = \cos(\theta)$ in slab geometry.

We present the one-speed steady-state transport equation in 1D slab geometry with isotropic scattering and an isotropic source

$$\mu\frac{\partial\psi(x,\mu)}{\partial x} + \Sigma_t(x)\psi(x,\mu) = \frac{1}{2}\left(\Sigma_s(x) + \nu_f(x)\Sigma_f(x)\right)\int_{-1}^{1} d\mu'\psi(x,\mu') + \frac{1}{2}q(x). \tag{2.45}$$

for

$$0 \le x \le X, \qquad -1 \le \mu \le 1 \tag{2.46}$$

with the boundary conditions

$$\psi|_{x=0} = \psi_0^{in}(\mu), \text{for } \mu > 0, \tag{2.47}$$

$$\psi|_{x=X} = \psi_X^{in}(\mu), \text{for } \mu < 0. \tag{2.48}$$

We can also define the scalar flux similarly as

$$\phi(x) = \int_{-1}^{1} d\mu' \; \psi(x, \mu'). \tag{2.49}$$

Solutions and techniques for the transport equation in slab geometry are explored in the following chapter.

Chapter 3

Deterministic Solutions to the Neutron Transport Equation

Deterministic approaches to solving the neutron transport equation include direct methods and iterative methods [4, 37]. Direct methods apply a discretization mesh to the phase space and solve the partial differential equation in one calculation. Iterative methods in contrast find a sequence of increasingly accurate iterates through a fixed point map that is both consistent and stable. Both types of methods have been applied to the neutron transport equation but the majority of our focus is on *discrete ordinates*, iterative methods, and acceleration schemes. Specifically, we discuss solutions to the single material one-speed fixed-source problem

$$\mu \frac{\partial \psi(x,\mu)}{\partial x} + \Sigma_t \psi(x,\mu) = \frac{1}{2}\left[\Sigma_s \phi(x) + q(x)\right], \tag{3.1}$$

for

$$0 \leq x \leq X, \quad -1 \leq \mu \leq 1 \tag{3.2}$$

with the boundary conditions

$$\psi|_{x=0} = \psi_0^{in}(\mu), \quad \text{for } \mu > 0, \tag{3.3}$$

$$\psi|_{x=X} = \psi_X^{in}(\mu), \quad \text{for } \mu < 0. \tag{3.4}$$

3.1 Source Iteration

One solution technique for the neutron transport equation is source iteration. We create a stable and consistent contraction mapping, $\mathcal{S}(\phi)$, for the scalar flux that converges to the "true" scalar flux solution through Picard iteration. We denote the true solution as ϕ^*. If ϕ^* is known then (3.1) is solvable for the angular flux by a *transport sweep*, which is discussed in sections further below. An overview of the

Picard iteration is Algorithm 1.

Algorithm 1 Source Iteration

`source_iteration`$(\phi^{(0)}, \tau)$

Begin with initial iterate $\phi^{(0)}$ and tolerance τ.
$\phi^{(c)} = \mathcal{S}(\phi^{(0)})$.
Calculate initial residual $r = \|\phi^{(0)} - \phi^{(c)}\|$.
while $r > \tau$ **do**
  Evaluate $\phi^{(+)} = \mathcal{S}(\phi^{(c)})$.
  Calculate $r = \|\phi^{(c)} - \phi^{(+)}\|$.
  Update $\phi^{(c)} = \phi^{(+)}$.
end while

3.2 Source Iteration Map for the Continuous Problem

We begin by formulating the source iteration mapping for the continuous problem, following the steps found in [8, 9, 30]. We show that the transport equation is used to derive the integral form

$$\phi(y) - \int_0^\tau k(x-y)\,\phi(x)\,dx = g(y). \tag{3.5}$$

We manipulate the transport equation by a change of variables, with the spatial variable $x = \Sigma_t z$, assuming Σ_t constant. The resulting equation is

$$\mu\Sigma_t \frac{\partial\psi(z,\mu)}{\partial z} + \Sigma_t\psi(z,\mu) = \frac{1}{2}\left[\Sigma_s\phi(z) + q(z)\right]. \tag{3.6}$$

We define a new constant $c = \Sigma_s/\Sigma_s$, called the scattering ratio, and the new source $\tilde{q}(x) = q(x)/\Sigma_t$. Dividing the altered transport equation by Σ_t, we obtain

$$\mu\frac{\partial\psi(z,\mu)}{\partial z} + \psi(z,\mu) = \frac{1}{2}\left[c\phi(z) + \tilde{q}(z)\right]. \tag{3.7}$$

The change of variables morphs the spatial domain from $x \in [0, X]$ to $z \in [0, \tau]$ where $\tau = X/\Sigma_t$. We continue with $\mu > 0$:

$$\begin{aligned}
\mu \frac{\partial \psi}{\partial z} + \psi(z,\mu) &= \frac{1}{2}\left[c\phi(z) + \tilde{q}(z)\right], \\
e^{z/\mu}\left[\mu \frac{\partial \psi}{\partial z} + \psi(z,\mu)\right] &= \frac{1}{2} e^{z/\mu}\left[c\phi(z) + \tilde{q}(z)\right], \\
\mu \frac{\partial}{\partial z}\left[e^{z/\mu}\psi(z,\mu)\right] &= \frac{1}{2} e^{z/\mu}\left[c\phi(z) + \tilde{q}(z)\right], \\
\frac{\partial}{\partial z}\left[e^{z/\mu}\psi(z,\mu)\right] &= \frac{1}{2\mu} e^{z/\mu}\left[c\phi(z) + \tilde{q}(z)\right], \\
\int_0^y \frac{\partial}{\partial z}\left[e^{z/\mu}\psi(z,\mu)\right] dz &= \int_0^y \frac{1}{2\mu} e^{z/\mu}\left[c\phi(z) + \tilde{q}(z)\right] dz, \\
e^{y/\mu}\psi(y,\mu) - \psi(0,\mu) &= \int_0^y \frac{1}{2\mu} e^{z/\mu}\left[c\phi(z) + \tilde{q}(z)\right] dz, \\
\psi(y,\mu) &= e^{-y/\mu}\int_0^y \frac{1}{2\mu} e^{z/\mu}\left[c\phi(z) + \tilde{q}(z)\right] dz + e^{-y/\mu}\psi_0^{in}(\mu), \\
\psi(y,\mu) &= \int_0^y \frac{1}{2\mu} e^{(z-y)/\mu}\left[c\phi(z) + \tilde{q}(z)\right] dz + e^{-y/\mu}\psi_0^{in}(\mu), \\
\psi(y,\mu) &= \int_0^y \frac{1}{2\mu} e^{-|z-y|/|\mu|}\left[c\phi(z) + \tilde{q}(z)\right] dz + e^{-y/|\mu|}\psi_0^{in}(\mu).
\end{aligned}$$

The analogous steps are completed for $\mu < 0$ to obtain

$$\psi(y,\mu) = \int_y^\tau \frac{1}{2|\mu|} e^{-|z-y|/|\mu|}\left[c\phi(z) + \tilde{q}(z)\right] dz + e^{-|\tau-y|/|\mu|}\psi_\tau^{in}(\mu), \tag{3.8}$$

where $\psi(z,\mu)|_{z=\tau} = \psi_\tau^{in}(\mu) = \psi_X^{in}(\mu)$. Thus, we have formulas for the angular flux for both positive and negative angles μ. Recognize that the formula for the scalar flux is split as

$$\phi(x) = \int_{-1}^1 \psi(x,\mu)\, d\mu = \int_{-1}^0 \psi(x,\mu)\, d\mu + \int_0^1 \psi(x,\mu)\, d\mu, \tag{3.9}$$

which directly leads to

$$\begin{aligned}
\phi(y) = &\int_{-1}^0 \int_y^\tau \frac{1}{2|\mu|} e^{-|z-y|/|\mu|}\left[c\phi(z) + \tilde{q}(z)\right] dz\, d\mu + \int_{-1}^0 e^{-|\tau-y|/|\mu|}\psi_\tau^{in}(\mu)\, d\mu \\
&+ \int_0^1 \int_0^y \frac{1}{2\mu} e^{-|z-y|/|\mu|}\left[c\phi(z) + \tilde{q}(z)\right] dz\, d\mu + \int_0^1 e^{-y/|\mu|}\psi_0^{in}(\mu)\, d\mu.
\end{aligned} \tag{3.10}$$

We rewrite the scalar flux by splitting the double integrals as

$$\phi(y) = \int_{-1}^{0}\int_{y}^{\tau} \frac{c}{2|\mu|} e^{-|z-y|/|\mu|}\phi(z)\, dz\, d\mu + \int_{-1}^{0}\int_{y}^{\tau} \frac{1}{2|\mu|} e^{-|z-y|/|\mu|}\tilde{q}(z)\, dz\, d\mu + \int_{-1}^{0} e^{-|\tau-y|/|\mu|}\psi_{\tau}^{in}(\mu)\, d\mu$$

$$+ \int_{0}^{1}\int_{0}^{y} \frac{c}{2\mu} e^{-|z-y|/|\mu|}\phi(z)\, dz\, \mu + \int_{0}^{1}\int_{0}^{y} \frac{1}{2\mu} e^{-|z-y|/|\mu|}\tilde{q}(z) dz\, d\mu + \int_{0}^{1} e^{-y/|\mu|}\psi_{0}^{in}(\mu)\, d\mu \tag{3.11}$$

and combining the spatial integrals and changing the order of integration, we obtain

$$\phi(y) = \int_{0}^{\tau}\int_{0}^{1} \frac{c}{2\mu} e^{-|z-y|/\mu}\phi(z)\, d\mu\, dz + \int_{0}^{\tau}\int_{0}^{1} \frac{1}{2\mu} e^{-|z-y|/\mu}\tilde{q}(z)\, d\mu\, dz$$

$$+ \int_{-1}^{0} e^{-|\tau-y|/|\mu|}\psi_{\tau}^{in}(\mu)\, d\mu + \int_{0}^{1} e^{-y/|\mu|}\psi_{0}^{in}(\mu)\, d\mu. \tag{3.12}$$

The integral equation is then

$$\phi(y) - \int_{0}^{\tau} k(y-z)\, \phi(z)\, dz = g(y), \tag{3.13}$$

where the kernel

$$k(y-z) = \int_{0}^{1} \frac{c}{2\mu} e^{-|z-y|/\mu}\, d\mu \tag{3.14}$$

and

$$g(y) = \int_{0}^{\tau}\int_{0}^{1} \frac{1}{2\mu} e^{-|z-y|/\mu}\tilde{q}(z)\, d\mu\, dz + \int_{-1}^{0} e^{-|\tau-y|/|\mu|}\psi_{\tau}^{in}(\mu)\, d\mu + \int_{0}^{1} e^{-y/|\mu|}\psi_{0}^{in}(\mu)\, d\mu. \tag{3.15}$$

Continuing as in [30], the continuous form of source iteration is

$$\phi^{(+)} = \mathbf{K}(\phi^{(c)}) + g, \tag{3.16}$$

with the operator $\mathbf{K}$ defined as

$$\mathbf{K}(\phi)(x) = \int_{0}^{\tau} k(x-z)\, \phi(z)\, dz. \tag{3.17}$$

3.2.1 Convergence of Source Iteration

It is important to note that the operator $\mathbf{I} - \mathbf{K}$ is not invertible when $\tau = \infty$ and $c = 1$. Source iteration is slow to converge in cases where τ is large and c near unity from [9, 34]. In Appendix A.1, we prove that stationary iterative methods converge to a fixed point if $\|\mathbf{K}\| < 1$. Further information on induced matrix norms is located there, as well. We prove that the operator is indeed norm-bounded by one. We use the ℓ_1-norm,

$$\|f\|_1 = \int_{0}^{\tau} |f(x)|\, dx. \tag{3.18}$$

The definition of $\mathbf{K}$ acting on the scalar flux ϕ is

$$\begin{aligned}
\|\mathbf{K}(\phi)\|_1 &= \int_0^\tau |\mathbf{K}(\phi)(x)|\, dx, \\
&= \int_0^\tau \left| \int_0^\tau k(x,z)\phi(z)\, dz \right| dx, \\
&\leq \int_0^\tau \int_0^\tau |k(x,z)||\phi(z)|\, dz\, dx, \\
&= \int_0^\tau \int_0^\tau |k(x,z)||\phi(z)|\, dx\, dz, \\
&= \frac{c}{2} \int_0^\tau |\phi(z)| \left[\int_0^\tau \left| \int_0^1 \frac{1}{\mu} e^{-|z-x|/\mu}\, d\mu \right| dx \right] dz, \\
&= \frac{c}{2} \int_0^\tau |\phi(z)| \left[\int_0^\tau \int_0^1 \frac{1}{\mu} e^{-|z-x|/\mu}\, d\mu\, dx \right] dz, \\
&\leq \frac{c}{2} \int_0^\tau |\phi(z)| \left[\int_{-\infty}^\infty \int_0^1 \frac{1}{\mu} e^{-|z-x|/\mu}\, d\mu\, dx \right] dz, \\
&= \frac{c}{2} \int_0^\tau |\phi(z)| \left[\int_{-\infty}^\infty \int_0^1 \frac{1}{\mu} e^{-|y|/\mu}\, d\mu\, dy \right] dz, \\
&= \frac{c}{2} \int_0^\tau |\phi(z)|\, dz \int_{-\infty}^\infty \int_0^1 \frac{1}{\mu} e^{-|y|/\mu}\, d\mu\, dy, \\
&= \frac{c}{2} \|\phi\|_1 \int_{-\infty}^\infty \int_0^1 \frac{1}{\mu} e^{-|y|/\mu}\, d\mu\, dy, \\
&= \frac{c}{2} \|\phi\|_1 \int_0^1 \frac{1}{\mu} \int_{-\infty}^\infty e^{-|y|/\mu}\, dy\, d\mu, \\
&= \frac{c}{2} \|\phi\|_1 \int_0^1 \frac{2}{\mu} \int_0^\infty e^{-y/\mu}\, dy\, d\mu, \\
&= \frac{c}{2} \|\phi\|_1 \int_0^1 \frac{2}{\mu} \cdot \mu\, d\mu, \\
&= c\|\phi\|_1.
\end{aligned}$$

The resulting relationship

$$\|\mathbf{K}\phi\| \leq c\|\phi\|$$

for any element ϕ, describes a bound for the norm of the operator $\mathbf{K}$, which is $\|\mathbf{K}\| < c$. Thus, as stated previously, if $c < 1$ this implies that $\|\mathbf{K}\| < 1$ and so source iteration converges.

3.3 Transport Sweep and the Discrete Contraction Mapping

We discretize (3.1) on spatial nodes x_i from $i = 0, 1, \ldots, N$ and a standard S_N angular discretization for angle μ_j from $j = 1, 2, \ldots M$, similar to the description in Chapter 2. We have a uniform spatial grid, so

cell width $h = X/(N+1)$. We choose a Gaussian quadrature for angle (μ_j, w_j) as in [28], so that

$$\sum_{j=1}^{M} w_j \, \psi(x, \mu_j) = \int_{-1}^{1} \psi(x, \mu') \, d\mu'. \tag{3.19}$$

The angular flux, scalar flux, and external source are defined at grid nodes, as follows:

$$\psi_i^j = \psi(x_i, \mu_j); \quad \phi_i = \phi(x_i); \quad q_i = q(x_i). \tag{3.20}$$

We use a second order central difference approximation at cell faces $x_{i+1/2}$ for the streaming term and central averaging for all other terms. We obtain the fully discrete problem

$$\mu^j \frac{\psi_{i+1}^j - \psi_i^j}{h} + \Sigma_t \frac{\psi_{i+1}^j + \psi_i^j}{2} = \frac{S_{i+1} + S_i}{2}, \tag{3.21}$$

where the total source $S_i = \Sigma_s \phi_i + q_i$, and we have boundary conditions

$$\psi_0^j = \psi_0^{in}(\mu_j), \quad \text{for } \mu_j > 0, \tag{3.22}$$

$$\psi_N^j = \psi_N^{in}(\mu_j), \quad \text{for } \mu_j < 0. \tag{3.23}$$

Solving (3.21) for ψ_{i+1}^j, we have

$$\psi_{i+1}^j = \left(\frac{1}{h} + \frac{\Sigma_t}{2}\right)^{-1} \left[\frac{S_{i+1} + S_i}{2} + \left(\frac{1}{h} - \frac{\Sigma_t}{2}\right)\psi_i^j\right]. \tag{3.24}$$

Given the boundary condition ψ_0^j, we forward march for $\mu_j > 0$ using the explicit formula for the next spatial grid point. This is called a *forward sweep*. Similarly, we form a reverse marching formula for ψ_{i-1} using a backwards central difference formula for $x_{i-1/2}$. This produces the explicit formula

$$\psi_{i-1}^j = \left(\frac{\Sigma_t}{2} - \frac{1}{h}\right)^{-1} \left[\frac{S_i + S_{i-1}}{2} - \left(\frac{1}{h} + \frac{\Sigma_t}{2}\right)\psi_i^j\right] \tag{3.25}$$

for $\mu_j < 0$, the reverse direction. This is called a *backward sweep*. Together a forward and backward sweep create a *transport sweep*, a unit of work in transport theory similar to a matrix-vector product or a floating point operation. The transport sweep calculates the angular flux at all grid points from a scalar flux.

For the contraction mapping $\mathcal{S}(\phi)$, the transport sweep is the first component. The second component is a closure relation that projects the angular flux onto the lower dimensional space where the scalar flux lives. In source iteration, the closure relation is a simple integral over μ. We use the Gaussian quadrature

to have

$$\phi_i^{(+)} = \sum_{j=1}^{M} w_j \, \psi_i^j, \tag{3.26}$$

where $\phi_i^{(+)} = \mathcal{S}(\phi^{(c)})_i$ is the output of the contraction mapping at the grid point x_i. Finally, the last element required to complete Algorithm 1 is to choose a initial iterate $\phi^{(0)}$ to begin the Picard iteration and an absolute stopping tolerance τ_a.

Many of the deterministic iterative methods in neutron transport use this algorithm as a framework. Various methods are formed by choosing a different closure relation in the second component of the contraction mapping [32, 37]. These alternate formulations are chosen to decrease the total number of transport sweeps in the calculation, and thereby the total computational work. We are also able to solve (3.16) by forming an invertible linear system. We never explicitly create the matrix of this linear system, but instead we describe an operator acting on a vector in a *matrix-free* way.

3.4 Solving the Transport Equation as a Linear System

Solution strategies for linear systems are well documented throughout the mathematics literature [27, 28]. We form a linear system from the integral equation we have derived. The source iteration map is

$$\phi^{(+)} = \mathbf{K}(\phi^{(c)}) + g. \tag{3.27}$$

If $\mathbf{K}$ is a linear operator, then the linear system of equations is

$$\begin{aligned} \phi - \mathbf{K}\phi &= g, \\ (\mathbf{I} - \mathbf{K})\phi &= g, \\ \mathbf{A}\phi &= g, \end{aligned} \tag{3.28}$$

where $\mathbf{A} = \mathbf{I} - \mathbf{K}$.

Source iteration is slow to converge for high scattering ratios and may take more transport sweeps than one might want to invest. A faster method of finding of a solution to this system is using Krylov methods, such as GMRES. See Appendix A.2.

3.4.1 Source Iteration Components for the Linear Solver

We use $\mathcal{S}$ to formulate the linear operator and right-hand side. We call the discretized fixed point map, $\mathbf{S}$. We seek a solution to the fix point problems

$$\phi = \mathbf{S}(\phi), \tag{3.29}$$

$$\phi = \mathbf{K}\phi + g; \tag{3.30}$$

therefore, we have $\mathbf{S}(\phi) = \mathbf{K}\phi + g$. To obtain g, we use

$$\mathbf{S}(\vec{0}) = \mathbf{K}\vec{0} + g = g, \tag{3.31}$$

because of the linearity of $\mathbf{K}$. From (3.28), the matvec is

$$\begin{aligned}\mathbf{A}\phi &= (\mathbf{I} - \mathbf{K})\phi, \\ &= \phi - \mathbf{K}\phi, \\ &= \phi - (\mathbf{S}(\phi) - g), \\ &= \phi - \mathbf{S}(\phi) + \mathbf{S}(\vec{0}).\end{aligned}$$

Thus, we solve the system

$$\mathbf{A}\phi = g \tag{3.32}$$

for the solution, ϕ. We may accelerate the convergence of source iteration with a modified fixed point iteration. Two examples are Anderson acceleration and nonlinear diffusion acceleration.

3.5 Anderson Acceleration

Anderson Acceleration [3, 55, 62] is a method for accelerating the convergence of fixed point iteration

$$u = G(u), \tag{3.33}$$

originally developed for electronic structure calculations. It has recently seen a resurgence in popularity with many black box, code coupling problems [21]. We define the residuals of this fixed point map as

$$F(u) = G(u) - u. \tag{3.34}$$

Accelerating source iteration, we use

$$F(\phi) = \mathbf{S}(\phi) - \phi. \tag{3.35}$$

The algorithm stores and uses, along with the residuals, the m most recent iterates. The Anderson algorithm which stores m iterates called Anderson(m). If we choose not to store any iterates then Anderson(0) is equal to fixed point iteration by design. Algorithm 2 is an overview of Anderson acceleration.

Algorithm 2 Anderson(m)

`anderson`$(u_0, \mathbf{S}, m)$

$u_1 = \mathbf{S}(u_0)$; $F_0 = \mathbf{S}(u_0) - u_0$

for $k = 1, \ldots$ **do**

$m_k = \min(m, k)$

$F_k = \mathbf{S}(u_k) - u_k$

Minimize $\| \sum_{j=0}^{m_k} \alpha_j^k F_{k-m_k+j} \|_2$ subject to

$\sum_{j=0}^{m_k} \alpha_j^k = 1.$

$u_{k+1} = (1 - \beta_k) \sum_{j=0}^{m_k} \alpha_j^k u_{k-m_k+j} + \beta_k \sum_{j=0}^{m_k} \alpha_j^k G(u_{k-m_k+j})$

end for

Anderson acceleration solves a linear least squares problem for the Anderson coefficients $\{\alpha_j^k\}$. The new iterate is defined as a linear combination of the iterates and the fixed point evaluations. The β_k terms are mixing parameters.

Algorithm 2 converges if the fixed point iteration converges (i.e. G is a contraction mapping), u_0 close enough to the solution, and $\{\alpha_j^k\}$ stay bounded throughout the iteration [55]. It is proven that in linear case the convergence rate of Anderson is no worse than fixed point iteration. In the nonlinear case, we can make the convergence speed arbitrarily close to fixed point iteration. We discuss Anderson acceleration under noisy function evaluations, once we have introduced stochastic methods for neutron transport in Chapter 4.

3.6 Nonlinear Diffusion Acceleration

Nonlinear diffusion acceleration (NDA) [32, 51, 65] is an acceleration method that uses the zeroth moment of the 1D transport equation. In slab geometry, the zeroth moment is

$$\frac{dJ}{dx}(x) + (\Sigma_t - \Sigma_s)\,\phi(x) = q(x), \tag{3.36}$$

where J is the current. In Section 3.3, we described a framework to solve the transport equation. We present a *low-order* (LO) problem that uses ψ from the transport sweep. We solve the LO problem for the next iterate $\phi^{(+)}$ using a specific closure relation for J.

Components of the framework include the *high-order* (HO) and LO problems. The HO problem is computing a transport sweep with the most recent scalar flux iterate. The angular flux resulting from the transport sweep is ψ^{HO} and the scalar flux is ϕ^{HO}. The LO problem is a function $G(\psi^{HO}, \phi^{HO})$ which outputs the new iterate $\phi^{(+)}$. For source iteration, the LO problem is simply $G(\psi^{HO}, \phi^{HO}) = \phi^{HO}$. For NDA, we model transport as diffusion to formulate the LO problem.

3.6.1 NDA Derivation

We follow the same steps and formulation as found in [32]. Recall, the HO problem for ψ^{HO} is

$$\mu \frac{\partial \psi^{HO}}{\partial x} + \Sigma_t \psi^{HO} = \frac{1}{2}\left[\Sigma_s \phi^{(c)} + q\right]. \tag{3.37}$$

First, we define the current:

Definition 3.6.1. *J is the neutron current. $J(\vec{r}, E, t)\, d^3\vec{r}\, dE\, dA$ is the expected net number of particles crossing area dA moving with velocity $v(E)$ in volume $d^3\vec{r}$ about $\vec{r}$, having energies in dE around E, and at time t.*

The current is

$$J(x) = \int_{-1}^{1} \mu\psi(x,\mu)\, d\mu, \tag{3.38}$$

the first moment of the angular flux. Analogously, the scalar flux is the zeroth moment, because

$$\phi(x) = \int_{-1}^{1} \mu^0\psi(x,\mu)\, d\mu. \tag{3.39}$$

Next, the zeroth moment of the transport equation is

$$\begin{aligned}
\int_{-1}^{1}\left[\mu\frac{\partial\psi}{\partial x}(x,\mu) + \Sigma_t\psi(x,\mu)\right] d\mu &= \int_{-1}^{1}\left[\frac{\Sigma_s}{2}\phi(x) + \frac{1}{2}q(x)\right] d\mu,\\
\int_{-1}^{1}\mu\frac{\partial\psi}{\partial x}(x,\mu)\, d\mu + \int_{-1}^{1}\Sigma_t\psi(x,\mu)\, d\mu &= \int_{-1}^{1}\left[\frac{\Sigma_s}{2}\phi(x) + \frac{1}{2}q(x)\right] d\mu\\
\frac{d}{dx}\int_{-1}^{1}\mu\psi(x,\mu)\, d\mu + \Sigma_t\int_{-1}^{1}\psi(x,\mu)\, d\mu &= \int_{-1}^{1}\left[\frac{\Sigma_s}{2}\phi(x) + \frac{1}{2}q(x)\right] d\mu,\\
\frac{dJ}{dx}(x) + \Sigma_t\phi(x) &= \Sigma_s\phi(x) + q(x),\\
\frac{dJ}{dx}(x) + (\Sigma_t - \Sigma_s)\,\phi(x) &= q(x).
\end{aligned} \tag{3.40}$$

The crux of NDA is the approximation of J using Fick's law of diffusion and a consistency term. We approximate the current as

$$J = -\frac{1}{3\Sigma_t}\frac{d\phi}{dx} + \hat{D}\phi, \tag{3.41}$$

where $\hat{D}$ is a vector. Combining the two equations, the LO problem is

$$\frac{d}{dx}\left[-\frac{1}{3\Sigma_t}\frac{d\phi}{dx} + \hat{D}\phi\right] + (\Sigma_t - \Sigma_s)\phi = q, \tag{3.42}$$

which we solve for ϕ. This resulting ϕ is next iterate in the NDA fixed point map. We calculate $\hat{D}$ in terms of known quantities from the HO problem. The $\hat{D}$-equation is

$$\hat{D} = \frac{J^{HO} + \frac{1}{3\Sigma_t}\frac{d\phi^{HO}}{dx}}{\phi^{HO}}, \tag{3.43}$$

with definitions

$$J^{HO} = \int_{-1}^{1} \mu\psi^{HO}(x,\mu)\, d\mu \tag{3.44}$$

and

$$\phi^{HO} = \int_{-1}^{1} \psi^{HO}(x,\mu)\, d\mu. \tag{3.45}$$

$\hat{D}$ is a function explicitly depending on J^{HO} and ϕ^{HO}, or $\hat{D}(J^{HO}, \phi^{HO})$. We embed $\hat{D}$ into (3.42), to obtain

$$\frac{d}{dx}\left[-\frac{1}{3\Sigma_t}\frac{d\phi}{dx} + \hat{D}(J^{HO}, \phi^{HO})\phi\right] + (\Sigma_t - \Sigma_s)\phi = q. \tag{3.46}$$

A full overview of NDA is seen in Algorithm 3.

Algorithm 3 NDA

nda($\phi^{(0)}, \tau$)

Begin with initial iterate $\phi^{(0)}$ and tolerance τ.
$r = \infty$.
Set $\phi^{(c)} = \phi^{(0)}$.
while $r > \tau$ **do**
  Solve HO problem for ψ^{HO} from $\phi^{(c)}$.
  Calculate J^{HO} and ϕ^{HO} from ψ^{HO}.
  Compute $\hat{D}$ from (3.43).
  Solve LO problem for $\phi^{(+)}$ using $\hat{D}$.
  $r = \|\phi^{(+)} - \phi^{(c)}\|_\infty$.
  Update $\phi^{(c)} = \phi^{(+)}$.
end while

3.6.2 NDA Discretization

For the HO problem, we use the *diamond difference* discretization from [32]

$$\mu^j \frac{\psi^j_{i+1/2} - \psi^j_{i-1/2}}{\Delta x} + \Sigma_t \frac{\psi^j_{i+1/2} + \psi^j_{i-1/2}}{2} = \Sigma_s \phi_i + q_i, \tag{3.47}$$

where ϕ_i is a cell-averaged quantity and $\psi_{i+1/2}$ is a cell-edge quantity. We consider here only a consistent discretization for the HO and LO problem. The μ^j are chosen to be the Gauss nodes and have corresponding weights w_j, so that

$$\phi_i = \frac{1}{2}\int_{-1}^{-1} (\psi_{i+1/2} + \psi_{i-1/2})\, d\mu = \sum_j \frac{w_j}{2}\left(\psi^j_{i+1/2} - \psi^j_{i-1/2}\right). \tag{3.48}$$

The discretized closure relation is

$$J^{HO}_{i+1/2} = -\frac{1}{3\Sigma_t}\frac{\phi^{HO}_{i+1} - \phi^{HO}_i}{\Delta x} + \frac{\hat{D}_{i+1/2}}{2}(\phi^{HO}_{i+1} + \phi^{HO}_i), \tag{3.49}$$

where

$$J^{HO}_{i+1/2} = \int_{-1}^{-1} \mu \psi^{HO}_{i+1/2}\, d\mu \tag{3.50}$$

and

$$\phi^{HO}_i = \frac{1}{2}\int_{-1}^{-1} (\psi^{HO}_{i+1/2} + \psi^{HO}_{i-1/2})\, d\mu. \tag{3.51}$$

We directly solve (3.49) for $\hat{D}_{i+1/2}$. We use $\hat{D}_{i+1/2}$ to solve the discrete version of (3.46) to obtain

$$-\frac{1}{3\Sigma_t}\frac{\phi_{i+1} - 2\phi_i + \phi_{i-1}}{\Delta x} + \frac{\hat{D}_{i+1/2}}{2\Delta x}(\phi_{i+1} + \phi_i) - \frac{\hat{D}_{i-1/2}}{2\Delta x}(\phi_i + \phi_{i-1}) + (\Sigma_t - \Sigma_s)\phi_i = q_i. \tag{3.52}$$

The LO boundary conditions are chosen to maintain consistency between the HO and LO scalar fluxes and currents. Specifically, we set

$$\frac{J^{LO}_{1/2}}{\phi^{LO}_1} = \frac{J^{HO}_{1/2}}{\phi^{HO}_1}. \tag{3.53}$$

With this formulation, ϕ^{HO} and ϕ^{LO} are consistent to within the tolerance τ once the iteration terminates.

3.7 Solving the k-eigenvalue Problem from the Multigroup Equations

The multigroup equations discretized in angle in 1D slab geometry with isotropic scattering are

$$\mu\frac{\partial\psi_g}{\partial x}(x,\mu)+\Sigma_{t,g}(x)\,\psi_g(x,\mu)=\frac{1}{2}\sum_{g'=1}^{G}\Sigma_{s,g'\to g}(x)\,\phi_{g'}(x)+\frac{\chi_g(x)}{2k}\sum_{g'=1}^{G}\nu_{g'}(x)\,\Sigma_{f,g'}(x)\,\phi_{g'}(x)+\frac{1}{2}Q_g(x), \quad (3.54)$$

for each group g for G total equations. We assume vacuum boundary conditions for all groups. The solution pair is ϕ_g and multiplicative factor k. We rewrite the scattering term as

$$\mu\frac{\partial\psi_g}{\partial x}(x,\mu)+\Sigma_{t,g}(x)\,\psi_g(x,\mu)=\frac{1}{2}\sum_{g'=1}^{g-1}\Sigma_{s,g'\to g}(x)\,\phi_{g'}(x)+\frac{1}{2}\Sigma_{s,g\to g}(x)\,\phi_g(x)+\frac{1}{2}\sum_{g'=g+1}^{G}\Sigma_{s,g'\to g}(x)\,\phi_{g'}(x)+\frac{\chi_g(x)}{2k}\sum_{g'=1}^{G}\nu_{g'}(x)\,\Sigma_{f,g'}(x)\,\phi_{g'}(x)+\frac{1}{2}Q_g(x), \quad (3.55)$$

where it is separated into down-scattering, in-scattering, and up-scattering. The *power iteration* method is

$$\mu\frac{\partial\psi_g^{(s)}}{\partial x}(x,\mu)+\Sigma_{t,g}(x)\,\psi_g^{(s)}(x,\mu)=\frac{1}{2}\sum_{g'=1}^{g-1}\Sigma_{s,g'\to g}(x)\,\phi_{g'}^{(s)}(x)+\frac{1}{2}\Sigma_{s,g\to g}(x)\,\phi_g^{(s)}(x)+\frac{1}{2}\sum_{g'=g+1}^{G}\Sigma_{s,g'\to g}(x)\,\phi_{g'}^{(s-1)}(x)+\frac{\chi_g(x)}{2k}\sum_{g'=1}^{G}\nu_{g'}(x)\,\Sigma_{f,g'}(x)\,\phi_{g'}^{(s-1)}(x)+\frac{1}{2}Q_g(x), \quad (3.56)$$

$$k^{(s)}=\frac{\sum_{g=1}^{G}\int_0^X dx\,\nu_g(x)\,\Sigma_{f,g}(x)\phi_g^{(s)}(x)}{\sum_{g=1}^{G}\left[J_g^{(s)}(X)-J_g^{(s)}(0)\right]+\sum_{g=1}^{G}\int_0^X dx\,\Sigma_{a,g}(x)\phi_g^{(s)}(x)} \quad (3.57)$$

for a sequence of iterates over index s. We lag the fission and up-scattering terms in (3.56) as in [37]. In (3.57), $k^{(s)}$ is a ratio of neutron production and neutron loss at the s iteration. The loss term is composed

of leakage out the boundaries and absorption in the domain for all g. We define a new source term,

$$\tilde{Q}_g(x) = \frac{1}{2}\sum_{g'=1}^{g-1} \Sigma_{s,g'\to g}(x)\,\phi_{g'}^{(s)}(x) + \frac{1}{2}\sum_{g'=g+1}^{G} \Sigma_{s,g'\to g}(x)\,\phi_{g'}^{(s-1)}(x) + \frac{\chi_g(x)}{2k^{(s-1)}}\sum_{g'=1}^{G} \nu_{g'}(x)\,\Sigma_{f,g'}(x)\;\phi_{g'}^{(s-1)}(x) + \frac{1}{2}Q_g(x), \quad (3.58)$$

which contains all terms except for the in-scattering. We define each component as

$$\tilde{Q}_g^{down(s)}(x) = \frac{1}{2}\sum_{g'=1}^{g-1} \Sigma_{s,g'\to g}(x)\,\phi_{g'}^{(s)}(x) \quad (3.59)$$

$$\tilde{Q}_g^{up(s)}(x) = \frac{1}{2}\sum_{g'=g+1}^{G} \Sigma_{s,g'\to g}(x)\,\phi_{g'}^{(s-1)}(x), \quad (3.60)$$

and

$$\tilde{Q}_g^{fission(s)}(x) = \frac{\chi_g(x)}{2k^{(s-1)}}\sum_{g'=1}^{G} \nu_{g'}(x)\Sigma_{f,g'}(x)\phi_{g'}^{(s-1)}(x). \quad (3.61)$$

The result is a one-group fixed-source problem at every iteration and in each group,

$$\mu\frac{\partial\psi_g^{(s)}}{\partial x}(x,\mu) + \Sigma_{t,g}(x)\,\psi_g^{(s)}(x,\mu) = \frac{1}{2}\Sigma_{s,g\to g}(x)\,\psi_g^{(s)}(x) + \tilde{Q}_g(x). \quad (3.62)$$

We solve (3.62) using any of the methods we have introduced, such as NDA. Within each iteration s, we calculate ϕ_g for the highest energy group, $g = 1$, first. We continue incrementally until $g = G$. We have two edge cases for $\tilde{Q}_g$. When $g = 1$, we have $\tilde{Q}_1^{down(s)} = 0$. When $g = G$, we have $\tilde{Q}_G^{up(s)} = 0$. Power iteration continues until ϕ_g and k converge to a provided tolerance. A full overview of the multigroup algorithm is seen in Algorithm 4.

Algorithm 4 Multigroup

`multigroup`$(\phi_g^{(0)}, k^{(0)}, \tau_\phi, \tau_k, \tau_{\phi_g})$

Begin with initial iterate $\phi^{(0)}$, $k^{(0)}$ and tolerance τ.

$s = 0$.

$r_\phi = \infty$.

$r_k = \infty$.

while $r_\phi > \tau_\phi$ or $r_k > \tau_k$ **do**

$s = s + 1$.

for g = 1, . . . , G **do**

Calculate $\tilde{Q}_g^{up(s-1)}$ and $\tilde{Q}_g^{fission(s-1)}$.

end for

for g = 1, . . . , G **do**

Calculate $\tilde{Q}_g^{down(s)}$.

Set $\phi_g^{(s),c} = \phi_g^{(0)}$.

$r_{\phi_g} = \infty$

while $r_{\phi_g} > \tau_{\phi_g}$ **do**

Solve HO problem (3.62) for $\psi_g^{(s),+}$ from $\phi_g^{(s),c}$ and $\tilde{Q}_g$.

Solve LO problem for $\phi_g^{(s),+}$.

$r_{\phi_g} = \|\phi_g^{(s),+} - \phi_g^{(s),c}\|$.

Update $\phi_g^{(s),c} = \phi_g^{(s),+}$.

end while

Set $\phi_g^{(s)} = \phi_g^{(s),+}$.

end for

Calculate $k^{(s)}$.

$r_\phi = \|\phi^{(s+1)} - \phi^{(s)}\|$.

$r_\phi = |k^{(s+1)} - k^{(s)}|$.

end while

Chapter 4

Introduction to Monte Carlo for Neutronics

The Monte Carlo method [39, 52] simulates a finite number of particle histories inside a given domain through random number generation. Each particle samples known *probability distribution functions* for particle location, direction, and energy. Particles complete flights from birth to death, either through absorption or leaving the system. During their histories, particles stream, scatter, fission, or cross material boundaries. Random number generation also determines collision location and collision type based on the material cross-sections. After the simulation, we calculate a mean value estimate, $\hat{x}$, of some desired quantity. Given an N_p-sample Monte Carlo simulation, the mean value estimate is

$$\hat{x} = \frac{1}{N_p} \sum_{n=1}^{N_p} x_n, \tag{4.1}$$

where x_n is the contribution from the n^{th} sample. So, we *tally* the x_n due to each history to estimate $\hat{x}$ at the end of the calculation. Examples of x_n include total number of collisions and the total track length distance traveled. Tallies are objects in the Monte Carlo method providing users with flux estimates.

All estimates contain stochastic error because we use a finite number of samples, but the variance in those estimates decreases as the number of samples increases. The uncertainty in the Monte Carlo estimate is known to converge $O(1/\sqrt{N_p})$ [37]. The attractive features of Monte Carlo include continuous treatment of angle, space, and potentially energy; therefore, the Monte Carlo simulation has no discretization error. For deterministic transport methods, we use discrete grids for each variable in the phase space. Each discretization is an additional approximation that adds to the final solution's overall error. Monte Carlo avoids discretizations in exchange for stochastic error. We reduce the stochastic error by taking more samples and through variance reduction techniques. Also, we may parallelize the Monte Carlo method. A particle history is independent from all other histories and may be computed on sep-

arate processors. Once all histories are completed, one designated processor calculates and reports the mean value estimate.

We begin this chapter with *analog* Monte Carlo and its components. We discuss variance reduction techniques in the coming sections as well.

4.1 Analog Monte Carlo

Analog Monte Carlo is a basic form of Monte Carlo for neutronics. We focus on 1D monoenergetic transport with constant cross sections and isotropic scattering in slab geometry.

First, *probability density functions* are the tools we use to relate events and quantities with randomly generated numbers.

Definition 4.1.1. *A probability density function $f(x)$ describes the probability of a random variable taking certain values from the sample space.*

The probability of the random variable x falling in the interval $[a, b]$ is

$$P\{a \leq x \leq b\} = \int_a^b f(x)\, dx. \tag{4.2}$$

A density function has the requirements that it is non-negative and

$$\int_\infty^\infty f(x)\, dx = 1. \tag{4.3}$$

Any non-negative integrable function g has a corresponding density function,

$$f(x) = \frac{g(x)}{\int_{-\infty}^{\infty} g(x)\, dx}. \tag{4.4}$$

The *cumulative distribution function* (CDF) is

$$F(a) = P\{x \leq a\} = \int_{-\infty}^{a} f(x)\, dx. \tag{4.5}$$

It is a monotone increasing function that describes the probability that the random variable is less than the value a. From the definition, we see

$$\lim_{x\to\infty} F(x) \equiv F(\infty) = 1 \tag{4.6}$$

and

$$\lim_{x\to-\infty} F(x) \equiv F(-\infty) = 0. \tag{4.7}$$

We restate (4.2) in terms of CDFs as

$$P\{a \leq x \leq b\} = F(b) - F(a). \tag{4.8}$$

Random numbers ξ are generated uniformly on $[0, 1]$, the range space of the CDF. We transform ξ as

$$F(x) = \xi, \tag{4.9}$$

or

$$x = F^{-1}(\xi), \tag{4.10}$$

to obtain a sample of x. The inversion of the F is not trivial. We introduce techniques for this inversion for relevant cases.

4.1.1 Components of Monte Carlo

The Monte Carlo components include starting location, travel distance, direction, and collision type. We obtain each by sampling distributions and computing the CDF inversions when needed.

The first CDF inversion determines the number of mean free paths the particle travels. the distribution function for mean free paths from [37] is

$$f(x) = e^{-x} \tag{4.11}$$

and the CDF is

$$F(x) = 1 - e^{-x}. \tag{4.12}$$

The inversion of F is

$$x = -\ln(1 - \xi), \tag{4.13}$$

which we simplify to

$$x = -\ln(\xi). \tag{4.14}$$

ξ is uniform on $[0, 1]$, so $1-\xi$ is also uniform on $[0, 1]$. This simplification reduces excessive computation which is important when simulating large numbers of histories. Given a number of mean free paths, the streaming distance is

$$x_{strm} = \frac{x}{\Sigma_t}. \tag{4.15}$$

A second CDF inversion finds the starting particle locations. Assume we have a piecewise constant source density $s(x) \in [0, \infty)$ with $x \in [0, \tau]$, like in [37]. We choose a starting particle location with a

random number through the CDF of $s(x)$. The density function is

$$s(x) = \{x \in [x_{i-1}, x_i) \mid s_i\}, \tag{4.16}$$

for $i = 1, \ldots, M$. The piecewise constant density function leads to the piecewise linear CDF

$$S(x) = \frac{1}{x_i - x_{i-1}} \left[(x - x_{i-1})S_i + (x_i - x)S_{i-1}\right], \tag{4.17}$$

where

$$S_i = \sum_{j=1}^{i} (x_j - x_{j-1})s_j. \tag{4.18}$$

To find $x = S^{-1}(\xi)$, we first find which interval $[S_{i-1}, S_i]$ that ξ is located. Then we solve (4.17), $S(x) = \xi$, to obtain

$$x_0 = \frac{(x_i - x_{i-1})\xi - x_i S_{i-1} + x_{i-1} S_i}{S_i - S_{i-1}}, \tag{4.19}$$

where x_0 is the particle's starting location.

Direction μ for an isotropic source and isotropic scattering is sample directly from a uniform distribution. All angles are equally likely, so we sample from $Uni(-1, 1)$. With $\xi \in Uni(0, 1)$, the translation is

$$\mu(\xi) = 2\xi - 1. \tag{4.20}$$

The final probabilistic event is the collision. Given a starting location, a direction, and a travel distance, we compute the collision locations,

$$x_{coll} = x_0 + \mu \cdot x_{strm}. \tag{4.21}$$

If x_{coll} is outside the domain, we terminate the history and proceed to the next. If x_{coll} is inside the domain, we collide. Collisions are determined randomly, as well. The cross section data determines the likelihoods of each collision type occurring. Assume the fission-free case, then we know

$$\Sigma_t = \Sigma_s + \Sigma_a. \tag{4.22}$$

If we sample ξ from $Uni(0, 1)$, then the particle undergoes scattering for

$$\xi \leq \frac{\Sigma_s}{\Sigma_t} \tag{4.23}$$

and undergoes absorption for

$$\xi > \frac{\Sigma_s}{\Sigma_t}. \tag{4.24}$$

When a particle is absorbed, we terminate the history and proceed to the next. When a particle scatters, we sample a new μ and x_{strm}. This process continues until the particle leaves the system. After all N_p histories are completed, we compute the quantities of interest from the tallies.

4.1.2 Analog Monte Carlo Tally

In analog Monte Carlo, we tally information for each history as the simulation runs. The information is averaged over the N_p histories to obtain a mean value with an associated variance. We use a weighted average, where each particle has weight, w_n. The uniform weights are

$$w_n = \frac{1}{N_p} \int_0^\tau s(x)\, dx. \tag{4.25}$$

We discuss two types of tallies for $\hat{\phi}$: the *track-length* tally and the *collision estimator* tally. The track-length tally stores the flight paths of the particles, ℓ_n. Specifically, ℓ_n is the distance traveled by the n^{th} particle in a volume V. We estimate the scalar flux $\hat{\phi}$ in a volume V with N_p histories as

$$\hat{\phi} = \frac{1}{N_p V} \sum_{n=1}^{N_p} w_n \ell_n. \tag{4.26}$$

The collision estimator tally stores the number of collisions c_n in V. The collision density is

$$\hat{c} = V \Sigma_t \hat{\phi}, \tag{4.27}$$

or the mean number of collisions in V. Solve for $\hat{\phi}$, to obtain

$$\hat{\phi} = \frac{1}{V \Sigma_t} \hat{c}, \tag{4.28}$$

where

$$\hat{c} = \frac{1}{N_p} \sum_{n=1}^{N} w_n c_n. \tag{4.29}$$

Similarly, we estimate $\hat{J}$ through a surface of area A using the surface crossing tally. The estimate is

$$\hat{J} = \frac{1}{N_p A} \sum_{n=1}^{N_p} sign(\mu) w_n z_n, \tag{4.30}$$

where z_n is the number of crossings of face A for the n^{th} particle. We present a full overview of analog Monte Carlo method in Algorithm 3.

Algorithm 5 Analog Monte Carlo

`Analog MC`(s, N)

Begin with source distribution s and number of histories N.
Compute total source.
Normalize s to $\bar{s}$ and find the CDF, F.
Initialize weights w_n.
for $n = 1, \ldots, N$ **do**
  Randomly sample ξ for location.
  Solve $F(x_0) = \xi$ for x_0.
  Randomly sample direction μ.
  while n^{th} particle remains in system **do**
    Randomly sample streaming distance.
    Compute collision location.
    if Particle n leaves domain **then**
      Terminate history.
    end if
    Randomly sample for collision type.
    if Particle n scatters **then**
      Compute new direction μ for next flight.
    else if Particle n is absorbed **then**
      Terminate history.
    end if
    Store tallies.
  end while
end for
Calculate mean value from tallies.

4.2 Variance Reduction

The variance for a random variable x is

$$\sigma^2(x) = E\left[(x - \hat{x})^2\right] = \int_{-\infty}^{\infty} dx\,(x - \hat{x})^2 f(x), \tag{4.31}$$

where E is the expected value operator and f is the probability density function of x. The expected value of x is mean value we obtain as the number of samples $N_p \to \infty$. So, the variance is the expected value of the second moment about the mean.

The Monte Carlo method calculates a mean value

$$\hat{x} = \int x f(x)\, dx \tag{4.32}$$

with an associated variance. Choose a modified probability density function, $\tilde{f}$, to induce a weight function, $w(x)$, such that

$$\tilde{f}(x)w(x) = f(x). \tag{4.33}$$

Then, (4.32) is equal to

$$\hat{x} = \int x w(x)\tilde{f}(x)\, dx. \tag{4.34}$$

Appropriate choices of $\tilde{f}$ lead to an unbiased estimator with smaller variance. The estimator is unbiased because $\hat{x}$ remains unchanged. We may perform a non-analog Monte Carlo process, represented by $\tilde{f}$, that produces a more precise $\hat{x}$. Non-analog steps may occur during collisions, boundary crossings, or other events.

4.2.1 Implicit Capture and Russian Roulette

Implicit capture trades any chance of absorption for particle weight reduction [37, 52]. All particles are forced to scatter in the non-fission case. The particle's weight after scattering is

$$w_{n,i+1} = w_{n,i}\left(1 - \frac{\Sigma_a}{\Sigma_t}\right), \tag{4.35}$$

where $w_{n,i}$ is the weight of the n^{th} particle after the i^{th} collision. The initial weights $w_{n,0}$ are equal to the uniform weights in (4.25). The N_p particles remain in the system for more events leading to longer histories. This increases the computational runtime of each history. In exchange, longer histories increase expected number of particles that contributes to the estimator, thereby lowering the variance [37].

The loss of absorption forces us to formulate a new termination criterion. Otherwise, histories only end when they leak out the domain boundary. Weights quickly decrease for low-scattering ratios, so their contributions to the tallies become negligible. Killing histories when weights fall below a threshold saves runtime. Importantly, if we outright kill small-weighted particles, we bias the estimator. We cannot remove the tails of histories without changing $w_{n,i}$. One solution is *Russian roulette*.

Russian roulette occurs when a particle falls below a user chosen weight threshold. We sample ξ from $Uni(0, 1)$ and check the inequality

$$\xi > \frac{1}{d}, \tag{4.36}$$

where d is a chosen parameter. If the inequality holds we kill the history. If the inequality does not hold we increase the weight by the factor d. This ensures the estimator is unbiased while still reducing the

variance through implicit capture.

4.2.2 Particle Splitting

Particles are split when they become highly important, for example, when entering particular regions of interest. It is advantageous to split the particle at a collision site or boundary crossing into d particles. Figure 4.1 shows particles splitting upon entering an important region. The d particles continue with their flights as normal particles and are terminated under the same criteria. Each of the d particles have $1/d$ of the original particle weight to preserve the unbiased estimators. This process is known to reduce variance in the calculation, but increase computational time [37]. Each particle contribution is smaller, but the number of contributions is increased. More non-zero samples for the estimator leads to reduced variance in the mean estimate.

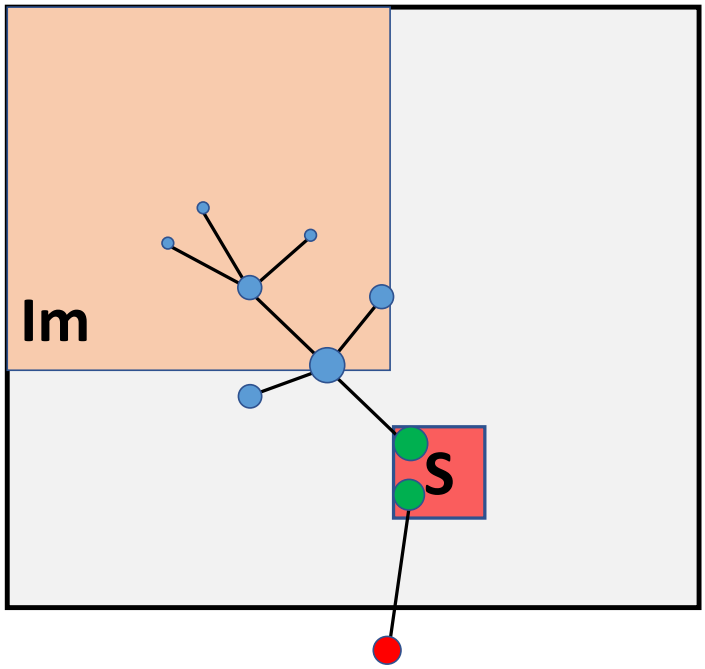


Figure 4.1: Monte Carlo particles splitting in an important region with $d = 3$.

Consider the case where we desire flux values at a location far away from any neutron source. The likelihood particles pass through this region is low. If we split the few particles that travel through that region, there are greater number of particles in the region of interest. More particles leads to an increased number of non-zero contributions. Particle splitting is essential for modeling low probability events with

Monte Carlo. Further variance reduction techniques are found in [37].

4.3 Monte Carlo Criticality Calculations

Criticality calculations with Monte Carlo measure the multiplication factor of the system. The system contains fissionable material and particles may spawn additional fission neutrons during the simulation. The k-eigenvalue problem is solved using the Monte Carlo version of *power iteration*. Let f^ℓ and k^{ell} be the eigenvector-eigenvalue pair corresponding to the fission source at generation ℓ and the multiplication factor. Then the iteration is

$$f^{\ell+1} = \frac{1}{k^\ell}\hat{\mathbf{A}} f^\ell \tag{4.37}$$

$$k^{\ell+1} = \frac{\int \hat{\mathbf{A}} f^\ell d\Gamma}{\int f^\ell d\Gamma} = k^\ell \frac{\int f^{\ell+1} d\Gamma}{\int f^\ell d\Gamma}, \tag{4.38}$$

where $\hat{\mathbf{A}}$ is the transport operator and Γ is the phase space of f.

A *cycle* is a single advancement of all particles to their next event. The algorithm calculates the fission source for the next cycle from the previous cycle. As generations continue, fission produces additional neutrons and absorption removes fission neutrons. The sequence k^ℓ converges to the k-eigenvalue within statistical noise after a significant number of cycles [37].

With the introduction of fissionable material, we may never simulate all neutrons from birth to death. The addition of fission neutrons each cycle makes this impossible. Therefore, we select a total number of cycles, L_C, to simulate for the criticality calculation. We divide L_C in two phases: *inactive* cycles and *active* cycles. Inactive cycles do not gather any tallies, and instead are used to converge the fission source. Power iteration is slow to converge when the *dominance ratio* of the system is near unity [28]. The dominance ratio is $|k_2|/|k_1|$, where k_1 is the largest eigenvalue and k_2 is the second largest eigenvalue. Estimates are poor when fission is not sufficiently converged. During active cycles, we record tallies that form the estimates. Methods are available that accelerate fission convergence [37]. Standard Monte Carlo power iteration is presented in Algorithm 6.

Algorithm 6 MC Power Iteration

`power_iteration`(f^0, L_C)
  Sample an initial fission source, f^0, for cycle $\ell = 0$.
  for $\ell \in \{1, \cdots, L_C\}$ **do**
    Perform N random walks of fission neutrons.
    Sample the fission source for the next iteration, $f^{\ell+1}$
    Tally $k^{\ell+1}$ during active cycles
  end for

Chapter 5

Hybrid Methods for Neutronics

Hybrid methods for neutronics combine stochastic and deterministic methods. We use deterministic methods to accelerate the underlying stochastic method, a Monte Carlo simulation. We investigate two hybrid techniques: Monte Carlo embedded in HO-LO framework and Monte Carlo informed by a coarse deterministic solution. The second technique is *consistent adjoint driven importance sampling* (CADIS).

5.1 Monte Carlo High Order Problem

For source iteration, we invert the streaming, absorption, and scattering operators to solve the fixed source problem

$$\hat{\Omega} \cdot \nabla \psi_+(\vec{r}, \hat{\Omega}) + \Sigma_t \psi_+(\vec{r}, \hat{\Omega}) = \frac{1}{4\pi}[\Sigma_s(\vec{r})\phi_c(\vec{r}) + q(\vec{r})]. \tag{5.1}$$

The *Monte Carlo transport sweep* in [65] solves (5.1) by considering the scattering operator as a component of a new source term

$$S_c(\vec{r}) = \Sigma_s(\vec{r})\phi_c(\vec{r}) + q(\vec{r}), \tag{5.2}$$

then it completes a MC simulation without any scattering. Thus, we solve

$$\hat{\Omega} \cdot \nabla \psi_+(\vec{r}, \hat{\Omega}) + \Sigma_t \psi_+(\vec{r}, \hat{\Omega}) = \frac{1}{4\pi} S_c(\vec{r}). \tag{5.3}$$

The MC transport sweep allows particles to take a single flight before they are absorbed. This transport sweep is the HO problem for HO-LO framework in Chapter 3. A full Monte Carlo simulation converges scattering. Here, we complete multiple simple MC transport sweeps each iteration and converge scattering as the iteration converges.

The Monte Carlo embedded algorithm is seen in Algorithm 7.

Algorithm 7 MC Transport

`mc_transport`(ϕ_0, τ)

Begin with initial iterate ϕ_0 and tolerance τ.
$r = \infty$.
Set $\phi_c = \phi_0$.
while $r > \tau$ **do**
  $S_c = \Sigma_s \phi_c + q$
  Compute scatter-free MC solution $\psi_+^{MC,HO}$ from S_c.
  Solve LO problem for ϕ_+ from $\psi_+^{MC,HO}$.
  $r = \|\phi_+ - \phi_c\|_\infty$.
  Update $\phi_c = \phi_+$.
end while

In our results, we apply Anderson acceleration to the MC Transport with $F(\phi) = G(\phi) - \phi$, where $G(\phi)$ is the function that takes ϕ_c to ϕ_+. G may contain an accelerator as its LO problem, such as NDA.

5.1.1 Numerical Results

We consider two monoenergetic fixed source problems in single material slab geometry with isotropic scattering. We consider the moderate and high scattering cases where the scattering ratio $c = \Sigma_s/\Sigma_t = .80$ and .99. We apply source iteration and NDA to each case with the Profugus code [15] as the HO solve. In addition, we accelerate both of these methods using Anderson(2).

We fix the number of histories for each Profugus computation. For $c = .80$, we have $N_{MC} = 1e6$ during each step of the iteration. The same is repeated for $c = .99$ but instead with $N_{MC} = 4e6$ histories. The high scattering case requires a higher number of particles because the mean spectral radius is near unity. For a deterministic transport sweep the spectral radius of source iteration is bounded above by the scattering ratio. Moderate stochastic error in the function evaluations may lead to poor stability iteration to iteration and even divergence. This occurs because the error in the stochastic transport sweep is too large, ultimately leading to loss of contractivity.

We use the parameter values in Table 5.1 from [64] along with the moderate scattering case. In Figures 5.1 and 5.2, we plot the average relative residuals over 10 independent runs on the y-axis and the cumulative number of particle histories on the x-axis, with each marker indicating an iterate's residual. We ran each of the iterations until the error in the function evaluation stagnates the residual reduction.

Anderson(2) accelerates source iteration and provides marginal speed up of nonlinear diffusion acceleration. In Figure 5.1, Anderson(2) quickly reduces the residual relative to source iteration until the noise in the function evaluation dominates. Source iteration stagnates at the same residual as Ander-

Table 5.1: Problem Data from [64]

Parameter	Case 1	Case 2
Σ_t	10	10
Σ_s	8.0	9.9
τ	1	1
q	.5	.5
Spatial Cells	50	50

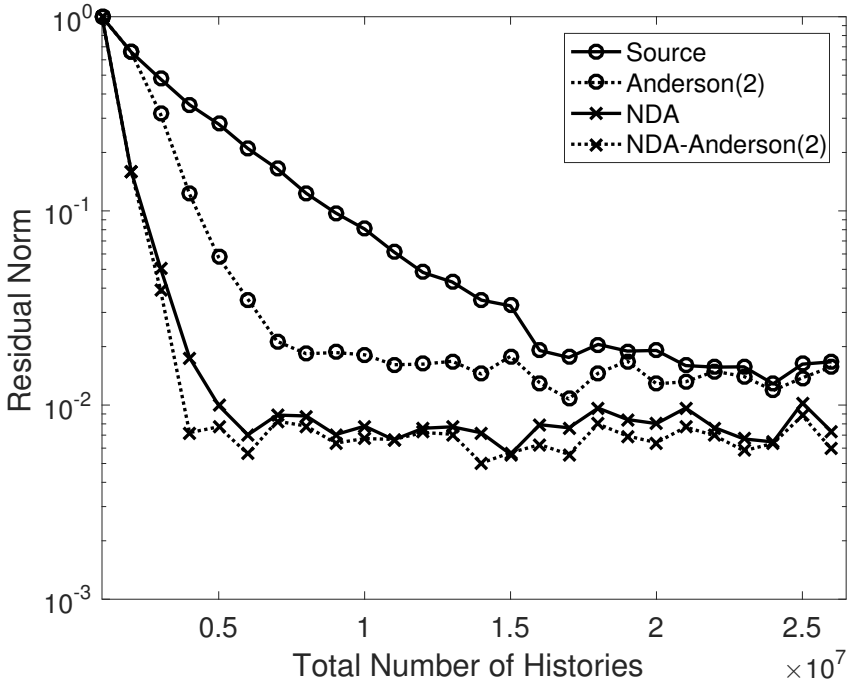


Figure 5.1: Residual histories of Profugus dependent fixed source calculations with $c = .80$.

son(2), but the convergence is slower as expected. Comparing NDA and NDA-Anderson(2), we observe that the two methods have roughly the same residual history and stagnate simultaneously. This supports the theory in [54] that states Anderson does no worse than Picard iteration if the noise is well bounded with high probability.

In Figure 5.2, source iteration with $c = .99$ has difficulty reducing the residual each iteration. Anderson(2) lowers the spectral radius of source iteration and achieves faster convergence. Though not plotted in the high scattering case, source iteration and its Anderson accelerated version stagnate at roughly 70 and 30 iterations, respectively. Again, NDA and NDA-Anderson(2) follow the same convergence and

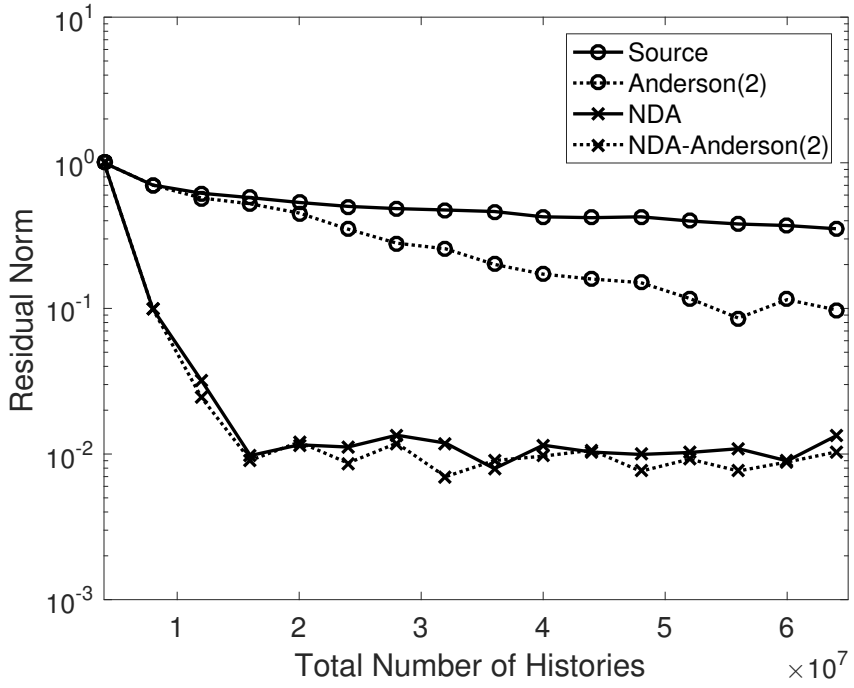


Figure 5.2: Residual histories of Profugus dependent fixed source calculations with $c = .99$.

both stagnate at the same residual.

Lastly, we observe negligible differences in iteration histories storing more than two Anderson vectors and only slight incremental improvement on intermediate iterates using Anderson(2) over Anderson(1). We also observe that Anderson accelerated versions are more stable once stagnation is reached.

5.2 CADIS and FW-CADIS

Consistent adjoint driven importance sampling (CADIS) and *forward-weighted* CADIS(FW-CADIS) are hybrid variance reduction techniques from [57–59, 61]. They provides an optimized single region tally and distribution, respectively. FW-CADIS may also optimize multiple tally regions at once. The hybrid methods CADIS and FW-CADIS calculate a coarse 3D discrete ordinates solution to the *adjoint transport equation*. The adjoint solution informs the Monte Carlo simulation through appropriate *weight windows*. Weight windows decrease weights of particles traveling away from target tally region causing flights to terminate quickly. Particles above the weight window threshold undergo splitting. This reduces variance at the locations of interest by having a greater chance that particles remain alive and travel through the tally regions.

5.2.1 The Adjoint Transport Equation

The solution of *adjoint* transport equation is another tool that may assist with certain classes of calculations, such as CADIS variance reduction. We follow the formulation in [37]. Consider the operator $\mathbf{H}$ and pair of function ψ and ψ^+ that meet appropriate boundary conditions. Assume $\mathbf{H}$ has vacuum boundary conditions. The adjoint operator $\mathbf{H}^+$ is defined by

$$\int d^3\vec{r} \int d\hat{\Omega} \int dE\ \psi^+ \mathbf{H}\psi = \int d^3\vec{r} \int d\hat{\Omega} \int dE\ \psi \mathbf{H}^+\psi^+. \tag{5.4}$$

In nonmultiplying transport,

$$\mathbf{H}\psi = \left(\hat{\Omega}\cdot\vec{\nabla} + \Sigma_t(\vec{r},E)\right)\psi(\vec{r},\hat{\Omega},E) - \int dE' \int d\hat{\Omega}'\ \Sigma_s(\vec{r},E'\to E,\hat{\Omega}'\cdot\hat{\Omega})\psi(\vec{r},\hat{\Omega}',E'). \tag{5.5}$$

To find $\mathbf{H}^+$, we reverse positions of ψ and ψ^+ from the known lefthand side in (5.4). We begin

$$\int d^3\vec{r} \int d\hat{\Omega} \int dE\ \psi^+ \mathbf{H}\psi = \int dV \int dE \int d\hat{\Omega}\ \psi^+(\vec{r},\hat{\Omega},E) \times \left[\left(\hat{\Omega}\cdot\vec{\nabla} + \Sigma_t(\vec{r},E)\right)\psi(\vec{r},\hat{\Omega},E) - \int dE' \int d\hat{\Omega}'\ \Sigma_s(\vec{r},E'\to E,\hat{\Omega}'\cdot\hat{\Omega})\psi(\vec{r},\hat{\Omega}',E')\right]. \tag{5.6}$$

First, the two components of streaming operator are commutative, so $\hat{\Omega}\cdot\vec{\nabla} = \vec{\nabla}\cdot\hat{\Omega}$. Apply this operator to $\psi^+\psi$ to obtain

$$\vec{\nabla}\cdot(\hat{\Omega}\psi^+\psi) = \psi^+\hat{\Omega}\cdot\vec{\nabla}\psi + \psi\hat{\Omega}\cdot\vec{\nabla}\psi^+. \tag{5.7}$$

Integrate the result over V to obtain

$$\int dV\ \vec{\nabla}\cdot(\hat{\Omega}\psi^+\psi) = \int dV\ \psi^+\hat{\Omega}\cdot\vec{\nabla}\psi + \int dV\ \psi\hat{\Omega}\cdot\vec{\nabla}\psi^+ \tag{5.8}$$

The divergence theorem reduces the lefthand side to the surface integral

$$\int dS\ \vec{n}\cdot\hat{\Omega}\psi^+\psi = \int dV\ \psi^+\hat{\Omega}\cdot\vec{\nabla}\psi + \int dV\ \psi\hat{\Omega}\cdot\vec{\nabla}\psi^+ \tag{5.9}$$

The vacuum boundary condition on $\mathbf{H}$ states that for $\hat{\Omega}\cdot n < 0$, $\psi \equiv 0$. Therefore, we enforce ψ^+ to satisfy

$$\psi^+(\vec{r},\hat{\Omega},E) = 0, \quad \vec{r}\in S,\ \hat{\Omega}\cdot\vec{n} \geq 0. \tag{5.10}$$

Thus, the surface integral is zero, and

$$\int dV\ \psi^+\ \hat{\Omega}\cdot\vec{\nabla}\psi = -\int dV\ \psi\ \hat{\Omega}\cdot\vec{\nabla}\psi^+. \tag{5.11}$$

Next, in the collision operator, Σ_t is a coefficient, so we reverse $\psi^+\Sigma_t\psi$ to $\psi\Sigma_t\psi^+$. Lastly, in the scattering operator, we reverse the dummy variables E, E' and $\hat{\Omega}, \hat{\Omega}'$ to see

$$\int d\hat{\Omega} \int dE\, \psi^+(\vec{r}, \hat{\Omega}, E) \int d\hat{\Omega}' \int dE'\, \Sigma_s(\vec{r}, E' \to E, \hat{\Omega}' \cdot \hat{\Omega})\psi(\vec{r}, \hat{\Omega}', E') = \int d\hat{\Omega} \int dE\, \psi(\vec{r}, \hat{\Omega}, E) \int d\hat{\Omega}' \int dE'\, \Sigma_s(\vec{r}, E \to E', \hat{\Omega} \cdot \hat{\Omega}')\psi(\vec{r}, \hat{\Omega}', E'). \quad (5.12)$$

The three equalities lead to the expression

$$\int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi^+ \mathbf{H}\psi = \int dV \int dE \int d\hat{\Omega}\, \psi(\vec{r}, \hat{\Omega}, E) \times \left[\left(-\hat{\Omega} \cdot \vec{\nabla} + \Sigma_t(\vec{r}, E)\right)\psi^+(\vec{r}, \hat{\Omega}, E) - \int dE' \int d\hat{\Omega}'\, \Sigma_s(\vec{r}, E \to E', \hat{\Omega} \cdot \hat{\Omega}')\psi^+(\vec{r}, \hat{\Omega}', E')\right], \quad (5.13)$$

and so the adjoint transport operator is

$$\mathbf{H}^+\psi^+ = \left(-\hat{\Omega} \cdot \vec{\nabla} + \Sigma_t(\vec{r}, E)\right)\psi^+(\vec{r}, \hat{\Omega}, E) - \int dE' \int d\hat{\Omega}'\, \Sigma_s(\vec{r}, E \to E', \hat{\Omega} \cdot \hat{\Omega}')\, \psi^+(\vec{r}, \hat{\Omega}', E'). \quad (5.14)$$

5.2.2 Detector Response Example

Given an external source q, we illustrate how to use the adjoint in solving the transport problem

$$\mathbf{H}\psi = q, \quad (5.15)$$

with vacuum boundary conditions on ψ, found in [37]. Specifically, assume we desire the detector response R over a specific region d with volume V_d. The response is represented by the integrated reaction rate over d,

$$R = \int_{V_d} d^3r \int_0^\infty dE\, \hat{\Sigma}_t(\vec{r}, E)\phi(\vec{r}, E). \quad (5.16)$$

Let

$$\hat{\Sigma}_d(\vec{r}, E) = \begin{cases} \Sigma_t(\vec{r}, E) & \vec{r} \in d \\ 0 & \vec{r} \notin d \end{cases} \quad (5.17)$$

be the characteristic function defined on d weighted by the total cross section of the detector. We define the corresponding adjoint problem as

$$\mathbf{H}^+\psi^+ = \hat{\Sigma}_d(\vec{r}, E), \quad (5.18)$$

where ψ^+ satisfies (5.10). Multiply both sides of (5.18) by ψ^+ to obtain

$$\psi^+\mathbf{H}\psi = \psi^+ q \quad (5.19)$$

and multiply both sides of (5.18) by ψ to obtain

$$\psi \mathbf{H}^{+} \psi^{+} = \psi \hat{\Sigma}_d(\vec{r}, E). \tag{5.20}$$

We integrate over $\vec{r}^3, \hat{\Omega}$, and E then use the identity (5.4) to see

$$\int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi^{+} \mathbf{H} \psi - \int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi \mathbf{H}^{+} \psi^{+} = \int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi^{+} q - \int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi \hat{\Sigma}_d(\vec{r}, E) \tag{5.21}$$

$$0 = \int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi^{+} q - R \tag{5.22}$$

Therefore, we obtain

$$R = \int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi^{+} q, \tag{5.23}$$

an integral of the adjoint weighted external source. To better understand the physical meaning of ψ^{+}, we define an external source emitting particles at a rate of one per second at location $\vec{r}_0$ in direction $\hat{\Omega}_0$ with energy E_0 as

$$q(\vec{r}, \hat{\Omega}, E) = \delta(\vec{r} - \vec{r}_0)\, \delta(\hat{\Omega} - \hat{\Omega}_0)\, \delta(E - E_0). \tag{5.24}$$

Substituting (5.24) into (5.23), we see

$$R = \int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi^{+}(\vec{r}, \hat{\Omega}, E)\, \delta(\vec{r} - \vec{r}_0)\, \delta(\hat{\Omega} \cdot \hat{\Omega}_0)\, \delta(E - E_0) \tag{5.25}$$

$$= \psi^{+}(\vec{r}_0, \hat{\Omega}_0, E_0). \tag{5.26}$$

Therefore, the expected contribution of a particle in $(\vec{r}_0, \hat{\Omega}_0, E_0)$ to the response R is ψ^{+} at $(\vec{r}_0, \hat{\Omega}_0, E_0)$. We view ψ^{+} as the importance of particles produced from q relative to R. This importance metric is critical in the CADIS and FW-CADIS variance reduction techniques.

We solve (5.18) with methods using discrete ordinates found in Section 2.3.1. Two methods used in production level codes are the *linear-discontinuous galerkin* finite element method [48,63], and the *step characteristics* method [35].

We use codes developed at Oak Ridge National Laboratory for results found in Chapter 6. The Denovo code [16] calculates the adjoint solution and the Shift code [46] performs the Monte Carlo simulation. Both codes are in Exnihilo of the Scale code suite [1].

5.2.3 Consistent Adjoint Driven Importance Sampling and Weight Windows

Considering the detector response problem, we rewrite the total response with the adjoint scalar flux as

$$R = \int dE \int d^3\vec{r}\, \phi^+(\vec{r}, E)\, q(\vec{r}, E). \tag{5.27}$$

From (5.4), we see

$$\int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi^+ \mathbf{H} \psi = \int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi \mathbf{H}^+ \psi^+ \tag{5.28}$$

$$\int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi^+ q = \int d^3\vec{r} \int d\hat{\Omega} \int dE\, \psi q^+, \tag{5.29}$$

with adjoint problem

$$\mathbf{H}^+ \psi^+ = q^+, \tag{5.30}$$

where q^+ is the adjoint source. Therefore, we also have

$$R = \int dE \int d^3\vec{r}\, \phi(\vec{r}, E)\, q^+(\vec{r}, E). \tag{5.31}$$

We let $q^+ = \sigma_d$, an objective function such as a *dose rate response* function. A dose rate response function is simply a flux-to-dose rate conversion factor. In (5.18), we have the reaction rate objective function $\sigma_d(\vec{r}, E) = \hat{\Sigma}_d(\vec{r}, E)$.

We solve (5.30) given q^+ and integrate to obtain ϕ^+. Considering ϕ^+ as a measure of particle importance to R, we bias the source relative to ϕ^+ and normalize. The biased source distribution is

$$\hat{q} = \frac{\phi^+(\vec{r}, E)\, q(\vec{r}, E)}{\int dE \int d^3\vec{r}\, \phi^+(\vec{r}, E)\, q(\vec{r}, E)} = \frac{\phi^+(\vec{r}, E)\, q(\vec{r}, E)}{R}. \tag{5.32}$$

If we sample particles from $\hat{q}$, samples are proportional to expected contributions to R. We also use ϕ^+ and $\hat{q}$ to form the *weight windows* for the Monte Carlo simulation.

A weight window has an upper and lower bound, $\hat{w}_u$ and $\hat{w}_\ell$. The lower bound is

$$\hat{w}_\ell(\vec{r}, E) = \frac{R}{\phi^+(\vec{r}, E)\left(\frac{c+1}{2}\right)}, \tag{5.33}$$

where c is the width of the window. It is a user parameter and defines the ratio

$$c = \frac{\hat{w}_u}{\hat{w}_\ell}. \tag{5.34}$$

Therefore, the upper bound is

$$\hat{w}_u = c\hat{w}_\ell. \tag{5.35}$$

If a particle's weight rises above $\hat{w}_u$, for example, when moving in the direction of the detector, the particle is split. If the particle's weight falls below $\hat{w}_\ell$, when moving away from the detector, the particle is rouletted. Each of these actions either brings the particle back into the allowable weight range or removes the particle from the system.

Finally, the biased source and weight windows are consistently defined, so initial weights for source particles must lie within the windows. This avoids immediate splitting and rouletting at birth. The weights $\hat{w}$ satisfy

$$\hat{w}_0(\vec{r}, E)\ \hat{q}(\vec{r}, E) = w_0(\vec{r}, E)\ q(\vec{r}, E) \tag{5.36}$$

where $w_0 \equiv 1$ in the unbiased problem. Therefore, the initial weights are

$$\hat{w}_0(\vec{r}, E) = \frac{q(\vec{r}, E)}{\hat{q}(\vec{r}, E)} = \frac{R}{\phi^+(\vec{r}, E)}. \tag{5.37}$$

5.2.4 Forward-Weighted CADIS

Forward-weighted CADIS has the same goal as CADIS but instead optimizes multiple detector regions or distributions over large parts of the problem space [59, 61]. The recommendation in [11] is that Monte Carlo particles be uniformly distributed throughout the domain. With this objective, we find an appropriate choice of q^+ to use in (5.31).

The physical neutron density $n(\vec{r}, \hat{\Omega}, E)$ in the system is proportional to the Monte Carlo particle density $m(\vec{r}, \hat{\Omega}, E)$ scaled by their average particle weights $\bar{w}(\vec{r}, \hat{\Omega}, E)$, so

$$n(\vec{r}, \hat{\Omega}, E) \propto \bar{w}(\vec{r}, \hat{\Omega}, E)\ m(\vec{r}, \hat{\Omega}, E). \tag{5.38}$$

We also know

$$\psi(\vec{r}, \hat{\Omega}, E) = n(\vec{r}, \hat{\Omega}, E)\ v(\vec{r}, \hat{\Omega}, E), \tag{5.39}$$

where v is the particle velocity. Substituting (5.38) into (5.39) and solving for m, we obtain

$$m(\vec{r}, \hat{\Omega}, E) = \psi(\vec{r}, \hat{\Omega}, E)\left[\frac{1}{\bar{w}(\vec{r}, \hat{\Omega}, E)\ v(\vec{r}, \hat{\Omega}, E)}\right]. \tag{5.40}$$

Setting m constant in (5.38), we see

$$n(\vec{r}, \hat{\Omega}, E) \propto \bar{w}(\vec{r}, \hat{\Omega}, E) \tag{5.41}$$

$$\psi(\vec{r}, \hat{\Omega}, E) \propto \bar{w}(\vec{r}, \hat{\Omega}, E)\ v(\vec{r}, \hat{\Omega}, E). \tag{5.42}$$

Integrate (5.40) over $\vec{r}^3, \hat{\Omega}$, and E to obtain

$$R' = \int d^3\vec{r} \int d\hat{\Omega} \int dE\ \psi(\vec{r}, \hat{\Omega}, E) \left[\frac{1}{\psi(\vec{r}, \hat{\Omega}, E)} \right]. \tag{5.43}$$

We notice that if we define

$$q^+(\vec{r}, \hat{\Omega}, E) = \frac{1}{\psi(\vec{r}, \hat{\Omega}, E)}, \tag{5.44}$$

then (5.31) becomes (5.43). We achieve uniform Monte Carlo particle density by weighting q^+ proportional to $1/\psi$.

In FW-CADIS, we weight the q^+ response function by $1/\phi$, where ϕ is a coarse forward scalar flux solution of (5.18). With the weighted q^+, we solve (5.30) for a forward weighted importance metric ϕ^+. Now, when forward flux is low, the adjoint importance is high and vice-versa. We use ϕ^+ to create a biased source and weight weights in CADIS for the Monte Carlo calculation.

As an example from [61], let $\sigma_d(\vec{r}, E)$ be the dose rate response function over a mesh tally. The total dose rate is

$$D(\vec{r}) = \int dE\ \phi(\vec{r}, E)\ \sigma_d(\vec{r}, E), \tag{5.45}$$

a spatially dependent quantity. The adjoint source is

$$q^+(\vec{r}, E) = \frac{\sigma_d(\vec{r}, E)}{\displaystyle\int dE\ \phi(\vec{r}, E)\ \sigma_d(\vec{r}, E)}. \tag{5.46}$$

And, if the goal is to obtain responses on a restricted volume V_d, we redefine (5.46) as

$$q^+(\vec{r}, E) = \begin{cases} \dfrac{\sigma_d(\vec{r}, E)}{\displaystyle\int dE\ \phi(\vec{r}, E)\ \sigma_d(\vec{r}, E)}, & \vec{r} \in V_d \\ 0, & \vec{r} \notin V_d. \end{cases} \tag{5.47}$$

Chapter 6

Hybrid and Parallel Domain Decomposed Monte Carlo

In this chapter, we discuss the motivation for domain decomposition Monte Carlo(DDMC) and diagnosing the load imbalance problem that arises. We propose a nonuniform processor allocation strategy to alleviate the load imbalance between subdomains and we introduce a diagnostic-based approach to optimize the allocations. The strategies are then applied to three test cases, including the Watts Bar Nuclear 1(WBN1) initial start up reactor. All relevant contributions to the Shift code base are found in Appendix A.3. All features are currently available to users in the production line code.

6.1 Parallel Computing

A parallel computation is a computation performed over concurrently running processor *cores*.

Definition 6.1.1. *A* **core** *is a single processing element of a central processing unit (CPU) that carries out program instructions.*

Each core runs its set of instructions on local data that is stored on the *node*.

Definition 6.1.2. *A* **node** *is a self contained compute unit of locally connected cores. Each node contains cores, memory, storage, and an input/output communication system.*

Node often exist within a computer cluster of many nodes, such as Titan. When a submitting parallel job to the job scheduler, users must select the number of processor cores per node to use for the calculation (up to 16 cores for Titan). Reducing the cores per node may reduce memory usage if users are facing memory constraints, but all other processors on the node remain idle during the calculation.

When necessary, data communication between cores or nodes is performed using network connections and message passing commands. Message Passing Interface(MPI) [38] is used for distributed

memory programs, required in node-to-node communication, and OpenMP [45] is used for shared memory programs, required in on-node core-to-core communication. We use the two messaging systems together when computing on a cluster of many nodes each with several cores. To send a message it is required to know the cores' *ranks* and the message size.

Definition 6.1.3. *A* **rank** *is the core's index in the fixed ordered set of cores.*

If we, for example, need to send a particle from one core to another, we use the ranks of both cores and the datasize of the particle. Furthermore, to complete a parallel computation we consolidate through messages the concurrent calculations to produce the final result.

Shift is one of the few radiation transport codes that supports parallel execution using both *domain replication* and *domain decomposition* [60]. In domain replication, multiple cores each maintain a full copy of the problem domain and perform independent simulations of particle histories and tally the corresponding responses. At the end of the simulation, the results from each core are consolidated to produce a final estimation of the desired responses. Because the work performed by each core is entirely independent, domain replication achieves a very high *parallel efficiency*, *weak scaling* at 99% efficiency up to 300k cores [46]. A calculation's parallel efficiency is

$$Eff_{parallel} = \frac{1}{N_{procs}} \left(\frac{T_{serial}}{T_{parallel}} \right), \tag{6.1}$$

and measures a parallel computation's resource utilization.

To understand an algorithms scalability, we obtain a series of measurements in parallel efficiency forming a scaling study, either weak or *strong*.

Definition 6.1.4. Weak scaling *measures the variation in solution time as the number of processors changes for a fixed problem size per processor*

Definition 6.1.5. Strong scaling *measures the variation in solution time as the number of processors changes for a fixed problem size.*

For example, a strong scaling study runs a series of $100k$ particle simulations with increasing processor counts. In contrast, a weak scaling study will scale the number of particles with the processor count (e.g. $100k$ particles for 1 processor and $400k$ particles for 4 processors).

6.2 Data Decomposition

However, for some problems it may not be feasible to fully replicate the problem domain due to the memory costs associated with storing the desired quantities on node. In this case, it is necessary to spatially decompose the problem such that each processor only has knowledge of a subset of the full

problem domain. If a particle history reaches the edge of a processor's domain that does not correspond to a global domain boundary, it must communicate the particle to a neighboring processor that owns the information needed to continue transporting the particle. Because communication is required between processors during this process, the parallel efficiency for domain-decomposed Monte Carlo is typically much lower than for the domain-replicated case. Shift uses a combination of replication and decomposition known as *multiple set, overlapping domain* (MSOD) [46]. An example of MSOD is seen in Figure 6.1.

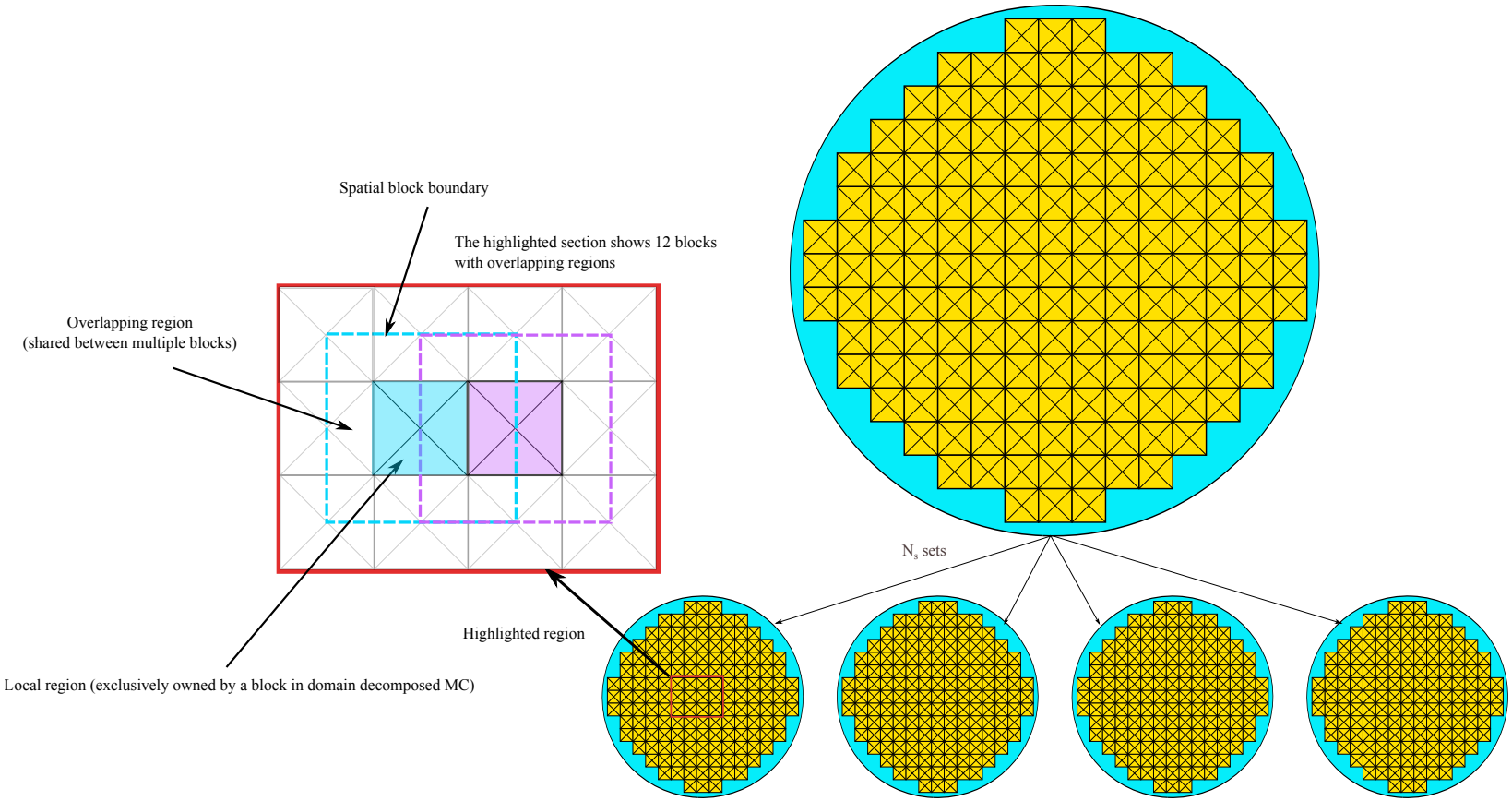


Figure 6.1: Example of MSOD geometry entirely replicated over 4 sets.

In this approach, the global problem domain is spatially decomposed into (possibly overlapping) domains. A collection of domains that spans the global problem domain is known as a *set*.

Definition 6.2.1. *A* **set** *is an instance of the fully replicated geometry that simulates a batch of particles.*

A set may be replicated as many times as is desired to utilize additional processors. For example, an N_p particle Monte Carlo simulation may be split into N_b *batches* each with N_p/N_b particles. If we assume perfect parallel scaling, then time to completion reduces by the factor N_b. We track different batches and processors transporting particles through set and domain ids. For domain replication, set and domain ids are identical.

Definition 6.2.2. *A* **domain id** *is the global rank of a processor transporting particles over a portion of the geometry.*

Because domain decomposition is less efficient than domain replication, the problem is spatially decomposed into the fewest domains that allows the problem to satisfy per-node memory limitations and then replicated as many times as possible with available computing resources.

6.3 Synchronous Domain Decomposed Monte Carlo

In *domain decomposed* Monte Carlo [26, 60], we divide the problem data by its spatial variable into N_d uniform subdomains, or *blocks*.

Definition 6.3.1. *A* **block** *is a spatial subdomain in the decomposition.*

Particles that lived on the undecomposed geometry, now live on blocks with shared boundaries. Each block is associated with distinct processors. Particles that cross block boundaries are added to a communication buffer between processors. We must decide when to communicate buffered particles to the new subdomain and continue with their flights.

We consider a synchronous domain decomposed scheme as opposed to an asynchronous scheme [7]. The synchronous domain decomposed Monte Carlo simulates a *generation* of particles between sync points.

Definition 6.3.2. *A* **generation** *is a number of particles transported from either sources or communication buffers.*

We also use the term *iterations* per generation. The algorithm for the synchronous DDMC implementation is seen in Algorithm 8, where N_{ppg} is the number of particles per generation.

Algorithm 8 Synchronous DDMC algorithm

`synchronous_ddmc`(N_p, N_d, N_{ppg})

```
Decompose geometry into N_d subdomains
while !complete do
  for i = 1 : N_ppg do
    if !bank.empty() then
      Transport bank particle to termination or into outgoing buffer
    else if !source.empty() then
      Transport source particle to termination or into outgoing buffer
    end if
  end for
  Communicate particle buffers and dump into existing banks
  Check for complete
  Update generation counter
end while
Global reduction of tally results across sets
```

The blocks are flagged as finished when their generation is completed or they run out of particles to transport. Processors are halted to transfer particles across boundaries and update fission sources. The simulation continues with the next generation. We complete generation after generation until we reach N_p samples from the source distribution and all buffers are empty. We store tally data as the simulation runs.

Our motivation for choosing the synchronous scheme is to avoid *race conditions* that may occur with particle splitting [6]. Race conditions are undesirable situations that occur when two elements of a system experience an ordering or timing problem leading to unexpected program behavior. This arises during asynchronous parallel operations. Syncing the processors invaluably prevents communication and halting faults but slows the overall runtime of the computation.

6.4 Load Imbalance

By definition, load imbalance is an unequal distribution of work across processors. Idle processors do not contribute and therefore do not reduce time to completion. In fixed-source Monte Carlo calculations with isolated sources, for example, a group of the processors may not be transporting particles. The majority of transported particles for fixed-source calculations are in subdomains near sources. Statistically few particles will travel to subdomains far away from sources. The processors allocated to these subdomains will sit idle after their small workload is completed. In hybrid Monte Carlo calculations, the particle

storms that form due to particle splitting create massive burdens on subdomains near target tally regions. If the load is not balanced, parallel efficiency greatly reduces.

The magnitude of this effect also depends on leakage fractions and the fineness of the decomposition [50]. Leakage between blocks reduces load imbalance by providing work to idle processors. Whereas, a finer decomposition increases the number of processors without useful work when under load imbalance. The objective is to reduce the load imbalance effect.

6.4.1 Domain Decomposition Numerical Results

The first case we consider investigates the cost of communication from the synchronous scheme without load imbalance. We consider a 12x12 assembly grid with each assembly containing a 17x17 grid of UO_2 fuel pins, seen in Figure 6.2. The design specifications are seen in Table 6.1. We calculated parallel efficiencies for decompositions with processor counts of $\{1, 8, 64, 216\}$ and constant generations of 1,000 particles. All calculations were performed on the *Titan Cray XK7* [42] at Oak Ridge National Laboratory.

Table 6.1: Fuel assembly grid specifications

Parameter	Dimensions	N_p	Source	Spectrum	Boundary Conditions
Value	$257 \times 257 \times 100cm$	1e6	Uniform	Watt	Vacuum

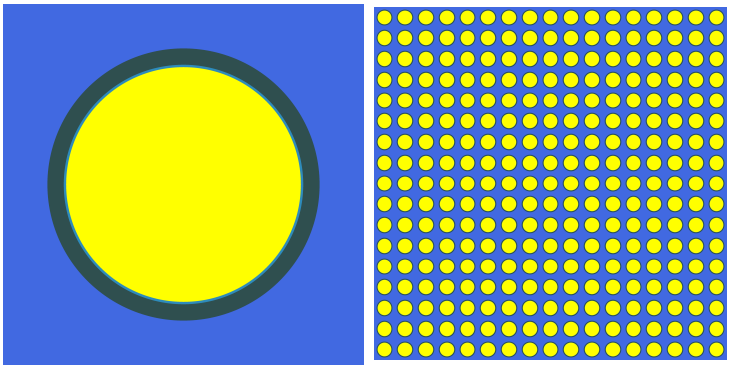


Figure 6.2: UO_2 fuel pin and assembly.

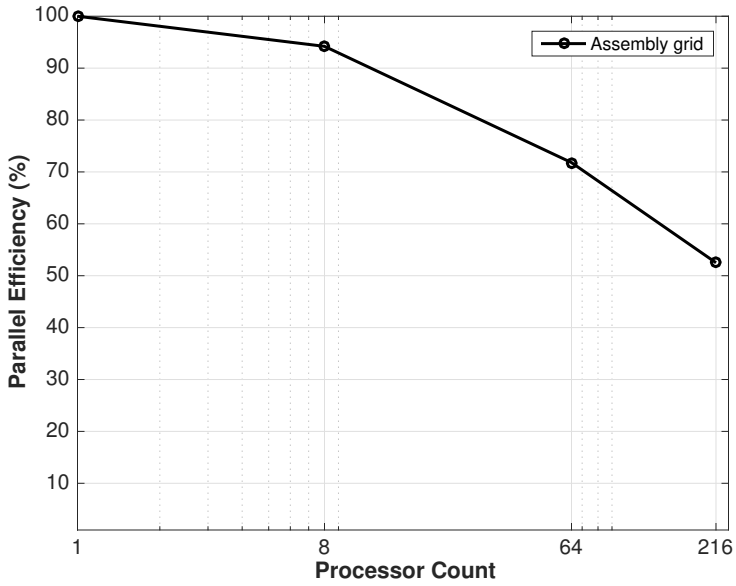


Figure 6.3: Parallel efficiencies for 12x12 UO_2 assembly grid with uniform source distribution.

From Figure 6.3, we see that parallel efficiency is deteriorating as processor count increases, with 216 processors barely reaching 50% efficiency. We aim to maintain efficiency in the 8 to 64 processor regime. This decomposition regime allows us to fit the decomposed problem data onto nodes.

The second case we consider investigates the cost of communication and load imbalance. We consider a cube of uniform water moderator of different widths. We artificially create load imbalance by placing a box source, having width equal to 20% of the total domain width, in either the domain center or a domain corner. We calculated the average *mean free path* of particles at energies 1e6–1e7 eVs in water to be 7.85cm using the Nemesis package in Scale [1].

Definition 6.4.1. *A* **mean free path** *is the material and energy dependent quantity that describes the average distance between collisions.*

The design specifications are seen in Table 6.2. We calculated parallel efficiencies for decompositions with processor counts of $\{1, 8, 64\}$ and constant generations of 1e3 particles.

Table 6.2: Uniform water moderator specifications

Test Case	Widths	N_p	Source	Spectrum	Boundary Conditions
1	$\{2, 4, 11\}$ mfps	1e6	Uniform	Uniform 1e6–1e7 eVs	Vacuum
2	$\{2, 4, 11\}$ mfps	1e6	Centered	Uniform 1e6–1e7 eVs	Vacuum
3	$\{2, 4, 11\}$ mfps	1e6	Cornered	Uniform 1e6–1e7 eVs	Vacuum
4	$\{2, 4, 11\}$ mfps	1e6	Cornered	Uniform 1e6–1e7 eVs	Reflective
5	$\{2, 4, 11\}$ mfps	1e6	Cornered	Uniform 1e6–1e7 eVs	Half-Reflective

In Table 6.2, *half-reflective* boundaries have reflective boundaries on the three faces nearest the source box and vacuum boundaries on the three faces farthest away.

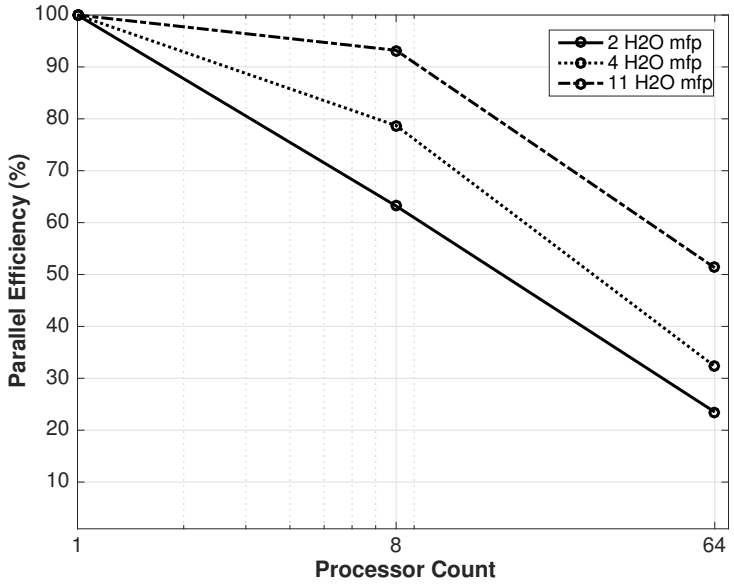


Figure 6.4: Parallel efficiencies for uniform water moderator with uniform source distribution.

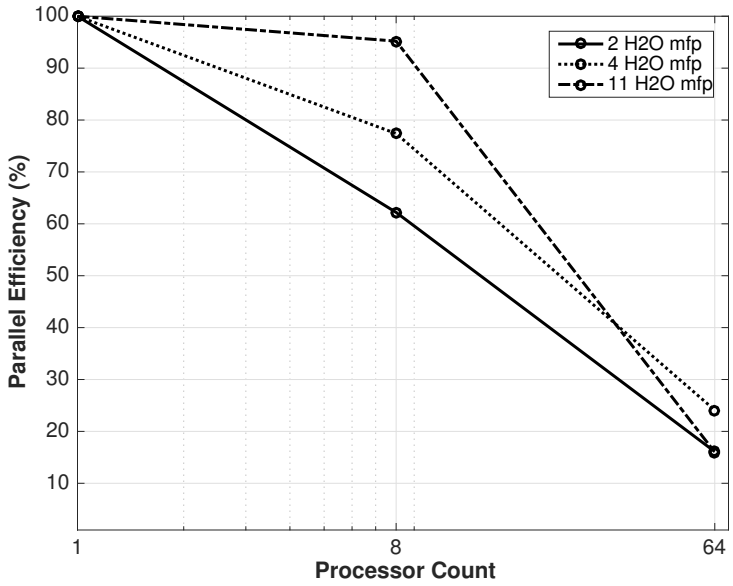


Figure 6.5: Parallel efficiencies for uniform water moderator with a centered source distribution.

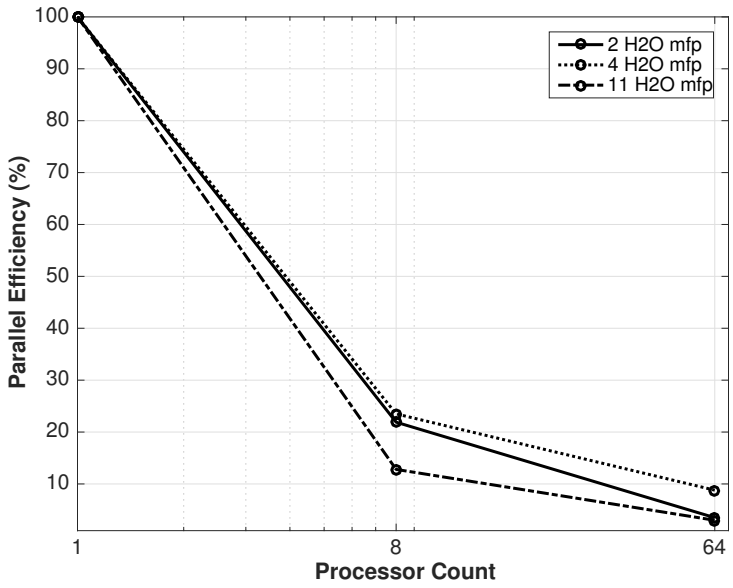


Figure 6.6: Parallel efficiencies for uniform water moderator with a cornered source distribution.

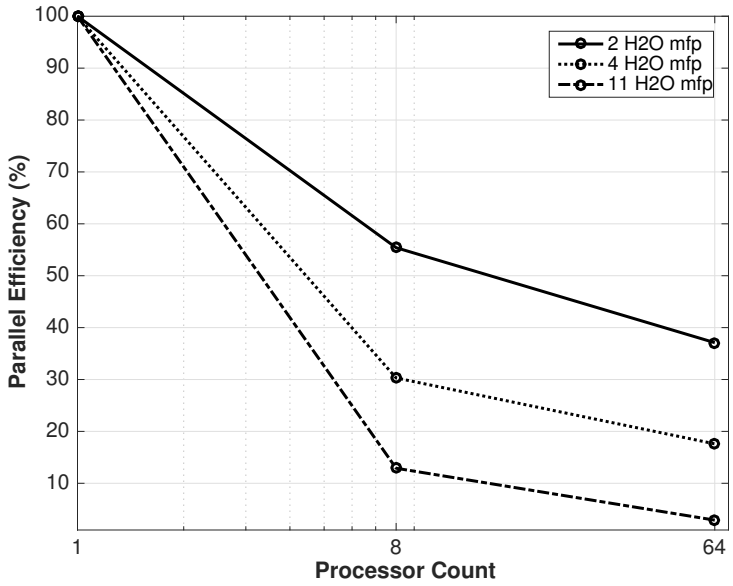


Figure 6.7: Parallel efficiencies for uniform water moderator with a cornered source distribution and full reflective boundaries.

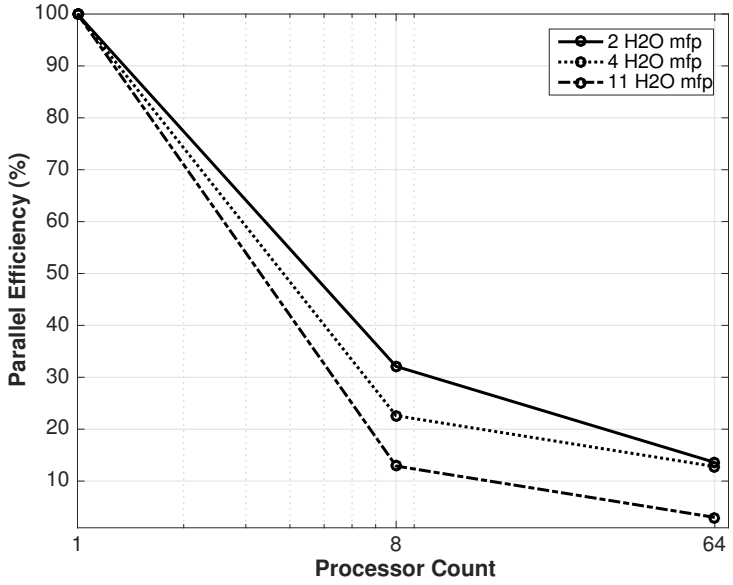
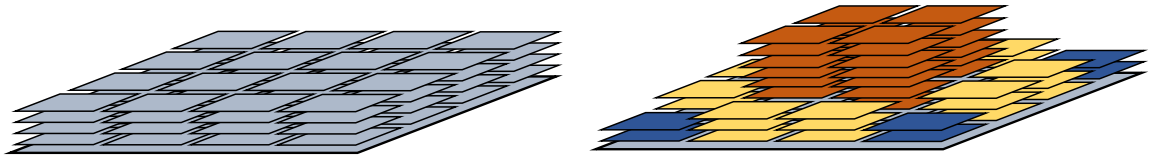


Figure 6.8: Parallel efficiencies for uniform water moderator with uniform source distribution and half reflective boundaries.

We see that efficiencies in Figure 6.4 are heavily affected by the width of the geometry. As the geometry size increases the parallel efficiency increases. The curve for 11 mfps, which is the approximate width of the assembly problem, closely resembles the curve in Figure 6.3. In Figure 6.5, the centered source calculations drastically lose efficiency when the decomposition increases to 64 processors. The cornered source test cases each have poor efficiencies for all decompositions. In Figure 6.6, no case reaches above 25% with vacuum boundaries. In Figure 6.7 and Figure 6.8, we observe that as reflective boundaries are added the efficiencies rise. This indicates that total domain leakage worsens the load imbalance effect. We also notice that as the width increases the efficiency decreases, in contrast to Figure 6.4. Therefore, we seek strategies to reduce this effect on parallel efficiency. In particular, we investigate assigning processors nonuniformly to subdomains.

6.5 Nonuniform Processor Allocation for Domain Decomposition

We extend the parallel decomposition strategy in Shift to allow a more flexible assignment of processors to Shift subdomains [14]. We show that the proposed scheme has the potential to improve Shift's parallel efficiency significantly. Current capabilities in Shift only allow for a uniform allocation of processors for each subdomain, see Figure 6.9a. We have a two dimensional domain decomposed into sixteen subdomains. In each of the subdomains, we have four processors transporting particles represented by the four stacked rectangles. If we allocate processors relative to their load, we reduce the load imbalance penalties from domain decomposition. With *nonuniform processor allocation*, we judiciously allocate processors, see Figure 6.9b. In this example, the source is strongest in the center subdomain, so we reassign processors from outer domains to alleviate the load imbalance.



(a) 64 processors uniformly over a 4×4 decomposition.

(b) 64 processors nonuniformly over a 4×4 decomposition.

Figure 6.9: Uniform vs. nonuniform allocations.

One critical limiting factor for the domain decomposed calculation is the subdomain with the greatest work to processor allocation ratio. In Shift, we have implemented a synchronous domain decomposed scheme as opposed to an asynchronous scheme [7]. The synchronous scheme simulates a fixed number of particles, called a *generation*, then all processors halt to communicate particles to neighboring

subdomains. The algorithm's synchronicity requires that all domains transport and communicate their particles before continuing to the next generation. One poorly balanced subdomain leads to parallel inefficiencies for the whole calculation.

Further, we have developed and implemented in Shift a new communication pattern for nonuniform processor allocation, see Figure 6.10. It allows any processor to communicate particles with a subset of its neighboring processors through particle buffers. Particle buffers are sized appropriately based on the constant generation size. We ensure load balance across processors in a subdomain by communicating a specific fraction of buffered particles to neighboring processors. The fractions are based on processors in the current subdomain and processors in the neighboring subdomain.

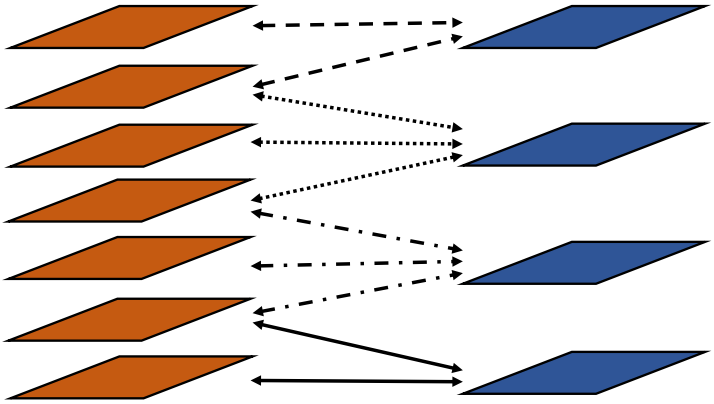


Figure 6.10: Subdomain 1 with 7 processors communicating with neighboring subdomain 2 with 4 processors.

For example, consider the processor allocations in Figure 6.10. Assume we have 7000 particles on subdomain 1 evenly divided amongst the 7 processors and all 7000 particles are to be communicated to subdomain 2. We ensure that after communication each processor on subdomain 2 has 1750 particles. To accomplish this, processor 1 from subdomain 1 transfers 100% of its buffered particles to processor 1 of subdomain 2. Processor 2 from subdomain 1 transfers 75% of its buffered particles to processor 1 of subdomain 2 and 25% to processor 2 of subdomain 2. Processor 3 from subdomain 1 transfers 100% of its buffered particles to processor 2 of subdomain 2. Processor 4 from subdomain 1 transfers 50% of its buffered particles to processor 2 of subdomain 2 and 50% to processor 3 of subdomain 2, and so on for all processors. This strategy reduces size and number of messages from neighbor to neighbor, thereby keeping communication costs small.

6.5.1 Optimization of Nonuniform Processor Allocation

Nonuniform processor allocation increases the degrees of freedom that must be optimized. Two approaches are introduced to determine the distribution of processors: one based on an implicit filtering optimization algorithm and one based on timing diagnostics collected during the execution of a small initial calculation. It is shown that the approach based on timing diagnostics is simpler to implement and produces a more optimal final result.

We aim to find the optimal static processor allocation across subdomains in order to best fit the load and reduce overall runtime. To formulate the optimization problem, the design vector, $q \in \mathbb{N}^{N_d}$, is a vector of integer processors per subdomain with N_d subdomains. We minimize the transport run-time of a highly parallel domain-decomposed Monte Carlo simulation. We order the elements of q with a column-major cardinal index, see Figure 6.11. For example, in the 3x3x3 decomposition, logical index

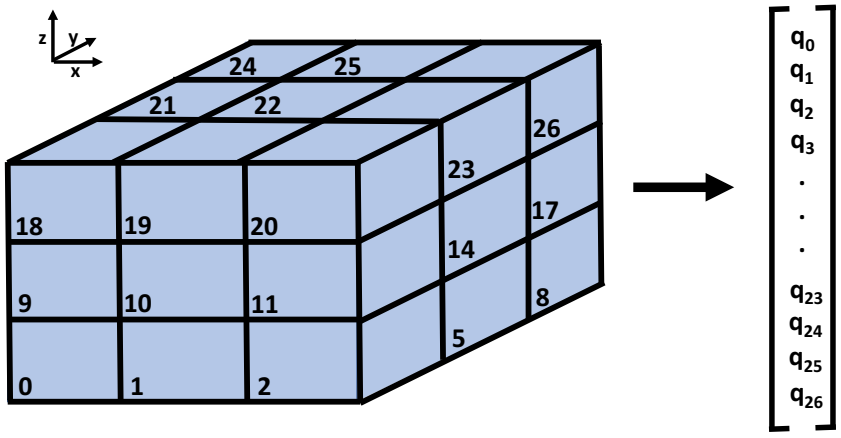


Figure 6.11: Cardinal ordering of subdomains for 3x3x3 decomposed domain.

$(2, 1, 0)$ maps to index 5 and $(1, 0, 2)$ maps to index 19.

We consider the discrete optimization problem

$$\min_{q \in \Omega} \; f(q), \tag{6.2}$$

where

$$\Omega = \{q \in \mathbb{N}^{N_d} \mid 1 \leq (q)_i \leq N_{procs} - (N_d - 1)\}, \tag{6.3}$$

the feasibility region and N_{procs} is the total number of processors to be allocated. We also require that

all processors are allocated creating the linear constraint

$$\sum_{i=1}^{N_d} (q)_i = N_{procs}. \tag{6.4}$$

Our first approach is to solve the minimization problem using *implicit filtering* [31]. Implicit filtering is a deterministic sampling algorithm for real-valued, bound constrained optimization. It is designed to minimize noisy, nonsmooth objective functions. It combines the advantages of pure sampling, like Nelder-Mead [41], and interpolatory methods [10, 47], which use first-order interpolation and a quasi-Newton model Hessian. Pure sampling explores the inherent noisy function surface well, and the quasi-Newton model Hessian exploits regions which are well approximated by smooth functions for accelerated convergence. Parallel timings are inherently noisy; therefore, pure gradient-based methods are ill-suited for this problem.

Second, we solve the minimization problem using run-time diagnostics collected during a low particle count calculation. The updated synchronous DDMC algorithm with diagnostic collection is seen in Algorithm 9.

Algorithm 9 Synchronous DDMC algorithm

`synchronous_ddmc`(N_p, N_d, N_{ppg})

```
Decompose geometry into N_d subdomains
while !complete do
  for i = 1 : N_ppg do
    if !bank.empty() then
      timer.start()
      Transport bank particle to termination or into outgoing buffer
      timer.stop()
    else if !source.empty() then
      timer.start()
      Transport source particle to termination or into outgoing buffer
      timer.stop()
    end if
  end for
  Communicate particle buffers and dump into existing banks
  Check for complete
  Update generation counter and diagnostics
end while
Global reduction of tally results across sets
```

After the calibration calculation, we have the total transport time, $t_{procs} \in \mathbb{R}_+^{N_procs}$, for each processor. We flatten t_{procs} to $t_d \in \mathbb{R}_+^{N_d}$ with $N_{procs} \geq N_d$. For example, if processors $0, 1, 2$ are allocated to subdomain 0, then $(t_d)_0$ is the cumulative processing time for processors $0, 1, 2$. These timings are then used to calculate a processor allocation using Algorithm 10 [44]. This algorithm is designed to reduce the maximum load-to-processor allocation ratio across all subdomains. We begin with one processor allocated to each subdomain and distribute each remaining processor sequentially to the subdomain with the current worst load.

Several metrics could be used to guide the processor allocation process. Total particles transported and sourced on each subdomain is a potential candidate. Alternatively, if communication is the bottleneck, communication diagnostics may be optimal. In this study, we target reducing overall transport time.

Algorithm 10 Processor Allocation

`allocate_procs`(t_d, N_{procs})

Set $q = (1, 1, \cdots, 1)^T \in \mathbb{N}^{N_d}$.

for $k \in \{1, \cdots, N_{procs} - N_d\}$ **do**

Find $i^* = \text{argmax}_{0 \le i \le N_d - 1} \ [(t_d)_i/(q)_i]$.

Increment $(q)_{i^*}$ by 1.

end for

Run-time diagnostics are an attractive alternative to iterative optimization methods because the diagnostic-based approach does not require multiple costly function evaluations. It only requires the single calibration step (often with very few calibration particles needed), then the calculation continues to the full particle MC run.

6.5.2 Nonuniform Processor Allocation Test Problems

The objective is to achieve the highest possible intra-set parallel efficiency by (1) spatially decomposing the problem into the smallest number of subdomains required and (2) optimizing the distribution of subdomains to improve load-balancing while reducing communication costs. Thus, we only consider single set cases in the results that follow because each set may be replicated multiple times to scale to an arbitrarily large number of nodes. Since the set-to-set scaling is nearly linear [23, 46], the efficiency achieved on a single set will be roughly equivalent to the scaling across thousands of sets, which is sufficient to utilize the largest multi-node machines that currently exist and are in development over the next decade.

We consider two test cases: a dogleg duct problem, comparable to [33], and a small modular reactor (SMR) vessel fluence problem. Both test cases are highly load imbalanced for both forward and hybrid calculations. All calculations were performed on Titan [42], a Cray XK7 supercomputer at Oak Ridge National Laboratory.

First, we consider the dogleg duct problem whose geometry is seen in Figure 6.12. The channel is filled with air, the walls are concrete, the source is uranium-238, and the detector is steel. The boundaries are all vacuum. The dogleg duct problem is primarily a hybrid calculation where we seek to optimize the tally region at the location of a detector. Load imbalance forms due to the isolated source and heavy particle splitting near the detector. Regions away from the source, detector, and channel transport far fewer particles. In the forward calculation, particles have low probabilities of fully traveling through the channel to the detector. The bulk of source particles remain on one half of the spatial domain, indicating strong load imbalance. Forward and adjoint fluxes are seen in Figure 6.13 and 6.14.

Second, we consider an SMR vessel fluence problem whose geometry is seen in Figure 6.15. The

SMR case is an axial slice containing a 7x7 assembly grid each with uranium-238 fuel pins. Outside of the assemblies we have a water moderator and the reactor is encased in a steel vessel. The top and bottom axial boundaries are reflecting and the others are vacuum. In the SMR forward calculation, a majority of particles remain in the core with few ever reaching the vessel creating load imbalance. In the hybrid ex-core calculation, the source is biased to produce a majority of particles near the core edges. Particles that travel to the center of the core are quickly terminated. In both cases, the load imbalance is significant enough to warrant nonuniform processor allocation when running domain decomposition. Forward and adjoint fluxes are seen in Figure 6.16 and 6.17.

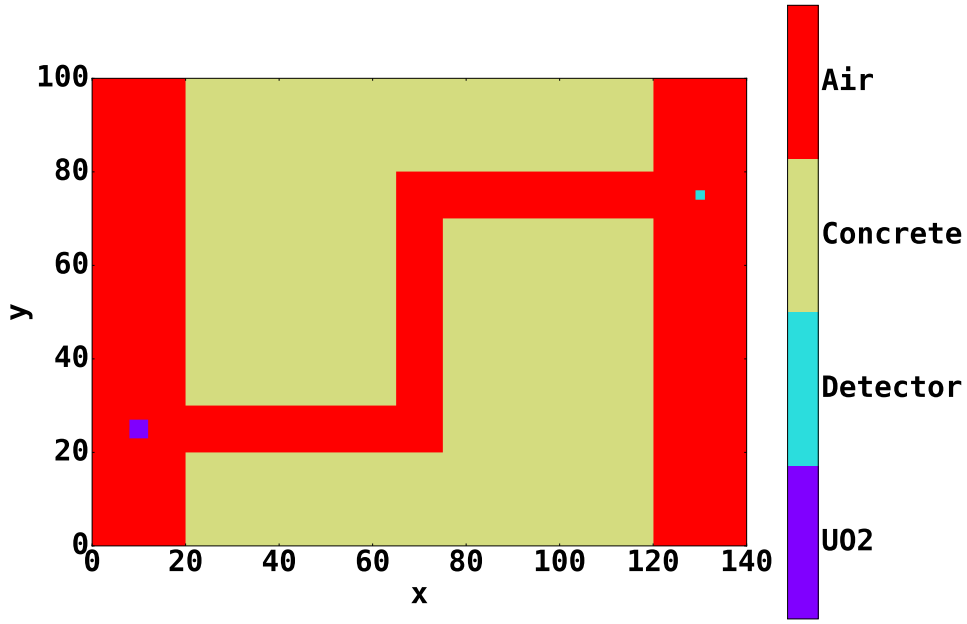


Figure 6.12: Geometry of dogleg duct.

6.5.3 Optimization Comparisons

We present a comparison of implicit filtering and the run-time diagnostic-based optimization for the dogleg duct problem with a $3 \times 2 \times 1$ and $6 \times 4 \times 1$ decomposition. In Figure 6.18, we plot convergence of three separate iterations versus total function evaluations. Each iteration of implicit filtering requires multiple function evaluations. For each function evaluation we simulate 240k, 120k, and 60k source particles and aim to minimize the Shift runtime. The initial iterate is a uniform allocation of 8 processors per subdomain, or 48 total processors for the $3 \times 2 \times 1$ decomposition. Currently, when coupling implicit

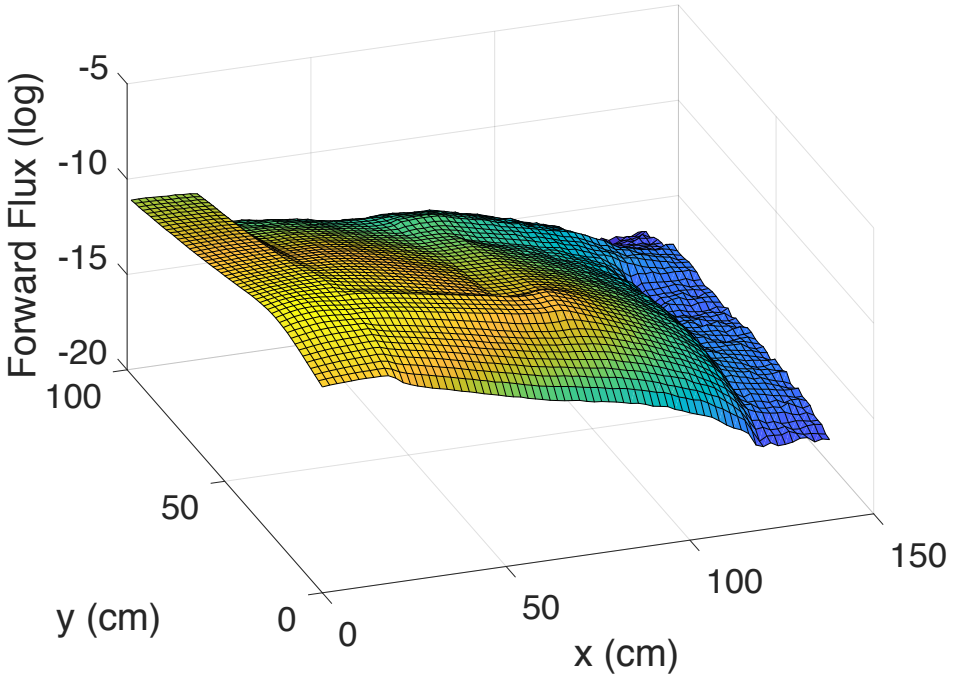


Figure 6.13: Forward flux for dogleg problem.

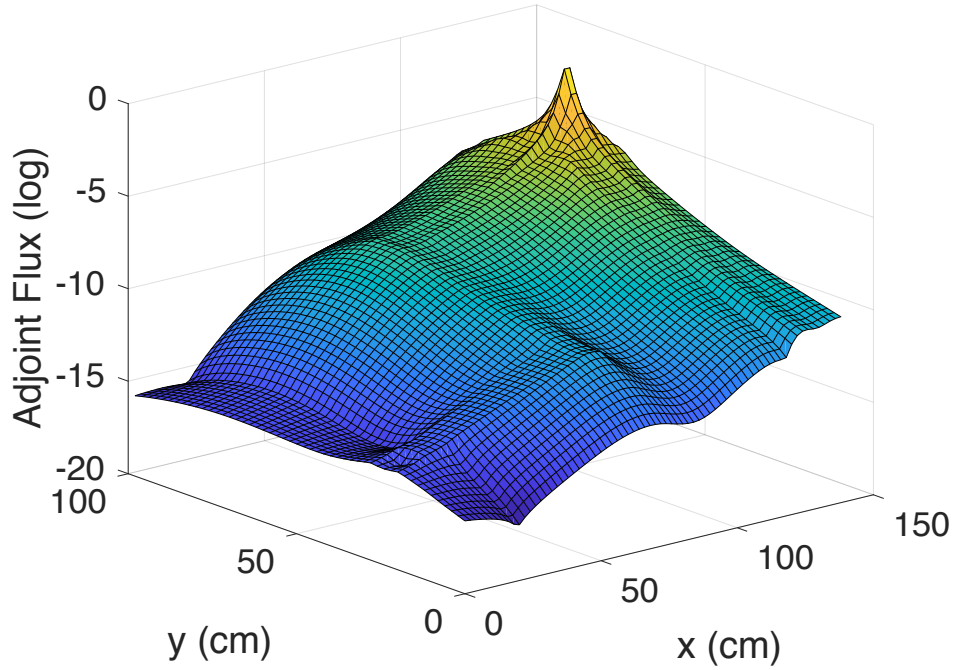


Figure 6.14: Adjoint flux for dogleg problem.

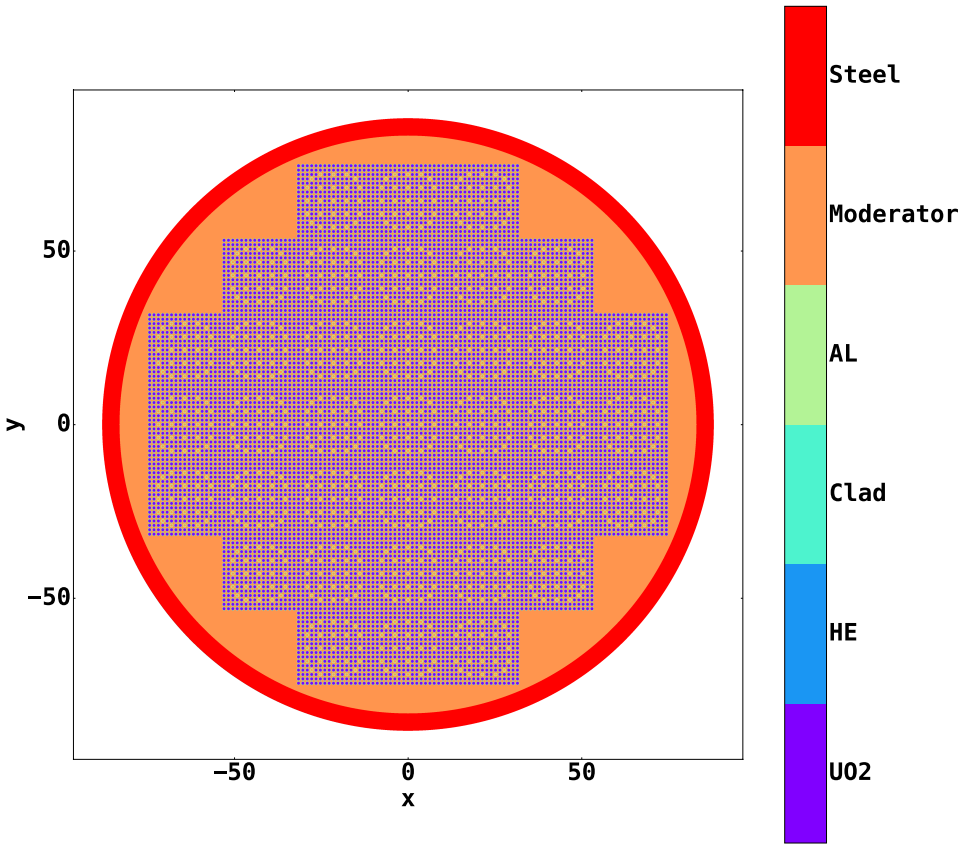


Figure 6.15: Geometry of SMR problem.

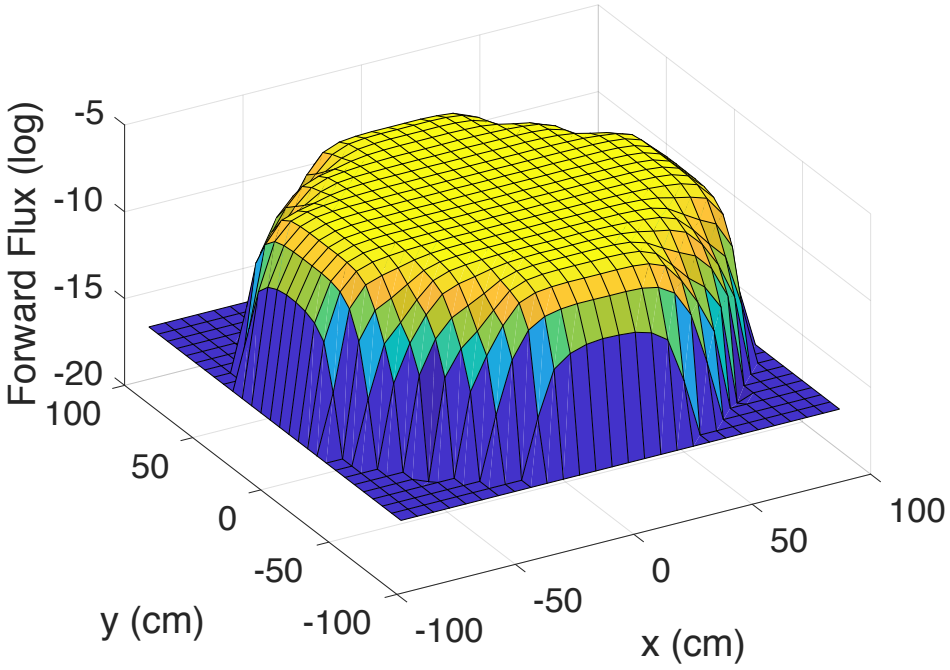


Figure 6.16: Forward flux for SMR problem.

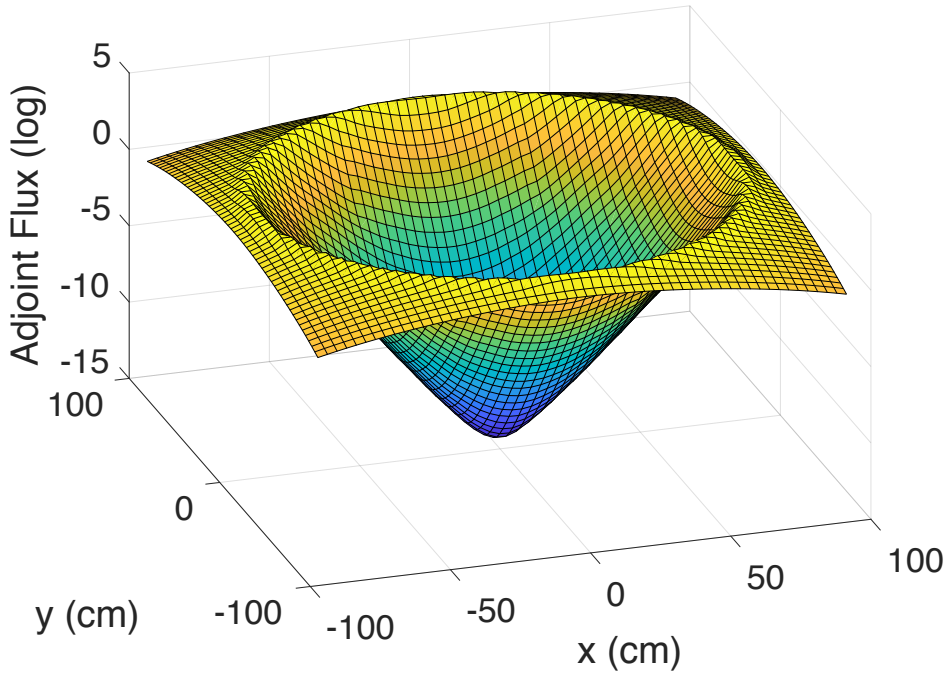


Figure 6.17: Adjoint flux for SMR problem.

filtering with Denovo and Shift, a full adjoint calculation must be performed each function evaluation. Adjoint calculations are not computationally cheap, and the number of necessary function evaluations only increases with finer decompositions. Implicit filtering requires at least 50 function evaluations to

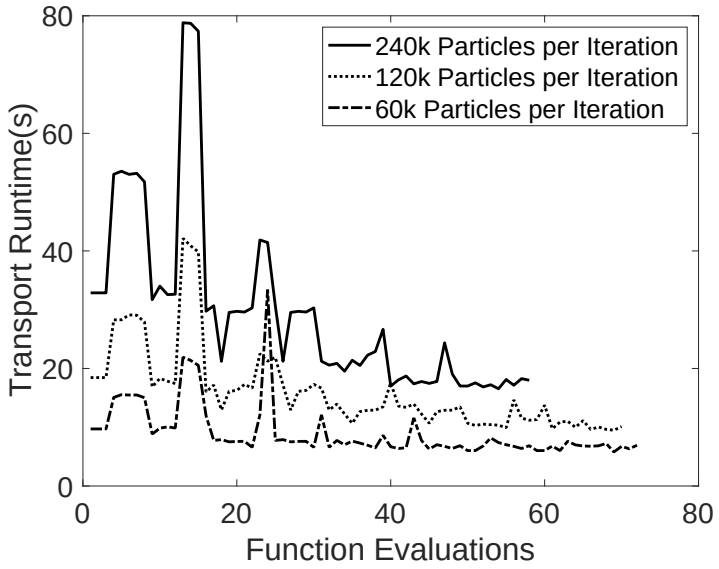


Figure 6.18: Implicit filtering iteration histories for the hybrid $3 \times 2 \times 1$ dogleg duct problem.

find the minimizer for the 240k particle case, which reduces the Shift runtime by 50%. As the number of source particles per iteration decreases, the improvement slightly worsens. With 60k particles per iteration, runtime is reduced by 40%.

We consider the $3\times2\times1$ and the $6\times4\times1$ decomposition (48 and 192 processors) for the diagnostic-based approach. The $6\times4\times1$ decomposition has 24 degrees of freedom and the linear constraint, which becomes an intractable problem for implicit filtering's direct search. To avoid the need for multiple function evaluations, the diagnostic-based approach collects the needed data and performs Algorithm 10, a single time. The diagnostic-based approach finds the minimizer in one function evaluation as long as the number of source particles is sufficient. In Figure 6.19, we plot the number of calibration source particles and the resulting post-calibration runtime for both decompositions. We simulate 2.4 million particles for the $3 \times 2 \times 1$ decomposition with 48 processors and 9.6 million particles for the $6 \times 4 \times 1$ decomposition with 192 processors. This maintains 50k particles per processor for each post-calibration run. With 1k calibration particles we roughly converge to the minimizer for both decompositions. The calibration step itself required a transport time of 1.4 seconds for the $6 \times 4 \times 1$ decomposition and 0.80 seconds for the $3\times2\times1$ decomposition. We observe, for these problems, that the optimization landscape

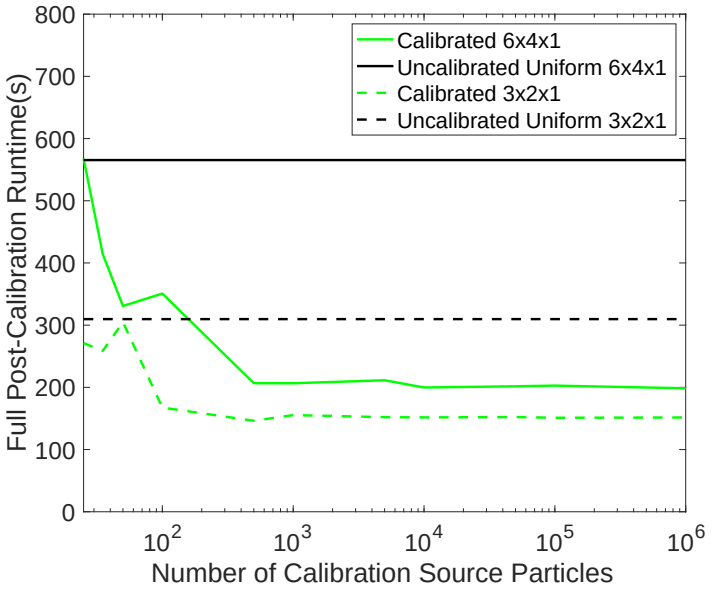


Figure 6.19: Source particles versus post-calibration runtime for the hybrid $3 \times 2 \times 1$ and $6 \times 4 \times 1$ dogleg duct problem.

near the minimizer is fairly flat, so the runtimes of calculations using allocations near the minimizer do not drastically change.

Once the optimized allocation is produced, the allocation remains static for the rest of the calculation. Further *dynamic* calibration may be required if the load shifts during the calculation. Figure 6.20 shows the transport times for each generation for each processor with and without the processor calibration. The generation timings for the uniform allocation has a conspicuous stair-stepping pattern. A step-down occurs when processors have less particles to transport than the generation size. After calibration, the stair-stepping is gone and each domain terminates at roughly generation 850. We reduce both the number of generations and the transport time for each generation. We see no part of the calibration where further calibration would be beneficial. The only load shifts still remaining are at the beginning and end of the calculation. Calibration specifically for these two periods would most likely not provide a net reduction in overall runtime. Comparatively, if we consider the time-dependent Implicit Monte Carlo method [17], it would require dynamic calibration due to the frequent shifts in load over time.

Importantly, the optimized generation timings contain single-generation spikes due to rare hybrid events. A particle with a large weight enters an important weight window region and massive particle splitting occurs. Each generation's runtime is dependent on the slowest processor, so the presence of many spikes may hurt the efficiency of the calculation.

In Figure 6.21, we marginalize out the generation variable to produce total transport and wait timings

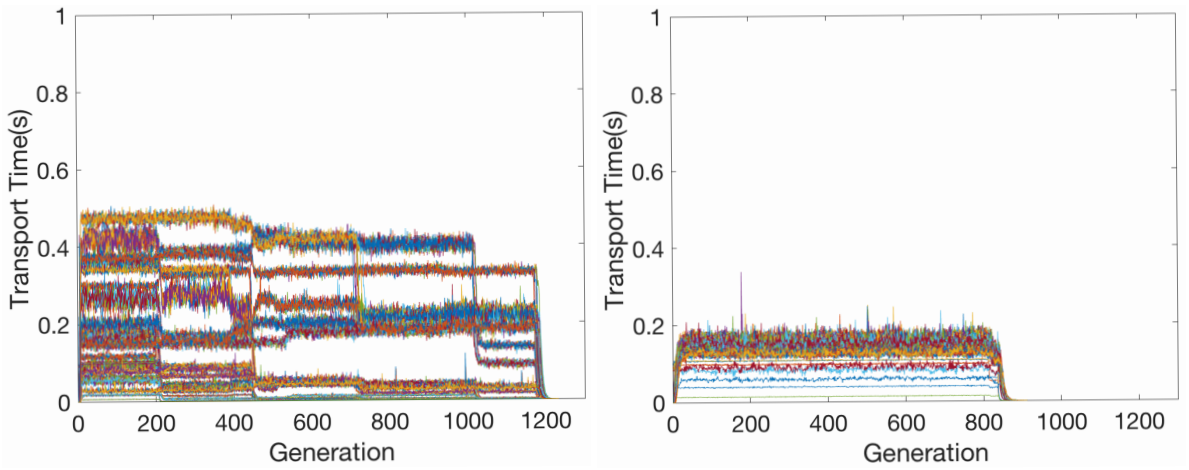


Figure 6.20: Uniform and optimized transport times each generation for every processor in the 6×4×1 hybrid dogleg duct problem.

for each processor. The wait time is the amount of time a processor spends idle after it transports its generation. The uniform transport timings cluster in groups of 8 processors for each subdomain in the problem, whereas the optimization transport timings all lie below a certain timing threshold. The timing of the highest uniform cluster is well above this optimized timing threshold. The difference reveals a remarkable reduction in runtime. The same is true for the wait timings, where the wait time is reduced from 500 seconds to roughly 50 seconds. The few processors above the threshold in the wait timings figure are processors with little work during the whole calculation. Their allocations are already set to just one processor and cannot be reduced further.

Overall, direct search algorithms, like implicit filtering, struggle to scale as the number of design variables increases [29]. The diagnostic-based approach scales excellently. It requires only enough particle histories to describe the space fully. Of the two methods, the diagnostic-based method outperforms implicit filtering in both performance and ease of implementation. We know of no cases where iterative methods would beat the diagnostic-based approach. Further results use diagnostics to optimize all processor allocations. We note that diagnostic-based optimization may have potential use for other algorithms with independent sampling that face on-node memory constraints and load imbalance.

6.5.4 Weak Scaling Study

We use a series of decompositions to explore how decomposition and load balancing affect transport performance. In the dogleg duct problem, we perform a weak scaling study maintaining 200k and 50k source particles for each processor in the calculation, respectively. A uniform allocation has 8 processors per subdomain. As an example, the 8×6×1 case uses 384 processors. Decompositions and Shift runtimes

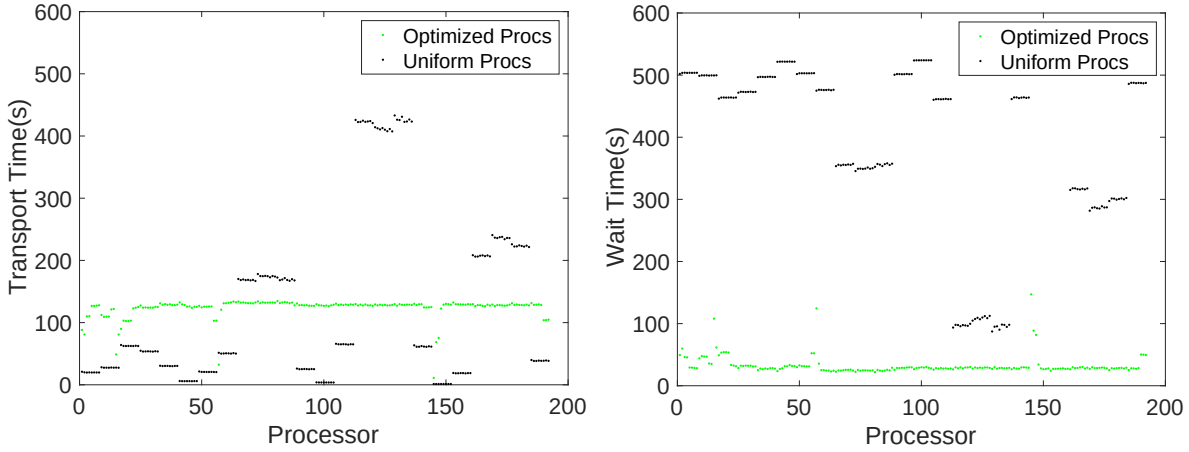


Figure 6.21: Transport and wait times over all generations for every processor in the $6 \times 4 \times 1$ hybrid dogleg duct problem.

Table 6.3: Weak scaling study and performance improvements for the forward dogleg duct problem.

Decomp.	**Processors**	**Shift Runtime (s) Uniform Procs**	**Shift Runtime (s) Optimized Procs**	**Ratio**
3x2x1	48	103.5	36.7	282.0%
4x3x1	96	131.7	41.1	320.4%
6x4x1	192	148.3	52.8	280.9%
8x6x1	384	306.3	100.9	303.6%

are seen in Tables 6.3 and 6.4 for the forward and hybrid cases, respectively. Processors and Shift parallel efficiencies are seen in Figures 6.22 and 6.23. The adjoint Denovo calculation is excluded in these runtimes, because we are measuring nonuniform processor allocation's effect on MC performance.

We achieved forward improvements of at least 224% with an upward trend as the decomposition is refined further. Optimized hybrid calculations produced at least a 186% improvement, also with an upward trend. Load imbalance and parallel inefficiency is known to increase with the number of subdomains [50]. In these cases, uniform processor allocation further loses parallel efficiency. And so, an optimal processor allocation sees more improvement as load imbalance worsens.

For the forward and hybrid SMR vessel fluence problems, we maintain 200k source particles per processor. The results of the weak scaling studies are provided in Table 6.5 and 6.6. Processors and Shift parallel efficiencies are seen in Figures 6.24 and 6.25.

Table 6.4: Weak scaling study and performance improvements for the hybrid dogleg duct problem.

Decomp.	Processors	Shift Runtime (s) Uniform Procs	Shift Runtime (s) Optimized Procs	Ratio
3x2x1	48	311.9	167.7	186.0%
4x3x1	96	371.1	182.5	203.3%
6x4x1	192	580.1	225.9	256.8%
8x6x1	384	780.3	383.8	203.3%

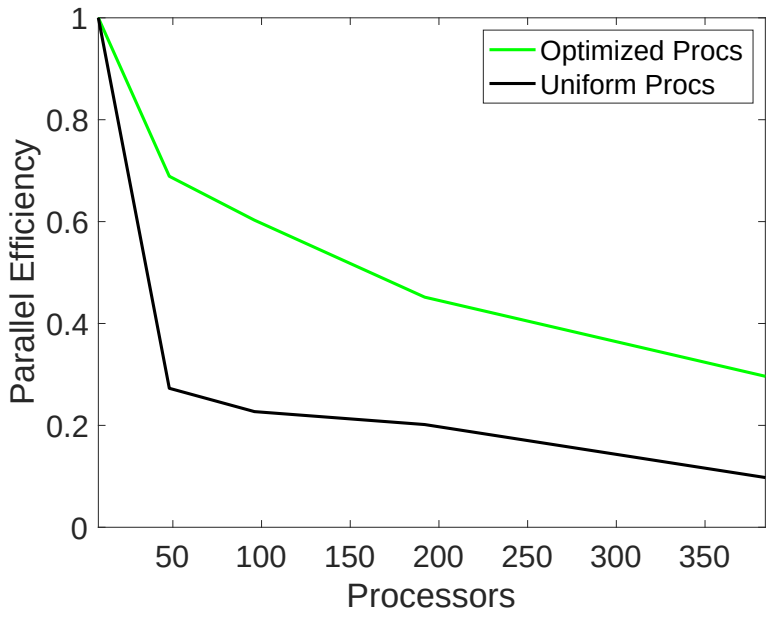


Figure 6.22: Forward Shift parallel efficiency versus processor count with uniform and optimized processor allocations for the dogleg duct problem.

Table 6.5: Weak scaling study and performance improvements for the forward SMR vessel fluence problem with fission source.

Decomp.	Processors	Shift Runtime (s) Uniform Procs	Shift Runtime (s) Optimized Procs	Ratio
2x2x1	32	116.3	116.3	100.0%
4x4x1	128	305.1	118.4	257.7%
6x6x1	288	371.8	149.3	249.0%
8x8x1	512	340.4	140.6	242.1%
10x10x1	800	393.3	135.7	289.8%

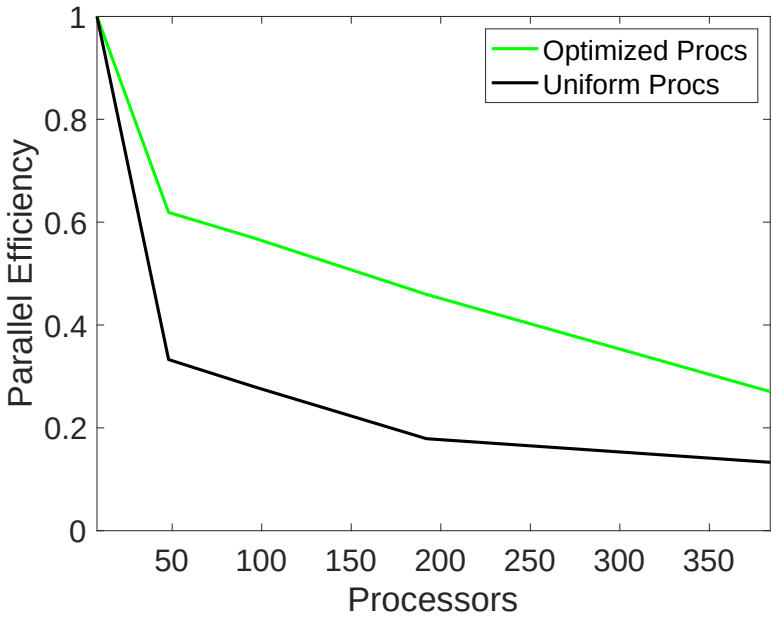


Figure 6.23: Hybrid Shift parallel efficiency versus processor count with uniform and optimized processor allocations for the dogleg duct problem.

Table 6.6: Weak scaling study and performance improvements for the hybrid SMR vessel fluence problem with biased source.

Decomp.	**Processors**	**Shift Runtime (s) Uniform Procs**	**Shift Runtime (s) Optimized Procs**	**Ratio**
2x2x1	32	119.4	119.4	100.0%
4x4x1	128	228.9	132.6	172.6%
6x6x1	288	409.0	186.1	219.8%
8x8x1	512	361.3	181.3	199.3%
10x10x1	800	657.4	262.1	250.8%

Forward calculations obtain runtime ratios of over 257%, whereas hybrid calculations see at least a ratio of 172% with an upward trend. The forward and hybrid $2\times2\times1$ cases see no improvement because there is only a single processor per subdomain. As with the dogleg duct problem, runtime increases as the decomposition is refined. The forward calculations with optimized allocations scale well to 800 processors for the $10\times10\times1$ decomposition. Timing increases due to domain decomposition are largely

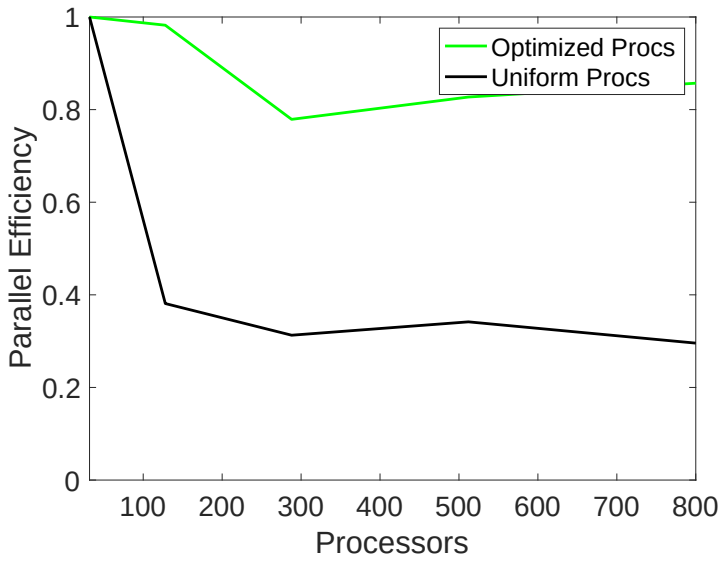


Figure 6.24: Forward Shift parallel efficiency versus processor count with uniform and optimized processor allocations for the SMR problem.

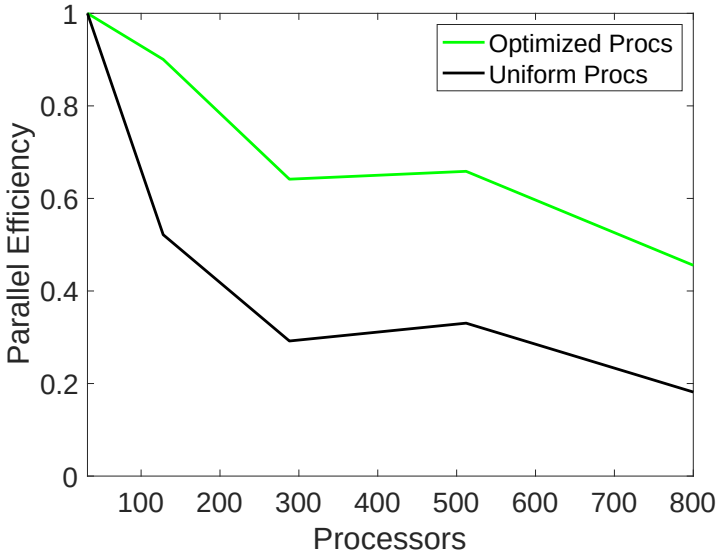


Figure 6.25: Hybrid Shift parallel efficiency versus processor count with uniform and optimized processor allocations for the SMR problem.

mitigated. This does not occur for uniform processor allocations. Hybrid calculation timings see upward trends for both uniform and optimized allocations. The trend for uniform allocations greatly outpaces the trend for optimized allocations. Runtime for uniform allocations grows by roughly a factor of 5.5 from the $2 \times 2 \times 1$ to the 10x10x1 decomposition. Comparatively, runtime for optimized allocations grows by only a factor of 2.2. Overall, nonuniform processor allocation provides greater improvements as load imbalance worsens.

We also present a weak scaling study of the hybrid $8 \times 8 \times 1$ SMR case over a series of total processor counts to observe how the ratios converge. High processor counts allow nonuniform processor allocation to fit the load's distribution more optimally. In Table 6.7, low processor counts do not fully capture the distribution and produce reduced speedups. As processor count increases, ratios quickly converge to near 200%. The same behavior is observed for other decompositions and test cases, as well. It is then possible to utilize the MSOD process, without loss of performance, to reduce the total number of communication buffers and messages. For example, we may restructure the 1024 processor calculation into 4 sets with 256 processors each at no loss of transport efficiency.

Table 6.7: Weak processor scaling study using an 8x8x1 decomposition for the hybrid SMR vessel fluence problem with biased source.

Processors	**Shift Runtime (s) Uniform Procs**	**Shift Runtime (s) Optimized Procs**	**Ratio**
64	319.9	319.9	100.0%
128	337.8	214.2	157.7%
256	352.4	174.3	202.2%
512	348.0	176.3	197.4%
1024	357.2	180.2	198.2%

6.6 Problem 5 Watts Bar Nuclear 1 Reactor Core

Problem 5 of The Virtual Environment for Reactor Applications (VERA) core physics benchmark progression problem specifications [19] models the Watts Bar Nuclear 1 (WBN1) initial start up core. It gives model specifications for simulation codes whose results are then compared with measurement data from WBN1. The WBN1 core's level of detail surpasses all previous test cases considered by our investigations.

6.6.1 Reactor Geometry

The WBN1 reactor geometry, seen in Figure 6.26, contains a full core of Westinghouse 17x17-type fuel assemblies at beginning-of-life. The dimensions of the core are 483x483x419cm. The specific assembly loading pattern is seen in Figure 6.27. It also contains the specific enrichment levels in percents of each assembly and the number of pyrex control rods. Control rods are used by the nuclear technicians to control the fission rate of the nuclear reactor. The model includes highly detailed representation of instrumentation at the top and bottom of the reactor. A more thorough overview of the geometry and instrumention representation is found in [19].

We may run the geometry with quarter symmetry but the instrumentation of the reactor is not symmetrical. In [19], only the quarter-core is considered due to the memory demands of the complex geometry. In their simulation, the quarter-core simulations required 10.7 GB of memory per core. Domain decomposition allows us to handle memory demands, so we may simulate the full-core geometry with asymmetrical instrumentation. We consider both forward and hybrid calculations for the full-core problem to test the processor allocation algorithm.

The resulting relative total fluxes of axial slices at various heights from the hybrid ex-core calculation are seen in Figures 6.28-6.31. Data is missing near the center of the core due to source biasing and repeated rouletting as particles move away from the vessel. In each slice, we see the vessel in which the core is placed. As we move away from the center axial height, total flux decreases substantially.

6.6.2 Numerical Results

We use a series of 3D decompositions to examine transport performance. We perform a weak scaling study maintaining 1 million source particles for each processor in the calculation. A uniform allocation has 16 processors per subdomain. Optimized allocations are calibrated using 10k particles for each processor. Decompositions, memory usage, and Shift runtimes are seen in Tables 6.8 and 6.9 for the forward and hybrid cases, respectively. We use 8 processor cores per node when reporting the per node memory usages for each Shift calculation (i.e. the number of nodes is equal to the number of processors divided by 8).

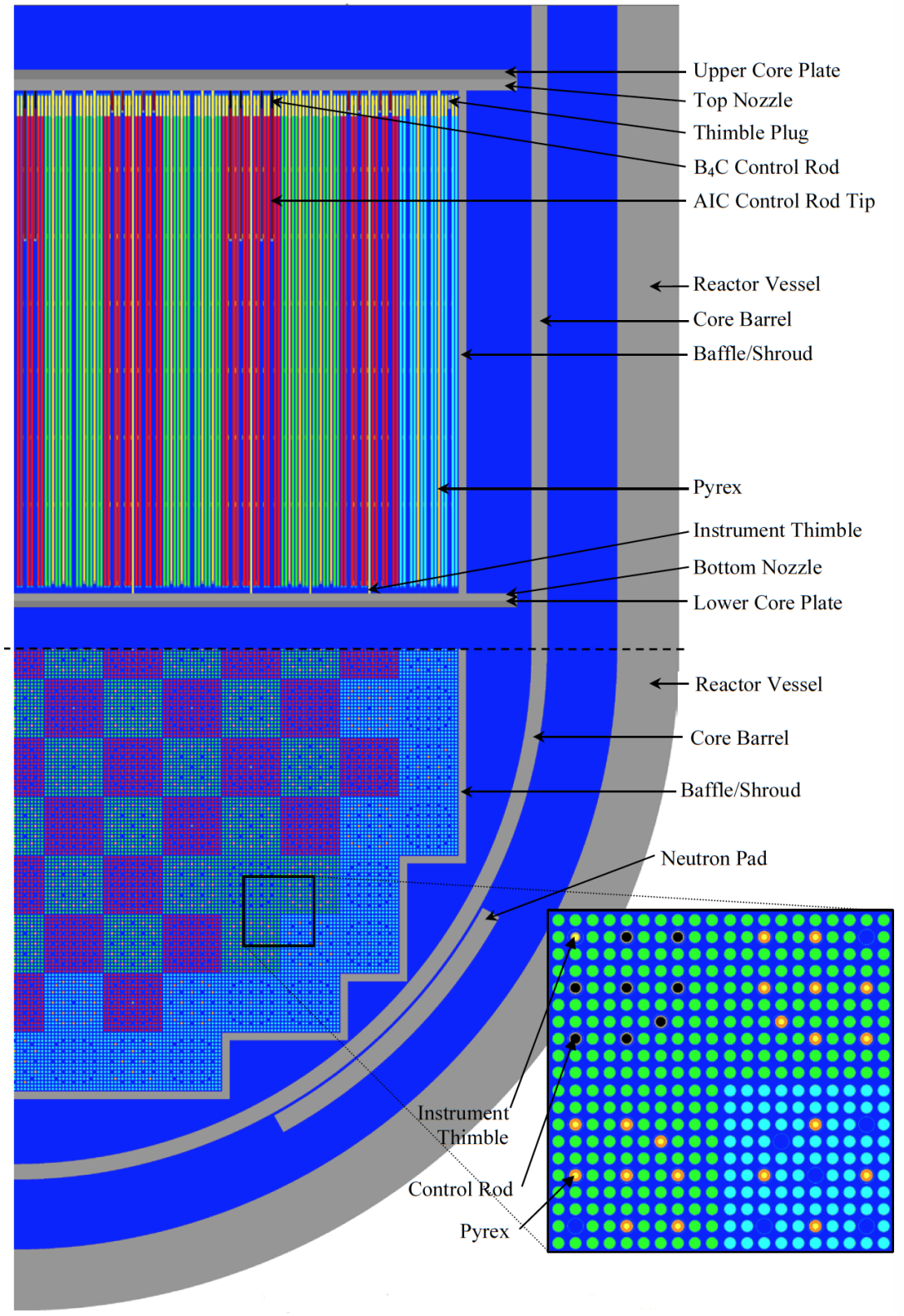


Figure 6.26: Problem 5 WBN1 reactor geometry.

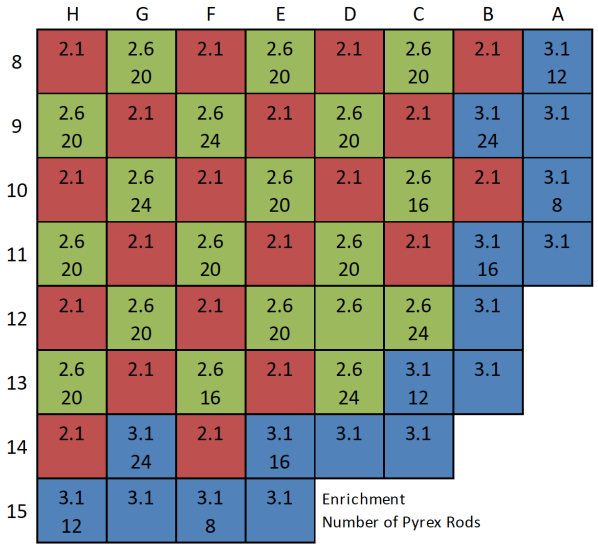


Figure 6.27: Problem 5 loading pattern with enrichment percentages and number of control rods.

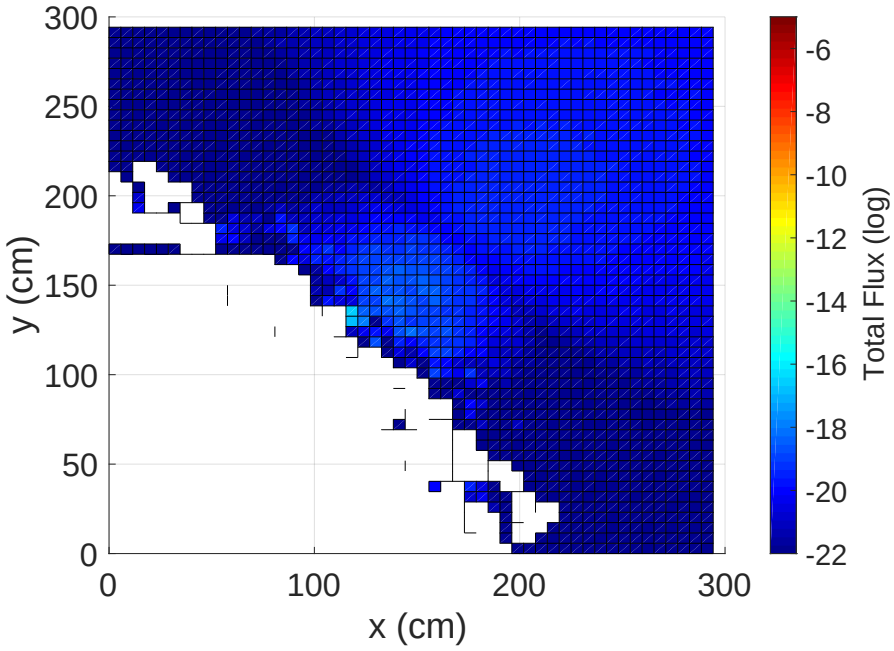
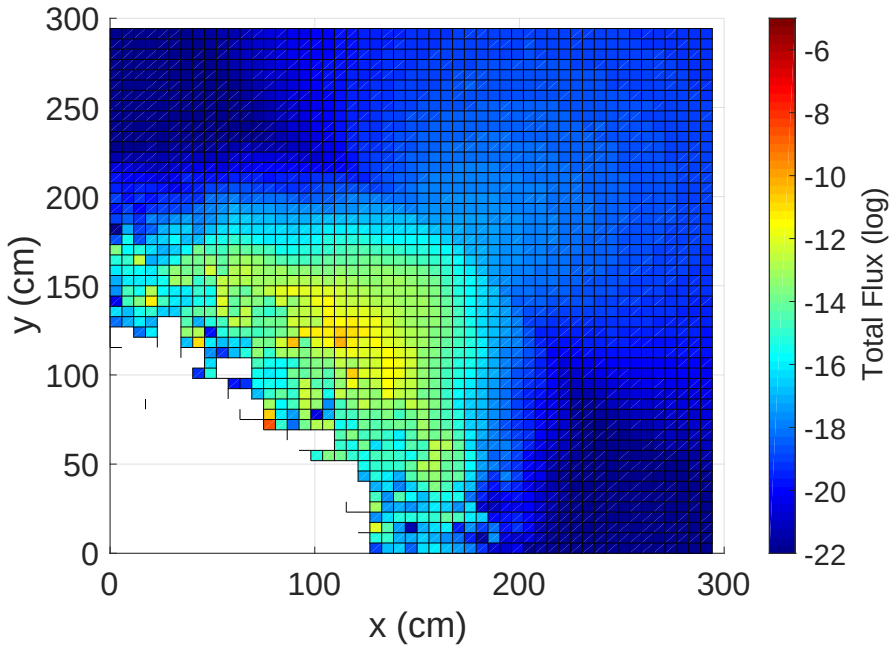


Figure 6.28: Problem 5 total flux for hybrid calculation in quarter symmetry at z = 40cm.



p5

Figure 6.29: Problem 5 total flux for hybrid calculation in quarter symmetry at z = 200cm.

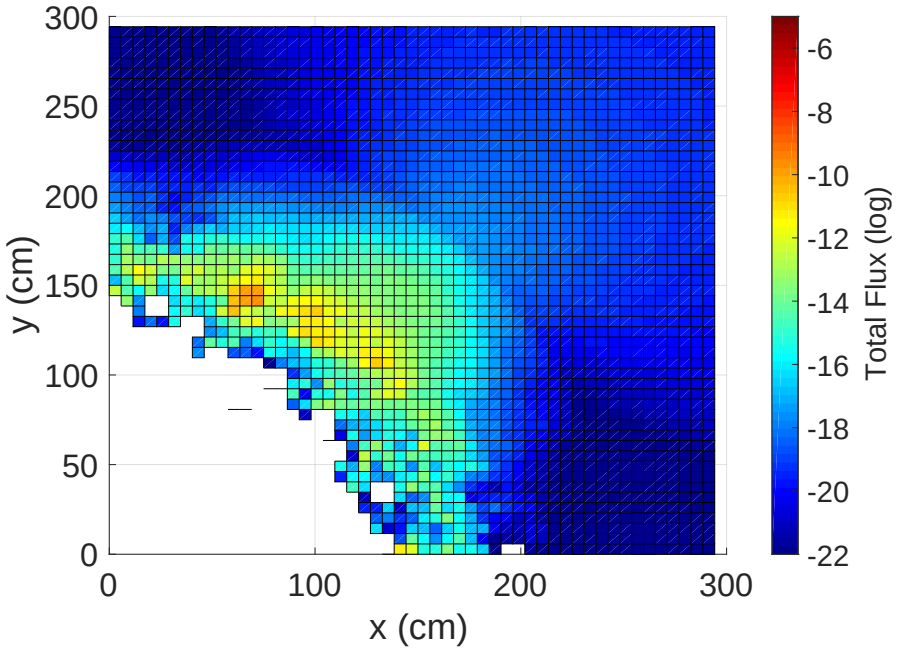


Figure 6.30: Problem 5 total flux for hybrid calculation in quarter symmetry at z = 325cm.

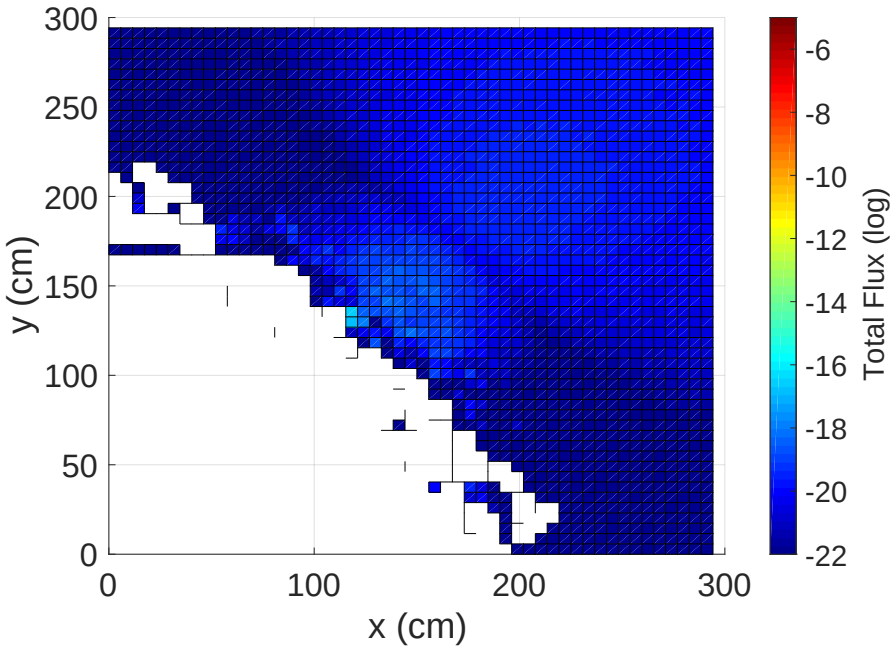


Figure 6.31: Problem 5 total flux for hybrid calculation in quarter symmetry at z = 400cm.

Table 6.8: Weak scaling study and performance improvements for the full forward P5 initial start up core.

Decomp.	**Processors**	**Memory Per Node**	**Shift Runtime (s) Uniform Procs**	**Shift Runtime (s) Optimized Procs**	**Ratio**
1x1x1	16	8.3GB	826.8	826.8	100.0%
1x1x2	32	8.5GB	917.3	917.3	100.0%
1x1x4	64	8.4GB	1482	927.6	160.0%
1x1x8	128	8.4GB	1764	945.0	186.7%
2x2x1	64	8.4GB	852.4	852.4	100.0%
2x2x2	128	8.4GB	934.2	885.9	105.5%
2x2x4	256	8.5GB	1524	918.8	165.9%
4x4x1	256	8.6GB	3350	1287	260.3%
4x4x2	512	9.0GB	3701	1360	272.1%
4x4x4	1024	9.9GB	6024	1331	452.6%

Table 6.9: Weak scaling study and performance improvements for the full hybrid P5 initial start up core.

Decomp.	Processors	Memory Per Node	Shift Runtime (s) Uniform Procs	Shift Runtime (s) Optimized Procs	Ratio
1x1x1	16	> 32.0GB	MEM	MEM	—%
1x1x2	32	20.8GB	417.5	417.5	100.0%
1x1x4	64	15.0GB	433.4	365.2	118.7%
1x1x8	128	12.1GB	455.9	418.7	108.9%
2x2x1	64	14.9GB	351.2	351.2	100.0%
2x2x2	128	11.9GB	364.1	364.1	100.0%
2x2x4	256	10.4GB	459.3	411.2	111.7%
3x3x1	144	11.6GB	746.1	386.8	192.9%
3x3x2	288	10.3GB	806.1	401.1	201.0%
3x3x4	576	9.8GB	1023	457.0	224.9%
4x4x1	256	11.1GB	630.0	624.6	100.8%
4x4x2	512	10.1GB	704.3	717.4	98.1%
4x4x4	1024	10.4GB	903.4	770.2	117.3%

Forward calculations obtain runtime ratios of over 160% with greater gains for finer decompositions. The 4×4×4 decomposition achieves a ratio of 452%. Decompositions that see zero or little improvement, such as the $2 \times 2 \times 2$ decomposition, are already load balanced. As we further refine in z, runtimes increase for both the uniform and optimized cases. The $2 \times 2 \times 1$ and $2 \times 2 \times 2$ decompositions are both geometrically symmetric (ignoring the top and bottom reactor instrumentation), so uniform and optimized runtimes do not differ as much as the $2 \times 2 \times 4$ case. For all cases, calibration does no worse than the uniform default; therefore, users should not fear that MC transport times will increase.

Hybrid calculations obtain more moderate runtime ratios for most cases—with the exception of the 3×3 decompositions. Hybrid runtimes did not degrade as rapidly as the forward runtimes. Small hybrid runtime degradation occurs when we decompose the z component past 2 divisions. Compared to other decompositions, the 3×3 decompositions are outliers. The 1×1, 2×2, and 4×4 were all relatively load balanced and saw zero to moderate speedups. Also, the optimal processor allocations produced were not drastically different than the uniform allocations. Only in the 3×3 cases do we see substantial improvements. This is due to how the subdomain boundaries lie, showing that the user chosen decomposition is important to performance. The optimized 3×3 decompositions allocated all

processors to the four corner subdomains in the x, y-plane. Neither the 2×2 nor 4×4 isolates the work enough to lead to a load imbalance.

As a recommendation, users should target coarse decompositions that avoid elongated subdomain shapes and minimize the surface-area-to-volume ratio. Higher surface area leads to a greater chance that particles cross subdomain boundaries, requiring particle communication. Also, if users have prior knowledge of the path particles will take, they should avoid placing successive subdomain boundaries in the path. This is revealed by considering the difference between the $2\times2\times4$ and $4\times4\times1$ hybrid decompositions. Both have 16 subdomains but have different runtimes, both uniform and optimized. CADIS is biasing particles radially outward towards the reactor vessel, so reducing the number of communications per particle history may reduce runtime.

The memory scalings show noticeable difference between the forward and hybrid calculations. The main memory burden of forward calculations is the continuous energy data, roughly 8GB, which must exist on each node. Comparatively, the main memory burdens of the hybrid calculations are the continuous energy data, the weight window mesh, and the biased sources. For hybrid calculations, the continuous energy data still exists on each node, but we partition both the weight window mesh and the biased sources. This explains the fairly flat memory scaling for the forward calculations and the asymptotic memory scaling for the hybrid calculations.

Processors and Shift parallel efficiencies are seen in Figures 6.32 and 6.33. In both figures, each tick marker is a given $x \times y$ decomposition and the z decomposition varies. For example, the 1×1 decompositions are represented by a '+' and $z = \{1, 2, 3, \text{or } 4\}$. This allows us to see the runtime effects of different decompositions that use the same number of subdomains. We see consistent degradation of efficiency for both uniform and optimized as the decompositions are refined, though degradation is far worse without calibration. In the forward case, we remain above 50% efficiency for all optimized cases and well above 80% for the coarsest cases, the common target decompositions. In the hybrid case, improvements are more moderate, but we also stay above 80% efficiency for optimized cases up to $3 \times 3 \times 4$. The optimized 4×4 decompositions are noticeably separated from the other optimized decompositions.

Overall, the WBN1 calculations obtain relatively similar improvements to the dogleg duct and SMR problem when decompositions are load imbalanced. Nonuniform processor allocation and the diagnostic-based optimization strategy reliably reduce load imbalance on domain decomposed radiation transport calculations. Using this technique we recover much of the parallel efficiency lost by uniform processor allocations that was used in Shift's original domain decomposition implementation. The default nature of most source-detector radiation transport problems is load imbalanced, so exploring effective strategies is essential. The diagnostic-based algorithm far surpasses iterative optimization methods in performance by avoiding multiple costly function evaluations in a high performance computing environment. Together, nonuniform processor allocation and diagnostic-based optimization deliver performance improvements that scale with the severity of the load imbalance and problem size, at minimal

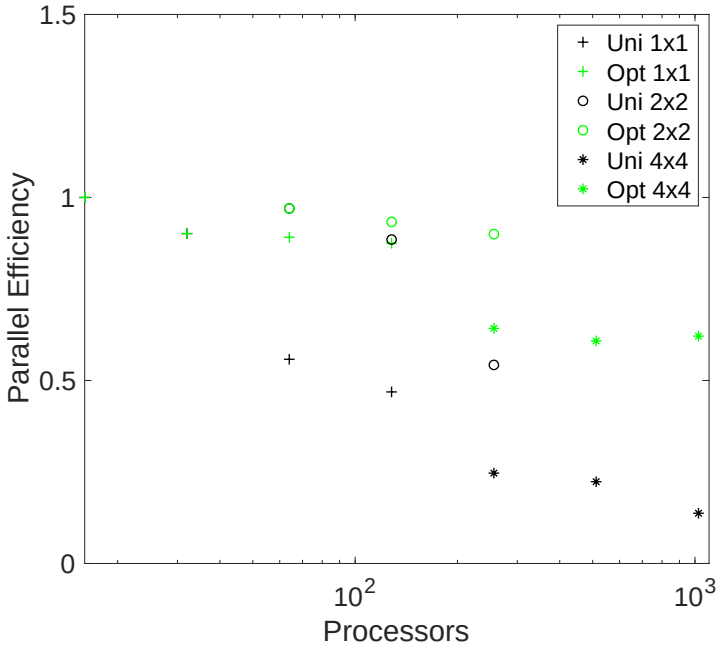


Figure 6.32: Forward Shift parallel efficiency versus processor count with uniform and optimized processor allocations for the WNB1 reactor problem.

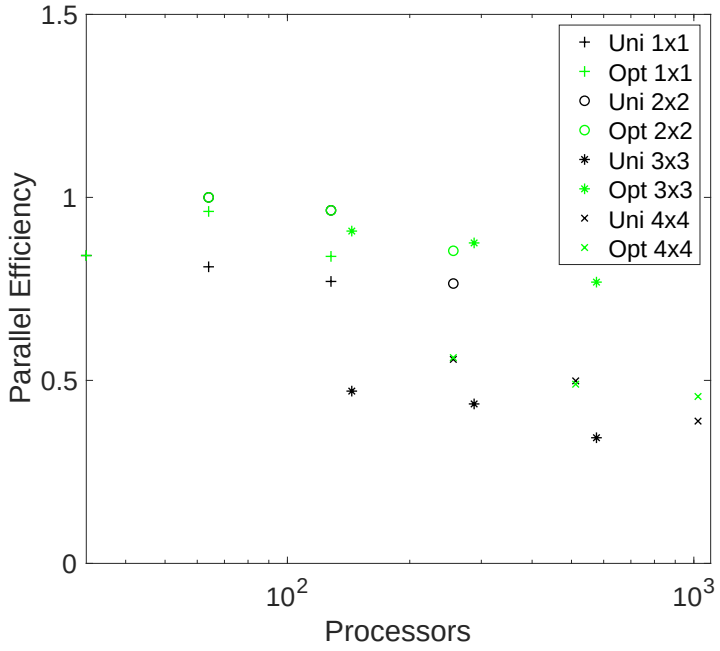


Figure 6.33: Hybrid Shift parallel efficiency versus processor count with uniform and optimized processor allocations for the WNB1 reactor problem.

additional cost to Shift users. Problems with greater load imbalance will see an even greater reduction in runtime over the uniform implementation. We improve our parallel efficiency in nearly all cases, and commonly above 60-80% in the single set case. This allows users to use domain replication to scale out to hundreds of thousands of cores on leadership compute platforms. Running domain decomposed Monte Carlo efficiently in general is a challenge, and so Shift users facing data burdens will be more apt to decompose their problem without complete deterioration of parallel performance.

Chapter 7

Conclusions

In this thesis, we have introduced techniques for solving the fixed source neutron transport equation. The main focus has been on diagnosing and improving performance of the domain decomposed Monte Carlo implementation found in the massively parallel Shift Monte Carlo code developed at Oak Ridge National Laboratory. We began with critical background information such as the transport equation derivation, transport terms, boundary conditions, and various discretizations. Next, we introduced basic deterministic methods for solving the one-speed transport equation in slab geometry, along with two acceleration methods: Anderson acceleration and nonlinear diffusion acceleration. We analyzed the effects of the convergence of these acceleration schemes when randomized noise was embedded into the function evaluation. We observed that as long as the noise was small enough the Anderson accelerated case converged no worse than the unaccelerated case.

We continued to stochastic Monte Carlo methods for solving the 3D transport equation. The Monte Carlo method involves simulating a finite number of independent particle histories inside a domain. Particles are birthed by sampling probability distribution functions for particle location, direction, and energy. The Monte Carlo method also constructs and samples various probability distribution functions to advance particles through their histories. The Monte Carlo method then collects quantities of interest at specific locations through tally objects declared by the user. The quantities of interest have associated variances that may be decreased through variance reduction techniques. We described basic techniques, such as Russian rouletting and implicit capture, and hybrid techniques, such as consistent adjoint-driven importance sampling(CADIS).

We explained how CADIS is implemented in the Exnihilo code suite that contains both the Shift Monte Carlo code and the Denovo deterministic code. CADIS combines a deterministic adjoint and stochastic calculation to produce a massive reduction in variance at a single tally location. The target calculations of our investigations are large hybrid problems. As problems scale larger and larger, memory demands become limiting; therefore, we must employ domain decomposition in order to fit the problem on compute nodes. We described the exact domain decomposition algorithm implemented

in Shift and how a load imbalance problem arises. We introduced the nonuniform processor allocation capability that we developed to match the problem's load with an appropriate number of processors for each subdomain. We optimize this allocation with a cheap diagnostic-based approach, which far outperforms iterative-based optimization algorithms. All contributions to the Shift code base is found in Appendix A.3.

Numerical results include the new Shift capabilities applied to two representative test cases: a shielding dogleg duct problem and a small modular reactor(SMR) problem. We show that the new approach cheaply and robustly reduces runtime and improves parallel performance. Static processor allocations appear to be sufficient in alleviating the load imbalance; therefore, further dynamic processor allocation seems unnecessary for the time-independent case. We continue to a larger challenge problem, the Watts Bar Nuclear 1 initial start up core. For the forward calculations, different decompositions had a noticeable impact on runtimes. Decompositions which lost significant performance with uniform allocations were able to recover efficiency with optimized allocations. Improvement ratios for hybrid calculations were even more dependent on the decompositions—a majority of the decompositions were already load balanced. This dependence indicates that user selected subdomain boundaries influence uniform and optimized Shift performance greatly. As a recommendation, users should try to minimize the surface-area-to-volume ratio in their decompositions and avoid placing subdomain boundaries in the "high traffic" areas.

7.1 Future Work

The Shift development team is currently working on a CUDA implementation of domain decomposition and nonuniform processor allocation for Shift's GPU-based transport algorithm. The CUDA programming language [2] was developed by NVIDIA for programming their NVIDIA GPUs, which are used by leadership high performance computing platforms, like the Titan and Summit machines at the Oak Ridge Leadership Computing Facility (OLCF) and the Sierra machine at Lawrence Livermore National Laboratory [36, 42, 43]. The Shift team has reported dramatic GPU acceleration on both Titan and Summit over CPU implementations [22, 23]. The current plan is to continue expanding Shift capabilities that can be run on GPUs.

When considering domain decomposition and its optimization, a new set of constraints are present in GPU implementations. The two conditions of primary importance are GPU occupancy and reduction in *thread divergence*. GPUs contain many independent *multiprocessors*, processors that execute *kernels* on a large number of threads simultaneously. Kernels are sections of CUDA code that CPUs launch on the GPUs during a calculation. Multiprocessor threads are broken down into *thread-blocks* which are further decomposed into sets of 32 threads called *warps*. A GPU's occupancy is the the ratio of active warps on the multiprocessor to the maximum possible number of active warps. Applications may have difficulty keeping the GPUs occupied due to the GPUs fast processing speed and the communication

latency of getting new work to the GPUs.

As for thread divergence, warps use the *single instruction, multiple threads*(SIMT) execution model, which demands that any instruction executed by a thread must be executed by every thread in the warp. Thread divergence occurs when warps encounter branching statements. Threads are marked *inactive* if the thread is not meant to go down a specific branch path. The inactive thread still performs the calculation but the result is, ultimately, discarded. If high thread divergence occurs, warps may waste considerable compute time on unnecessary calculations impacting performance. Reducing thread divergence through wise kernel launching may substantially improve runtime and efficiency. This was shown by the event-based MC algorithm implemented in Shift [22].

The Shift CUDA implementation uses a work pool of active particles to keep warps occupied and kernel launches efficient. Kernels are launched on subsets of the active particles. For example, there are separate kernels for moving particles to the next geometric boundary, processing collisions, sampling the source distribution, etc. When a particle is killed during a collision, it is set to inactive and replaced by a new particle from source. The first step in a CUDA implementation of domain decomposition would add new events for subdomain boundary crossings and replacing inactive particles with bank particles. We must also develop a new syncing strategy to determine when to communicate particles across subdomains.

The Summit machine's Volta-series V100 GPUs have 80 multiprocessors and 16GB of memory. The implementation must minimize the size of communication particle buffers and particle banks that must be maintained, occupying valuable device memory. The communication pattern we developed for the CPU implementation in Chapter 6 is a good place to begin. Perhaps a single communication buffer the size of the generation on the GPU might be optimal, instead of individual buffers for neighboring processors. Then, we send the single particle buffer from GPU to CPU. The CPU sorts particles into individual communication buffers to send out to appropriate CPUs. CPUs will upload incoming particles to its GPUs and dump them into bank for the GPUs to transport.

Further, optimizing nonuniform processor allocation must take into account memory limitations as it effects GPU occupancy to maintain performance. A more nuanced allocation algorithm would need to be developed, including a GPU performance model. The performance model would accept as inputs calibration diagnostics and GPU memory limitations, and it would provide a nonuniform GPU allocation. Perhaps we should prefer a slightly increased number of sets over an optimally load balanced allocation. More sets with less "processors" per set could free up memory needed to maintain communication buffers. [22] showed that larger kernel launches led to higher throughput on each GPU. A thorough cost-benefit analysis would need to completed.

Another area of potential interest is in load balancing time-dependent Monte Carlo methods, such as Implicit Monte Carlo(IMC) [17]. Dynamic load balancing would be necessary as the load shifts over successive time steps. Investigations could examine the criteria (probably diagnostic-based) of when and how often to recalibrate. Computational costs and implementation complexity of frequent data transferal

would have to be taken into account for a successful method. Also, the Shift development team could investigate if there are other potential parameters that could be optimized with diagnostics using the calibration framework already implemented. One potential candidate is the particle generation size.

In conclusion, as the Department of Energy HPC community continually moves closer to exascale through next generation heterogeneous platforms, like the Summit and soon-to-come Frontier machines, efficient resource utilization becomes critical. We see nonuniform processor allocation and diagnostic-based optimization having a similar performance impact as long as we maintain GPU occupancy on single compute nodes.

REFERENCES

[1] *SCALE: A Comprehensive Modeling and Simulation Suite for Nuclear Safety Analysis and Design*. ORNL/TM-2005/39, Version 6.2, Oak Ridge National Laboratory, Oak Ridge, Tennessee (2016). Available from Radiation Safety Information Computational Center as CCC-834.

[2] CUDA C programming guide. Technical Report Tech. Rep. PG-02829-001_v7.5, NVIDIA, 2015.

[3] D.G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12(4):547–560, October 1965.

[4] G.I. Bell and S. Glasstone. *Nuclear Reactor Theory*. Van Nostrand Reinhold Company, New York, 1970.

[5] S.M. Bowman, I.C. Gauld, and J.C. Wagner. *Recommendations on Fuel Parameters for Standard Technical Specifications for Spent Fuel Storage Casks*. Oak Ridge National Laboratory, Oak Ridge, Tennessee, 2001.

[6] T.A. Brunner and P.S. Brantley. An efficient, robust, domain-decomposition algorithm for particle Monte Carlo. *J. Comp. Phys.*, 228:3882–3890, 2009.

[7] T.A. Brunner, T.J. Urbatsch, T.M. Evans, and N.A. Gentile. Comparison of four parallel algorithms for domain decomposed implicit Monte Carlo. *J. Comp. Phys.*, 212:527–539, 2006.

[8] I.W. Busbridge. *The Mathematics of Radiative Transfer*. Number 50 in Cambridge Tracts. Cambridge Univ. Press, Cambridge, 1960.

[9] S. Chandrasekhar. *Radiative Transfer*. Dover, New York, 1960.

[10] A.R. Conn, K. Scheinberg, and P.L. Toint. Recent progress in unconstrained optimization without derivatives. *Math. Program. Ser. B*, 79:397–414, 1997.

[11] M.A. Cooper and E.W. Larsen. Automated weight windows for global Monte Carlo particle transport calculations. *Nuclear Sci. Eng.*, pages 1–13.

[12] J.J. Duderstadt and L.J. Hamilton. *Nuclear Reactor Analysis*. John Wiley & Sons, Inc., New York, 1976.

[13] J.J. Duderstadt and W.R. Martin. *Transport Theory*. John Wiley & Sons, Inc., New York, 1979.

[14] J.A. Ellis, T.M. Evans, S.P. Hamilton, C.T. Kelley, and T.M. Pandya. Optimization of processor allocation for domain decomposed Monte Carlo calculations. *Submitted to Parallel Comp.*, 2018.

[15] T.M. Evans, S.P. Hamilton, and S.R. Slattery. ORNL-CEES/Profugus, 2016. Oak Ridge National Laboratory.

[16] T.M. Evans, A.S. Stafford, R.N. Slaybaugh, and K.T. Clarno. Denovo: A new three-dimensional parallel discrete ordinates code in SCALE. *Nuclear Tech.*, 171(2):171–200, 2010.

[17] J.A. Fleck and J.D. Cummings. An implicit Monte Carlo scheme for time and frequency dependent nonlinear radiation transport. *J. Comp. Phys.*, 8:313–342, 1971.

[18] R.W. Freund. A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems. *SIAM J. Sci. Comp.*, 14:470–482, 1993.

[19] A.T. Godfrey. Vera Core Physics Benchmark Progression Problem Specifications. Technical Report CASL-U-2012-0131-004, Consortium for Advanced Simulation of LWRs, 2014.

[20] Google. Googletest, 2018. https://github.com/google/googletest.

[21] S.P. Hamilton, M. Berrill, K.T. Clarno, R. Pawlowski, A. Toth, C.T. Kelley, T.M. Evans, and B. Philip. An assessment of coupling algorithms for nuclear reactor core physics simulations. *Journal of Computational Physics*, 311:241–257, 2016.

[22] S.P. Hamilton and T.M. Evans. Continuous-energy Monte Carlo neutron transport on GPUs in the shift code. *Submitted to Annals of Nuclear Energy*.

[23] S.P. Hamilton, S.R. Slattery, and T.M. Evans. Multigroup Monte Carlo on GPUs: Comparison of history- and event-based algorithms. *Annals of Nuclear Energy*, 113:506–518, March 2018.

[24] M.A. Heroux, R.A. Bartlett, V.E. Howle, R.J. Hoekstra, J.J. Hu, T.G. Kolda, R.B. Lehoucq, K.R. Long, R. Pawlowski, E.T. Phipps, A.G. Salinger, H.K. Thornquist, R.S. Tuminaro, J.M. Willenbring, A. Williams, and K.S. Stanley. An overview of the trilinos project. *ACM Trans. Math. Softw.*, 31(3):397–423, 2005.

[25] M.R. Hestenes and E. Steifel. Methods of conjugate gradient for solving linear systems. *J. of Res. Nat. Bureau Standards*, 49:409–436, 1952.

[26] N. Horelik, A.R. Siegel, B. Forget, and K.S. Smith. Monte Carlo domain decomposition for robust nuclear reactor analysis. *Parallel Computing*, 40:646–660, 2014.

[27] I. Ipsen. *Numerical Matrix Analysis*. SIAM, Philadelphia, Pennsylvania, 2009.

[28] C. T. Kelley. *Iterative Methods for Linear and Nonlinear Equations*. Number 16 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 1995.

[29] C. T. Kelley. *Iterative Methods for Optimization*. Number 18 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 1999.

[30] C.T. Kelley. Multilevel source iteration accelerators for the linear transport equation in slab geometry. *Transport Theory and Statistical Physics*, 24(4):679–707, 1995.

[31] C.T. Kelley. *Implicit Filtering*. SIAM, Philadelphia, 2011.

[32] D.A. Knoll, H. Park, and K.S. Smith. Application of the Jacobian-free Newton-Krylov method to nonlinear acceleration of transport source iteration in slab geometry. *Nuclear Sci. Eng.*, 167:122–132, 2011.

[33] K.A. Kobayashi. Proposal for 3D radiation transport benchmarks for simple geometries with void region. *Prog, in Nuc. Energy*, 39:119–144, 2001.

[34] E. Larsen. Diffusion-synthetic acceleration methods for discrete ordinates problems. *Trans. Th. Stat. Phys.*, 13:107–126, 1984.

[35] K.D. Lathrop. Spatial differencing of the transport equation: Positivity vs. accuracy. *J. Comp. Phys.*, 4:475–498, 1969.

[36] Lawrence Livermore National Laboratory Livermore Computing Center. Sierra, May 2018. https://hpc.llnl.gov/hardware/platforms/sierra.

[37] E.E. Lewis and W.F. Miller. *Computational Methods of Neutron Transport*. John Wiley & Sons, Inc., New York, 1984.

[38] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard Version 4.0, November 2017. http://mpi-forum.org/.

[39] N. Metropolis and S. Ulam. The Monte Carlo method. *J. Am. Stat. Assoc.*, 44:335–341, 1949.

[40] C.D. Meyer. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[41] J.A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7:308–313, 1965.

[42] Oak Ridge Leadership Computing Facility. Titan Cray XK7, August 2016. https://www.olcf.ornl.gov/olcf-resources/compute-systems/titan/.

[43] Oak Ridge Leadership Computing Facility. Summit: Oak Ridge National Laboratory's next high performance supercomputer, April 2018. https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.

[44] M.J. O'Brien. *Scalable Domain Decomposed Monte Carlo Particle Transport*. PhD thesis, University of California Davis, 2013.

[45] OpenMP. OpenMP Application Programming Interface Version 4.5, November 2015. http://www.openmp.org/.

[46] T.M. Pandya, S.R. Johnson, T.M. Evans, G.G. Davidson, S.P. Hamilton, and A.T. Godfrey. Implementation, capabilities, and benchmarking of Shift, a massively parallel Monte Carlo radiation transport code. *J. Comp. Phys.*, 308:239–272, 2016.

[47] M.J.D. Powell. Uobyqa: Unconstrained optimization by quadratic approximation. *Math. Program.*, 92:555–582, 2002.

[48] J. Reddy. *An Introduction to the Finite Element Method*. McGraw-Hill, New York, 2 edition, 1993.

[49] Y. Saad and M. Schultz. GMRES a generalized minimum residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Statist. Comput.*, 7:856–869, 1986.

[50] A.R. Siegel, K.S. Smith, P.K. Romano, B. Forget, and K. Felker. The effect of load imbalances on the performance of Monte Carlo algorithms in LWR analysis. *J. Comp. Phys.*, 235:901–911, 2013.

[51] K.S. Smith and J.D. Rhodes III. Full-core, 2-D, LWR core calculations with CASMO-4E. In *Proc. PHYSOR*, 2002.

[52] J. Spanier and E.M. Gelbard. *Monte Carlo Principles and Neutron Transport Problems*. Addison-Wesley Publishing Co., Reading, Massachusettes, 1969.

[53] The HDF Group. Hierarchical data format version 5, 2000-2010. http://www.hdfgroup.org/HDF5.

[54] A. Toth, J.A. Ellis, T.M. Evans, S.P. Hamilton, C.T. Kelley, R. Pawlowski, and S.R. Slattery. Local improvement results for Anderson acceleration with inaccurate function evaluations. *SIAM J. Sci. Comp.*, 39(5):S47–S65, 2017.

[55] A. Toth and C.T. Kelley. Convergence analysis for Anderson acceleration. *SIAM J. Numer. Anal.*, 53:805 – 819, 2015.

[56] H.A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant to Bi-CG for the solution of nonsymmetric systems. *SIAM J. Sci. Statist. Comput.*, 13:631–644, 1992.

[57] J.C. Wagner and A. Haghighat. Automated variance reduction of Monte Carlo shielding calculations using the discrete ordinates adjoint function. *Nuclear Sci. Eng.*, 128:186–208, 1998.

[58] J.C. Wagner and A. Haghighat. Monte Carlo variance reduction with deterministic importance functions. *Prog. Nucl. Energy*, 42:25–53, 2003.

[59] J.C. Wagner and A. Haghighat. Monte Carlo variance reduction with deterministic importance functions. *Prog. Nucl. Energy*, 42:25–53, 2003.

[60] J.C. Wagner, S.W. Mosher, T.M. Evans, D.E. Peplow, and J.A. Turner. Hybrid and parallel domain-decomposition methods development to enable Monte Carlo for reactor analyses. *Progress in Nuclear Sci. and Tech.*, 2:815–820, 2011.

[61] J.C. Wagner, D.E. Peplow, and S.W. Mosher. FW-CADIS method for global and regional variance reduction of Monte Carlo radiation transport calculations. *Nuclear Sci. Eng.*, 176:37–57, 2014.

[62] H.W. Walker and P. Ni. Anderson acceleration for fixed-point iterations. *SIAM J. Numerical Analysis*, 49(4):1715–1735, 2011.

[63] T.A. Wareing, J.M. McGhee, J.E. Morel, and S.D. Pautz. Discontinuous finite element S_N Methods on 3-D unstructured grids. *Nucl. Sci. Eng.*, 138:256–265, 2001.

[64] J.A. Willert, X. Chen, and C.T. Kelley. Newton's Method for Monte Carlo-Based Residuals. *SIAM J. Numer. Anal.*, 53:1738–1757, 2015.

[65] J.A. Willert, C.T. Kelley, D.A. Knoll, and H.K. Park. Hybrid deterministic/Monte Carlo neutronics. *SIAM J. Sci. Comp.*, 35:S62–S83, 2013.

APPENDIX

Appendix A

Auxiliary Material

A.1 Stationary Iterative Methods

As opposed to direct methods, stationary iterative methods [28,40] are used to solve large linear systems of equations in the form

$$\mathbf{A}x = b, \tag{A.1}$$

with $\mathbf{A} \in \mathbb{R}^{n\times n}$ and $x, b \in \mathbb{R}^{n\times 1}$. To form the iterative algorithm, we split $\mathbf{A}$ into nonsingular and singular matrices $\mathbf{M}$ and $\mathbf{N}$, such that $\mathbf{A} = \mathbf{M} - \mathbf{N}$, where $\mathbf{M}^{-1}$ exists. Then,

$$\mathbf{A}x = b, \tag{A.2}$$

$$(\mathbf{M} - \mathbf{N})x = b, \tag{A.3}$$

$$\mathbf{M}x = \mathbf{N}x + b, \tag{A.4}$$

$$x = \mathbf{M}^{-1}\mathbf{N}x + \mathbf{M}^{-1}b. \tag{A.5}$$

We define the iteration from x_i to x_{i+1} as

$$x_{i+1} = \mathbf{M}^{-1}\mathbf{N}x_i + \mathbf{M}^{-1}b, \tag{A.6}$$

with the iteration matrix $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$.

By definition, the spectral radius of a matrix $\mathbf{A}$ is

$$\rho(\mathbf{A}) = \max_{\lambda\in\sigma(\mathbf{A})} |\lambda|, \tag{A.7}$$

where $\sigma(\mathbf{A})$ is the eigenspectrum of $\mathbf{A}$. If $\rho(\mathbf{H}) < 1$, the algorithm converges

$$\lim_{i\to\infty} x_i = x = \mathbf{A}^{-1}b, \quad \forall x_0 \in \mathbb{R}^{n\times 1}, \tag{A.8}$$

where x_0 is the initial iterate. To prove A.8, let $\mathbf{A} = \mathbf{M}(\mathbf{I} - \mathbf{H})$. $\mathbf{I} - \mathbf{H}$ is nonsingular, and

$$(\mathbf{I} - \mathbf{H})^{-1} = \sum_{i=0}^{\infty} \mathbf{H}^i. \tag{A.9}$$

We prove A.9 using a Cauchy sequence of partial sums $\mathbf{S}_k$ in $\mathbb{R}^{n\times n}$, where

$$\mathbf{S}_k = \sum_{i=0}^{k} \mathbf{H}^i. \tag{A.10}$$

Now, WLOG let $j < k$, we see

$$\|\mathbf{S}_k - \mathbf{S}_j\| = \left\| \sum_{i=0}^{k} \mathbf{H}^i - \sum_{i=0}^{j} \mathbf{H}^i \right\|, \tag{A.11}$$

$$= \left\| \sum_{i=j+1}^{k} \mathbf{H}^i \right\|, \tag{A.12}$$

$$\leq \sum_{i=j+1}^{k} \|\mathbf{H}^i\|, \tag{A.13}$$

$$\leq \sum_{i=j+1}^{k} \|\mathbf{H}\|^i, \tag{A.14}$$

$$= \|\mathbf{H}\|^{j+1} \left(\frac{1 - \|\mathbf{H}\|^{k-(j+1)}}{1 - \|\mathbf{H}\|} \right). \tag{A.15}$$

Given $\|\mathbf{H}\| < 1$ and $\left(\frac{1-\|\mathbf{H}\|^{k-(j+1)}}{1-\|\mathbf{H}\|} \right)$ finite, we know $\|\mathbf{S}_k - \mathbf{S}_j\| \to 0$ as $j, k \to \infty$. A convergent Cauchy sequence implies $\exists \mathbf{S} \in \mathbb{R}^{n\times n}$ s.t. $\mathbf{S}_k \to \mathbf{S}$, and we know

$$\mathbf{S} = \mathbf{H}\mathbf{S} + \mathbf{I}, \tag{A.16}$$

$$\mathbf{S}(\mathbf{I} - \mathbf{H}) = \mathbf{I}, \tag{A.17}$$

$$\mathbf{S} = (\mathbf{I} - \mathbf{H})^{-1}. \tag{A.18}$$

The iteration as one step is

$$x_{i+1} = \mathbf{H}^{i+1} x_0 + \sum_{i=0}^{\infty} \mathbf{H}^i \mathbf{M}^{-1} b, \tag{A.19}$$

where $\mathbf{H}^{i+1} x_0 \to 0$, as $i \to \infty$, irrespective of any $x_0 \in \mathbb{R}^{n\times 1}$. We are left with $x_i \to \mathbf{S}\mathbf{M}^{-1} b$, where $\mathbf{S}\mathbf{M}^{-1} b = (\mathbf{I} - \mathbf{H})^{-1}\mathbf{M}^{-1} b = (\mathbf{M}(\mathbf{I} - \mathbf{H}))^{-1} b = \mathbf{A}^{-1} b$.

A.2 GMRES

The General Minimum RESidual (GMRES) is an iterative method for solving linear systems [28,40,49]. On the kth iteration, GMRES minimizes the residual error over an affine space

$$x_0 + \mathcal{K}_k, \tag{A.20}$$

with initial iterate x_0 and the Krylov subspace of residuals, $\mathcal{K}_k$, is

$$\mathcal{K}_k = \text{span}\{r_0, \mathbf{A}r_0, \ldots, \mathbf{A}^{k-1}r_0\}, \tag{A.21}$$

with the initial residual $r_0 \equiv b - Ax_0$. Thus, the minimization for the kth iteration is

$$\min_{x \in x_0 + \mathcal{K}_k} \|b - \mathbf{A}x\|_2, \tag{A.22}$$

and the iteration terminates when the residual is below a user specified tolerance or a maximum number of iterations are performed.

The attractiveness of GMRES comes from its speed given a well-conditioned system. The conditioning of a matrix $\mathbf{A}$ is determined by its *condition number*,

$$\kappa(\mathbf{A}) = \|\mathbf{A}\| \cdot \|\mathbf{A}^{-1}\|. \tag{A.23}$$

It is known [28] that GMRES is "best" suited to diagonalizeable systems. One way of improving the condition number and thus the speed of the GMRES iteration is by *preconditioning* the system. Preconditioning is a strategy that attempts to alleviate conditioning problems by multiplying the system by an approximate inverse of the matrix $\mathbf{A}$. So, for the system

$$\mathbf{A}x = b,$$

we use a precondition matrix $\mathbf{B}^{-1} \approx \mathbf{A}^{-1}$ as

$$\mathbf{B}^{-1}\mathbf{A}x = \mathbf{B}^{-1}b. \tag{A.24}$$

A.24 is the preconditioned system to be solved. This is called *left*-preconditioning because the new matrix $\mathbf{B}^{-1}$ acts on the left of the system. The hope is that $\mathbf{B}^{-1}\mathbf{A} \approx \mathbf{I}$ so that eigenvalues of the new system are clustered around one. For *Right*-preconditioning, we solve the two systems

$$\mathbf{A}\mathbf{B}^{-1}y = b, \tag{A.25}$$

$$\mathbf{B}^{-1}y = x, \tag{A.26}$$

for the unknown x. One advantage to right-preconditioning is that GMRES minimizes the original residual $\|b - \mathbf{A}x_k\|_2$, whereas left-preconditioning minimizes the preconditioned residual $\|\mathbf{B}^{-1}(b - \mathbf{A}x_k)\|_2$. If the user hopes to reduce the original residual down to a certain tolerance, then left-preconditioning needs to be used cautiously knowing that the preconditioned termination metric is not the same as the original termination metric.

Computer tools which perform the GMRES iterations for the system do not need a matrix $\mathbf{A}$ or preconditioner $\mathbf{B}^{-1}$ in explicit form. In practice, a function which accepts as input as vector and returns the matrix-vector product, or *matvec*, is often easier to code and preferable to storing a large, possibly dense, matrix. So, for an input vector u, the output of the matvec function would be a vector equal to $\mathbf{A}u$. Working with codes and solvers in this *matrix-free* manner is good practice and leads to convenient, component-oriented code.

Other notable methods are BiCGSTAB [56], TFQMR [18], and Conjugate Gradient [25]. One of the main drawbacks of GMRES is storage. The algorithm requires storage of the vectors in the space $\mathcal{K}_k$, appending one vector each iteration. Depending on the computing environment and the size of your system, memory constraints may fail the job.

There are strategies to prevent the storage burden from becoming overwhelming for the environment– one being GMRES with restarts, or GMRES(m). GMRES(m) will run for m iterations storing each necessary vector for the Krylov subspace, and then the iteration halts if the problem has not reduced the residual below the user specified tolerance. The solver will discard all past vectors in $\mathcal{K}_k$ and began again with the most recent iterate. This continues until the residual is reduced or the maximum iterations allowed for the whole calculation is reached.

A.3 List of Shift Code Base Contributions

The Exnihilo code suite provides a parallel, component library for transport application development on high performance computing platforms. The Exnihilo code suite contains both the Shift Monte Carlo solver [46] and the Denovo deterministic solver [16]. It contains pre- and post-processing tools for users integrated with Jupyter notebooks. Exnihilo leverages existing functionality from other libraries, such as SCALE [1], Trilinos [24], and HDF5 [53]. It uses an Oak Ridge National Laboratory internal GitLab code repository and issue tracking system for adding new features and fixing bugs. Exnihilo uses the design-by-contract tool, GoogleTest [20], to define interface specifications for software components. Exnihilo is developed using an Agile, Continuous-Integration workflow, which includes roughly 1200 unit tests checked on each merge. It also has acceptance and runtime performance tests that run weekly. Table A.1 shows the language makeup and the lines of code in Exnihilo.

Language	Executable	Test
C++	274,648	279,170
Python	30,543	19,569
CUDA	12,356	8,804
C	1,555	—
Fortran	934	55

Table A.1: Shift's programming language composition and lines of code.

Here is a list of all major contributions I have made to the Shift Monte Carlo code base as part of this thesis work. The two main areas have been, first, in the implementation of Shift's nonuniform processor allocation strategy and its optimization, and second, in finishing the implementation of the domain decomposed CADIS algorithm. All contributions have met the coding and documentation standards of the Exnihilo development team.

Load Balancing:

- `Transcore/mc/Boundary_Mesh_Nonuniform`: Developed, implemented, tested the nonuniform processor allocation strategy
- `Shift/mc_transport/detail/Communicator_Nonuniform`: Developed, implemented, tested an improved communication pattern for nonuniform processor allocations
- `Shift/mc_transport/DD_Sync_Source_Transporter, Omnibus/driver/detail/Processor_Allocator`: Developed, implemented, tested the diagnostic-based optimization algorithm
- `Omnibus/driver/Driver, Omnibus/python/omnibus/omn/shift.py`: Added user access to new features and program drivers for auto-calibration mode

Domain Decomposed Hybrid Calculations:

- Helped to complete domain decomposed hybrid implementation
  - `Shift/mc_sources/Separable_Biased_Source, Shift/mc_sources/Fixed_Sourcer_Base`: Fixed creation of biased sources in domain decomposition and nonuniform processor allocation mode
  - `Shift/mc_hybrid/Parallel_Grid`: Streamlined weight window data movement

  - `Shift/mc_hybrid/Grid_LG_Indexer`: Developed 3D local to global mesh indexer for weight windows

- `Nemesis/utils/Rebalancer`: Developed templated rebalance algorithms to rebalance particles across subblocks and sets during particle storm formation
- `Shift/mc_sources/detail/Source_Partitioner_Local`: Improved domain decomposed sourcing performance through partitioning (currently only Box shape sources)