MASSIVELY PARALLEL MONTE CARLO METHODS FOR DISCRETE LINEAR SYSTEMS

by

Stuart R. Slattery

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Nuclear Engineering and Engineering Physics)

at the

UNIVERSITY OF WISCONSIN–MADISON

31 May 2013

Acknowledgments

Great thanks are owed to my advisor and friend Paul Wilson for providing me years of guidance and the opportunity to develop this work. Without him, this work would have never been possible.

Tom Evans is responsible for presenting me with the seeds of this work and for that I thank him. His mentoring over the past few years has been invaluable.

Roger Pawlowski and other members of the Consortium for Advanced Simulation of Light Water Reactors and Trilinos teams as well as many staff members at Oak Ridge National Laboratory have provided tremendous resources for my professional and technical development and have greatly facilitated this work.

This work was performed under appointment to the Nuclear Regulatory Commission Fellowship program at the University of Wisconsin - Madison Department of Engineering Physics.

Contents

List of Figures

MASSIVELY PARALLEL MONTE CARLO METHODS FOR DISCRETE LINEAR SYSTEMS

Stuart R. Slattery

Under the supervision of Professor Paul P.H. Wilson
At the University of Wisconsin-Madison

Paul P.H. Wilson

Chapter 1

Introduction

In many fields of engineering and physics, linear and nonlinear problems are a primary focus of study. Recent focus on multiple physics systems in the nuclear reactor modeling and simulation community adds a new level of complexity to common nonlinear systems as solution strategies change when they are coupled to other problems (U.S. Department of Energy, 2011). Furthermore, a desire for predictive simulations to enhance the safety and performance of nuclear systems creates a need for extremely high fidelity computations to be performed for these coupled systems as a means to capture effects not modeled by coarser methods.

In order to achieve this high fidelity, state-of-the-art computing must be leveraged in a way that is both efficient and considerate of hardware-related issues. As scientific computing moves towards exascale facilities with machines of $O(1,000,000)$ cores already coming on-line, new algorithms to solve these complex problems must be developed to leverage this new hardware (Kogge and Dysart, 2011). Issues such as resiliency to node failure, limited growth of memory available per node, and scaling to large numbers of cores will be pertinent to robust algorithms aimed at this new hardware. Considering these issues, this dissertation develops a massively parallel Monte Carlo method for linear problems and a novel Monte Carlo method to advance solution techniques for nonlinear problems.

We discuss in this chapter motivation for advancing Monte Carlo solvers by providing multiphysics problems of interest in nuclear reactor analysis. Hardware-based motivations are also provided by considering the impact of forthcoming computing architectures. In addition, background on the current solver techniques for multiphysics problems and a brief comparison to the proposed methods is provided to further motivate this work.

1.1 Physics-Based Motivation

Predictive modeling and simulation capability requires the combination of high fidelity models, high performance computing hardware that can handle the intense computational loads required by these models, and modern algorithms for solving these problems that leverage this high performance hardware. For nuclear reactor analysis,

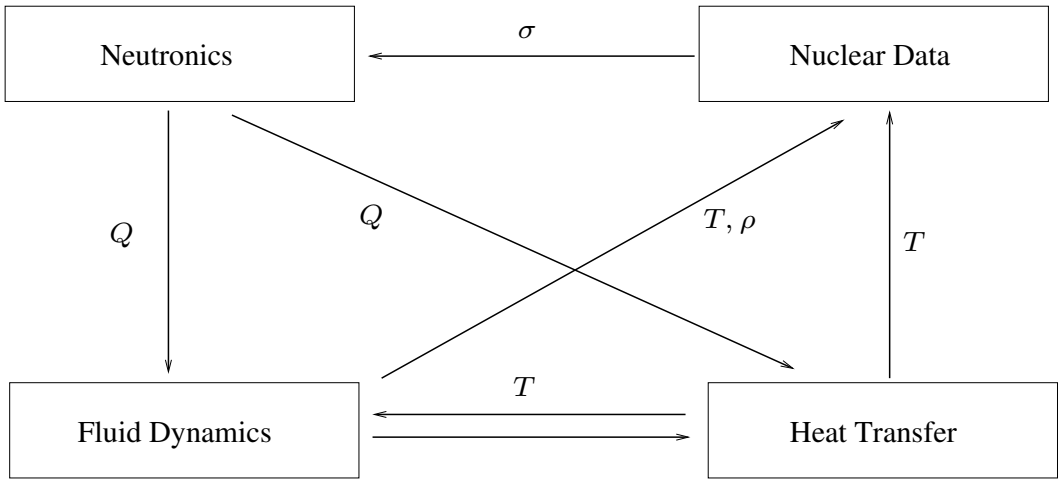

Figure 1.1: **Multiphysics dependency analysis of departure from nucleate boiling.** *A neutronics solution is required to compute power generation in the fuel pins, fluid dynamics is required to characterize boiling and fluid temperature and density, heat transfer is required to compute the fuel and cladding temperature, and the nuclear data modified with the temperature and density data. Strong coupling among the variables creates strong nonlinearities.*

this predictive capability can enable tighter design tolerances for improved thermal performance and efficiency, higher fuel burn-up and therefore reduction in generated waste, and high confidence in accident scenario models. The physics that dominate these types of analysis include neutronics, thermal hydraulics, computational fluid dynamics, and structural mechanics.

Although solution techniques in each of these individual categories has advanced over the last few decades and in fact leveraged modern algorithms and computer architectures, true predictive capability for engineered systems can only be achieved through a coupled, multiple physics analysis where the effects of feedback between physics are modeled. For example, consider the safety analysis of a departure from nucleate boiling scenario in the subchannel of a nuclear fuel assembly. When this event occurs, heat transfer is greatly reduced between the fuel and the coolant due to the vapor layer generated by boiling, causing the fuel center-line temperature to rapidly rise. To characterize this boiling phenomena and how it affects fuel failure we must consider a neutronics analysis in order to compute power generation in the fuel pins, fluid dynamics analysis to characterize coolant boiling, temperature, and density, solid material heat transfer to characterize fuel and cladding temperature and heat transfer with the coolant, and nuclear data processing to characterize how

changing material temperatures and densities changes the cross sections needed for the neutronics calculation. As shown in Figure 1.1, many couplings are required among individual physics components in order to accurately model this situation with each physics generating and receiving many responses. Those variables that are very tightly coupled, such as the temperatures generated by the fluid dynamics and heat transfer components, will have strong nonlinearities in their behavior and would therefore benefit from fully consistent nonlinear solution schemes instead of fixed-point type iterations between physics[1]. Furthermore, the space and time scales over which these effects occur will also vary greatly.

The computational resources required to solve such problems are tremendous. Recent work in modeling coupled fluid flow and solid material heat and mass transfer in a reactor subsystem, similar to the same components of the departure from nucleate boiling example above, was performed as part of analysis of the Department of Energy's Consortium for Advanced Simulation of Light Water Reactors (CASL) modeling and simulation hub. CASL used the Drekar multiphysics code developed at Sandia National Laboratories (Pawlowski et al., 2012) for modeling goals in grid-to-rod-fretting analysis and will use a similar coupled physics structure for future departure from nucleate boiling analysis with comparison to experimental data. Using Drekar, multiphysics simulations have been performed with fully consistent methods for the solution of nonlinear systems using meshes of $O(1 \times 10^9)$ elements leveraging $O(100,000)$ cores on leadership class machines. Neutronics components to be implemented in CASL for multiphysics analysis, such as the Exnihilo radiation transport suite developed at Oak Ridge National Laboratory (Evans et al., 2010), compute trillions of unknowns for full core reactor analysis on $O(1 \times 10^9)$ element meshes and $O(100,000)$ cores as well. Given the large scale and complexity of these problems, if we aim to advance multiphysics solution techniques, then we are motivated to advance the solution of complex and general nonlinear problems exploiting leadership class levels of parallelism.

1.2 Hardware-Based Motivation

As leadership class machines move towards the exascale, new algorithms must be developed that leverage their strengths and adapt to their shortcomings. Basic research is required now to advance methods in time for these new machines to become operational. Organized work is already moving forward in this area with the Department of Energy's Advanced Scientific Computing Research office specifically

[1]Fixed-point iterations between physics are commonly referred to as Picard iterations.

allocating funding for the next several years to research resilient solver technologies for exascale facilities (U.S. Department of Energy, 2012). Based on the language in this call for proposals, we can identify key issues for which a set of robust, massively parallel Monte Carlo solvers could provide a solution. As machines begin to operate at hundreds of petaflops peak performance and beyond, trends toward reduced energy consumption will require incredibly high levels of concurrency to achieve the desired computation rates. Furthermore, this drop in power consumption will mean increased pressure on memory as memory per node is expected to stagnate while cores per node is expected to increase. As the number of cores increases, their clock speed is expected to stagnate or even decrease to further reduce power consumption and manufacturing costs.

The end result of these hardware changes is that the larger numbers of low-powered processors will be prone to both soft failures such as bit errors in floating point operations and hard failures where the data owned by that processor cannot be recovered. Because these failures are predicted to be common, resilient solver technologies are required to overcome these events. With linear and nonlinear solvers based on Monte Carlo techniques, such issues are alleviated by statistical arguments. In the case of soft failures, isolated floating point errors in Monte Carlo simulation are absorbed within tally statistics while completely losing hardware during a hard failure is manifested as a high variance event where some portion of the Monte Carlo histories are lost. These stochastic methods are a paradigm shift from current deterministic solver techniques that will suffer greatly from the non-deterministic behavior expected from these exascale machines.

In addition to resiliency concerns, the memory restrictions on future hardware will hinder modern solvers that derive their robustness from using large amounts of memory. Stochastic methods that are formulated to use less memory than conventional methods will serve to alleviate some of this pressure. In addition, new parallel strategies that may be implemented with stochastic methods could offer a new avenue for leveraging the expected levels of high concurrency in exascale machines.

1.3 Research Outline

For some time, the particle transport community has been utilizing Monte Carlo methods for the solution of transport problems (Lewis, 1993). The partial differential equation (PDE) community has focused on various deterministic methods for solutions to linear problems (Saad, 2003; Kelley, 1995). In between these two areas are a not

widely known group of Monte Carlo methods for solving sparse linear systems (Forsythe and Leibler, 1950; Hammersley and Handscomb, 1964; Halton, 1962, 1994). In recent years, these methods have been further developed for radiation transport problems in the form of Monte Carlo Synthetic-Acceleration (MCSA) (Evans and Mosher, 2009; Evans et al., 2012) but have yet to be applied to more general sparse linear systems commonly generated by the computational physics community. Compared to other methods in this regime, MCSA offers three attractive qualities; (1) the linear problem operator need not be symmetric or positive-definite, thereby reducing preconditioning complexity, (2) the stochastic nature of the solution method provides a natural solution to the issue of resiliency, and (3) is amenable to parallelization using modern methods developed by the transport community (Wagner et al., 2010). The development of MCSA as a general linear solver and the development of a parallel MCSA method will be new and unique features of this work, providing a framework with which other issues such as resiliency may be addressed in the future.

In addition to linear solver advancements, nonlinear solvers may also benefit from a general and parallel MCSA scheme. In the nuclear engineering community, nonlinear problems are often addressed by either linearizing the problem or building a segregated scheme and using traditionally iterative or direct methods to solve the resulting system (Pletcher et al., 1997). In the mathematics community, various Newton methods have been popular (Kelley, 1995). Recently, Jacobian-Free Newton-Krylov (JFNK) schemes (Knoll and Keyes, 2004) have been utilized in multiple physics architectures and advanced single physics codes (Gaston et al., 2009). The benefits of JFNK schemes are that the Jacobian is never formed, simplifying the implementation, and a Krylov solver is leveraged (typically GMRES or Conjugate Gradient), providing excellent convergence properties for well-conditioned and well-scaled systems. However, there are two potential drawbacks to these methods for high fidelity predictive simulations: (1) the Jacobian is approximated by a first-order differencing method on the order of machine precision such that this error can grow beyond that of those in a fine-grained system (Kelley, 1995) and (2) for systems that are not symmetric positive-definite (which will be the case for most multiphysics systems and certainly for most preconditioned systems) the Krylov subspace generated by the GMRES solver may become prohibitively large (Knoll and McHugh, 1995). To address these issues, this thesis develops a new and novel method for nonlinear systems based on the MCSA method.

The Forward-Automated Newton-MCSA (FANM) method is developed as new nonlinear solution method. The key features of FANM are: full Jacobian generation

using modern Forward Automated Differentiation (FAD) methods (Bartlett et al., 2006), and MCSA as the inner linear solver. This method has several attractive properties. First, the first-order approximation to the Jacobian used in JFNK type methods is eliminated by generating the Jacobian explicitly with the model equations through FAD. Second, the Jacobian need not be explicitly formed by the user but is instead automated through FAD; this eliminates the complexity of hand-coding derivatives and has also been demonstrated to be more efficient computationally than evaluating difference derivatives. Third, unlike GMRES, MCSA does not build a subspace during iterations. Although the Jacobian must be explicitly formed to use MCSA, for problems that take more than a few GMRES iterations to converge the size of the Krylov subspace will grow beyond that of the Jacobian. Finally, using MCSA for the linear solve provides its benefits for preconditioning, potential resiliency, and parallelism.

This preliminary report outlines in Chapter 2 the conventional methods used in practice for solving linear problems to provide a mathematical basis upon which to build new algorithms that aim to solve some of the aforementioned issues. Parallel schemes for conventional methods are also provided for background and understanding of how current methods may or may not map to future hardware. In addition, some components of their parallel implementations may be applied to stochastic methods development. In Chapter 3, Monte Carlo algorithms for solving linear systems and new parallel strategies will be outlined in full with links made to past work and their potential for offering improvements to the multiphysics analysis community. From these stochastic methods, a new nonlinear method is developed in Chapter 5 and compared to conventional methods for solving nonlinear problems.

Chapter 2

Conventional Solution Methods for Linear Systems

The discretization of partial differential equations (*PDEs*) through common methods such as finite differences (LeVeque, 2007), finite volumes (LeVeque, 2002), and finite elements (Zienkiewicz et al., 2005) ultimately generates sets of coupled equations in the form of matrix problems. In many cases, these matrices are sparse, meaning that the vast majority of their constituent elements are zero. This sparsity is due to the fact that the influence of a particular grid element only expands as far as a few of its nearest neighbors depending on the order of discretization used and therefore coupling among variables in a particular discrete equation in the system leads to a few non-zero entries. Because of the natural occurrence of sparse matrices in common numerical methods many iterative techniques have been developed to solve such systems. We discuss here conventional stationary and projection methods for solving sparse systems to provide the necessary background for the remainder of this work. Details on the parallelization of conventional methods are discussed.[1]

2.1 Preliminaries

We seek solutions of the general linear problem in the following form:

$$\mathbf{A}\mathbf{x} = \mathbf{b}\,, \tag{2.1}$$

where $\mathbf{A} \in \mathbb{R}^{N\times N}$ is a matrix operator such that $\mathbf{A} : \mathbb{R}^N \to \mathbb{R}^N$, $\mathbf{x} \in \mathbb{R}^N$ is the solution vector, and $\mathbf{b} \in \mathbb{R}^N$ is the forcing term. The solutions to Eq (2.1) will be generated by inverting $\mathbf{A}$ either directly or indirectly:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}\,. \tag{2.2}$$

[1]The contents of this chapter, particularly those sections relating to projection methods and matrix analysis, are heavily based on Saad's text (Saad, 2003).

In addition we can define the residual:

$$\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}\,, \tag{2.3}$$

such that an exact solution $\mathbf{x}$ has been found when $\mathbf{r} = \mathbf{0}$. From the statement in Eq (2.2) we can already place a restriction on $\mathbf{A}$ by requiring that it be *nonsingular*, meaning that we can in fact compute $\mathbf{A}^{-1}$. In this work we will focus our efforts on approximately inverting the operator through various means.

In a discussion of methods for solving linear systems, several mathematical tools are useful in characterizing the qualities of the linear system. Among the most useful are the *Eigenvalues* of the matrix, $\sigma(\mathbf{A})$. We find these by solving the Eigenvalue problem:

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x},\ \lambda \in \sigma(\mathbf{A})\,. \tag{2.4}$$

By writing Eq (2.4) in a different form,

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0\,, \tag{2.5}$$

and demanding that non-trivial solutions for $\mathbf{x}$ exist, it is then required that $|\mathbf{A} - \lambda\mathbf{I}| = 0$. Expanding this determinant yields a characteristic polynomial in terms of λ with roots that form the set of Eigenvalues, $\sigma(\mathbf{A})$. Each component of $\sigma(\mathbf{A})$ can then be used to solve Eq (2.5) for a particular permutation of $\mathbf{x}$. The set of all permutations form the *Eigenvectors* of $\mathbf{A}$. A quantity of particular interest that is computable from the eigenvalues of a matrix $\mathbf{A}$ is the *spectral radius*, $\rho(\mathbf{A})$, defined by Saad (Saad, 2003) as:

$$\rho(\mathbf{A}) = \max_{\lambda\in\sigma(\mathbf{A})} |\lambda|\,. \tag{2.6}$$

In addition, for problems that have a large scale over which the independent variables may exist (e.g. a problem with events on timescales ranging from nanoseconds to hours), a good measure of this range is supplied by the *stiffness ratio*:

$$StiffnessRatio = \frac{\max_{\lambda\in\sigma(\mathbf{A})} |\lambda|}{\min_{\lambda\in\sigma(\mathbf{A})} |\lambda|} \tag{2.7}$$

Those problems that have a wide range of scales in their independent variables, which will then be reflected in the operator, will then have a large stiffness ratio. We will define such problems with large stiffness ratios as *stiff*.

General to both matrices and vectors, *norms* are a mechanism for collapsing objects of many elements to a single value. Per LeVeque's text (LeVeque, 2007), the q-norm

of a vector is defined as:

$$||\mathbf{v}||_q = \left[\sum_{i=1}^{N} |v_i|^q \right]^{1/q}, \ \mathbf{v} \in \mathbb{R}^N, \ q \in \mathbb{Z}^+ \tag{2.8}$$

where v_i is the i^{th} component of the vector. Depending on the value chosen for q, local or global qualities of the vector may be obtained. For example, $q = 2$ provides the root of a quadrature sum of all elements in the vector giving a global measure of the vector while $q = \infty$ gives the maximum value in the vector, a local quantity that does not give information regarding the other elements in the vector.

We can also compute the norm of a matrix by inferring from the norm of the vector on which it is operating. Per LeVeque, we search for a constant that is equivalent to $||\mathbf{A}||$:

$$||\mathbf{Ax}|| \leq C||\mathbf{x}|| \,, \tag{2.9}$$

where the minimum value of C that satisfies Eq (2.9) is equivalent to $||\mathbf{A}||$ and is valid $\forall \mathbf{x} \in \mathbb{R}^N$. The general definition in Eq (2.9) can be expanded in simple terms for common norms including the infinity norm:

$$||\mathbf{A}||_\infty = \max_{1 \leq i \leq N} \sum_{j=1}^{N} |a_{ij}| \,, \tag{2.10}$$

and the 2-norm:

$$||\mathbf{A}||_2 = \sqrt{\rho(\mathbf{A}^T\mathbf{A})} \,, \tag{2.11}$$

where ρ is the spectral radius as defined in Eq (2.6).

Knowing this, we can then define several useful properties of matrices including the *condition number*:

$$\kappa(\mathbf{A}) = ||\mathbf{A}|| \ ||\mathbf{A}^{-1}|| \,, \tag{2.12}$$

which gives as a metric on assessing how close to singular the system is. This is due to the fact $||\mathbf{A}^{-1}||$ is large near singularities (and undefined for a singular matrix) and thus a large condition number will be generated. We define such matrices as *ill-conditioned*.

2.2 Stationary Methods

Stationary methods for linear systems arise from splitting the operator in Eq (2.1):

$$\mathbf{A} = \mathbf{M} - \mathbf{N} \,, \tag{2.13}$$

where the choice of $\mathbf{M}$ and $\mathbf{N}$ will be dictated by the particular method chosen. Using this split definition of the operator we can then write:

$$\mathbf{M}\mathbf{x} - \mathbf{N}\mathbf{x} = \mathbf{b} \,. \tag{2.14}$$

By rearranging, we can generate a form more useful for analysis:

$$\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{c} \,, \tag{2.15}$$

where $\mathbf{H} = \mathbf{M}^{-1}\mathbf{N}$ is defined as the *iteration matrix* and $\mathbf{c} = \mathbf{M}^{-1}\mathbf{b}$. With the solution vector on both the left and right hand sides, an iterative method can then be formed:

$$\mathbf{x}^{k+1} = \mathbf{H}\mathbf{x}^{k} + \mathbf{c} \,, \tag{2.16}$$

with $k \in \mathbb{Z}^{+}$ defined as the *iteration index*. In general, we will define methods in the form of Eq (2.16) as *stationary methods*. Given this, we can then generate a few statements regarding the convergence of such stationary methods. Defining $\mathbf{e}^{k} = \mathbf{u}^{k} - \mathbf{u}$ as the solution error at the k^{th} iterate, we can subtract Eq (2.15) from Eq (2.16) to arrive at an error form of the linear problem:

$$\mathbf{e}^{k+1} = \mathbf{H}\mathbf{e}^{k} \,. \tag{2.17}$$

Our error after k iterations is then:

$$\mathbf{e}^{k} = \mathbf{H}^{k}\mathbf{e}^{0} \,. \tag{2.18}$$

In other words, successive application of the iteration matrix is the mechanism driving down the error in a stationary method. We can then place restrictions on the iteration matrix by using the tools developed in § (2.1). By assuming $\mathbf{H}$ is diagonalizable[2] (Saad, 2003), we then have:

$$\mathbf{e}^{k} = \mathbf{R}\mathbf{\Lambda}^{k}\mathbf{R}^{-1}\mathbf{e}^{0} \,, \tag{2.19}$$

[2]We may generalize this to non-diagonalizable matrices with the Jordan canonical form of $\mathbf{H}$.

where $\mathbf{\Lambda}$ contains the Eigenvalues of $\mathbf{H}$ on its diagonal and the columns of $\mathbf{R}$ contain the Eigenvectors of $\mathbf{H}$. Computing the 2-norm of the above form then gives:

$$||\mathbf{e}^k||_2 \leq ||\mathbf{\Lambda}^k||_2 \; ||\mathbf{R}||_2 \; ||\mathbf{R}^{-1}||_2 \; ||\mathbf{e}^0||_2 \, , \tag{2.20}$$

which gives:

$$||\mathbf{e}^k||_2 \leq \rho(\mathbf{H})^k \kappa(\mathbf{R}) ||\mathbf{e}^0||_2 \, . \tag{2.21}$$

For iteration matrices where the Eigenvectors are orthogonal, $\kappa(\mathbf{R}) = 1$ and the error bound reduces to:

$$||\mathbf{e}^k||_2 \leq \rho(\mathbf{H})^k ||\mathbf{e}^0||_2 \, . \tag{2.22}$$

We can now restrict $\mathbf{H}$ by asserting that $\rho(\mathbf{H}) < 1$ for a stationary method to converge such that k applications of the iteration matrix will not cause the error to grow in Eq (2.22).

2.3 Projection Methods

Among the most common iterative methods used in scientific computing today for sparse systems are of a broad class known as *projection methods*. These methods not only provide access to more powerful means of reaching a solution, but also a powerful means of encapsulating the majority of common iterative methods including the stationary methods just discussed in a common mathematical framework. All projection methods are built around a core structure where the solution to Eq (2.1) is extracted from a *search subspace* $\mathcal{K}$ and bound by a *constraint subspace* $\mathcal{L}$ that will vary in definition depending on the iterative method selected. We build the approximate solution $\tilde{\mathbf{x}}$ by starting with an initial guess $\mathbf{x}_0$ and extracting a correction $\boldsymbol{\delta}$ from $\mathcal{K}$ such that:

$$\tilde{\mathbf{x}} = \mathbf{x}_0 + \boldsymbol{\delta}, \;\; \boldsymbol{\delta} \in \mathcal{K} \, . \tag{2.23}$$

We bound this correction by asserting that the new residual, $\tilde{\mathbf{r}}$, be orthogonal to $\mathcal{L}$:

$$\langle \tilde{\mathbf{r}}, \mathbf{w} \rangle = 0, \;\; \forall \mathbf{w} \in \mathcal{L} \, . \tag{2.24}$$

We can generate a more physical and geometric-based understanding of these constraints by writing the new residual as $\tilde{\mathbf{r}} = \mathbf{r}_0 - \mathbf{A}\boldsymbol{\delta}$ and again asserting the residual must be orthogonal to $\mathcal{L}$. If $\tilde{\mathbf{r}}$ is to be orthogonal to $\mathcal{L}$, then $\mathbf{A}\boldsymbol{\delta}$ must be the projection of $\mathbf{r}_0$ onto the subspace $\mathcal{L}$ that eliminates the components of the residual that exist in

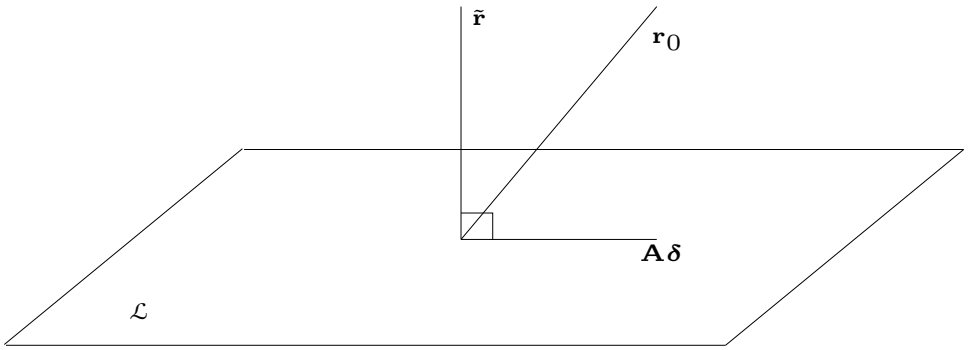

Figure 2.1: **Orthogonality constraint of the new residual with respect to $\mathcal{L}$.** *By projecting $\mathbf{r}_0$ onto the constraint subspace, we minimize the new residual by removing those components.*

$\mathcal{L}$. This situation is geometrically presented in Figure 2.1.

From Figure 2.1 we then note that the following geometric condition must hold:

$$||\tilde{\mathbf{r}}||_2 \leq ||\mathbf{r}_0||_2, \ \forall \mathbf{r}_0 \in \mathbb{R}^N \, , \tag{2.25}$$

meaning that the residual of the system will always be *minimized* with respect to the constraints. Given this minimization condition for the residual, we can form the outline of an iterative projection method. Consider a matrix $\mathbf{V}$ to form a basis of $\mathcal{K}$ and a matrix $\mathbf{W}$ to form a basis of $\mathcal{L}$. As $\boldsymbol{\delta} \in \mathcal{K}$ by definition in Eq (2.23), then $\boldsymbol{\delta}$ can instead be rewritten as:

$$\boldsymbol{\delta} = \mathbf{V}\mathbf{y}, \ \forall \mathbf{y} \in \mathbb{R}^N : \tag{2.26}$$

where $\mathbf{V}$ *projects* $\mathbf{y}$ onto $\mathcal{K}$. From the orthogonality constraint in Eq (2.24) it then follows that:

$$\mathbf{y} = (\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}_0 \, , \tag{2.27}$$

where here the projection onto $\mathcal{K}$ is constrained by the projection onto $\mathcal{L}$. Knowing this, we can then outline the following iteration scheme for a projection method:

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k \, , \tag{2.28a}$$

$$\mathbf{y}^k = (\mathbf{W}^T\mathbf{A}\mathbf{V})^{-1}\mathbf{W}^T\mathbf{r}^k \, , \tag{2.28b}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \mathbf{V}\mathbf{y}^k \, , \tag{2.28c}$$

where $\mathbf{V}$ and $\mathbf{W}$ are generated from the definitions of $\mathcal{K}$ and $\mathcal{L}$ and are updated prior

to each iteration.

From an iteration standpoint, as we choose $\boldsymbol{\delta}$ from $\mathcal{K}$ and constrain it with $\mathcal{L}$, each iteration performs a projection that systematically annihilates the components of the residual that exists in $\mathcal{L}$. This then means that if our convergence criteria for an iterative method is bound to the residual of the system, then Eq (2.25) tells us that each projection step guarantees us that the norm of the new residual will never be worse than that of the previous step and will typically move us towards convergence. Depending on the qualities of the system in Eq (2.1), the selection of the subspaces $\mathcal{K}$ and $\mathcal{L}$ can serve to both guarantee convergence and optimize the rate at which the residual is decreased.

Krylov Subspace Methods

Among the most common projection techniques used in practice are a class of methods known as *Krylov subspace methods*. Here, the search subspace is defined as the *Krylov subspace*:

$$\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0) = span\{\mathbf{r}_0, \mathbf{A}\mathbf{r}_0, \mathbf{A}^2\mathbf{r}_0, \ldots, \mathbf{A}^{m-1}\mathbf{r}_0\} \,, \tag{2.29}$$

where m denotes the dimensionality of the subspace. In order to accommodate a more general structure for the operator in Eq (2.1), we often choose an *oblique* projection method where $\mathcal{K} \neq \mathcal{L}$. If we choose $\mathcal{L} = \mathbf{A}\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$, then we are ultimately solving the normal system $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$ where $\mathbf{A}^T\mathbf{A}$ will be symmetric positive definite if $\mathbf{A}$ is nonsingular, thereby expanding the range of operators over which these methods are valid. This choice of constraint subspace also then gives us the result via Eq (2.24) that the residual is minimized for all $\boldsymbol{\delta} \in \mathcal{K}$, forming the basis for the *generalized minimum residual method* (GMRES) (Saad and Schultz, 1986).

Choosing GMRES as our model Krylov method, we are first tasked with finding a projector onto the subspace. We seek an orthonormal basis for $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0)$ by an orthogonalization procedure that is commonly based on, but not limited to, the *Arnoldi* recurrence relation. The Arnoldi procedure will generate an orthonormal basis, $\mathbf{V}_m \in \mathbb{R}^{N\times m}$, via a variant of the Gram-Schmidt procedure that re-applies the operator for each consecutive vector, thus forming a basis that spans the subspace in Eq (2.29). Due to its equivalent dimensionality, m, to that of the subspace, we will refer to such recurrence relations as *long recurrence relations*. Those orthogonal projection procedures that have a dimensionality less than m will be referred to as *short recurrence relations*. Once $\mathbf{V}_m$ is found, per the constraint subspace definition it then follows that its basis is defined as $\mathbf{W}_m = \mathbf{A}\mathbf{V}_m$. Knowing the projections onto

Algorithm 2.1 GMRES Iteration

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\beta := ||\mathbf{r}_0||_2$
$\mathbf{v}_1 := \mathbf{r}_0/\beta$ ▷ Create the orthonormal basis for the Krylov subspace
for $j = 1, 2, \cdots, m$ **do**
    $h_{ij} \leftarrow \langle w_j, v_j \rangle$
    $w_j \leftarrow w_j - h_{ij} v_i$
end for
$h_{j+1,j} \leftarrow ||w_j||_2$
$v_{j+1} \leftarrow w_j / h_{j+1,j}$ ▷ Apply the orthogonality constraints
$\mathbf{y}_m \leftarrow argmin_y ||\beta \mathbf{e}_1 - \mathbf{H}_m \mathbf{y}||_2$
$\mathbf{x}_m \leftarrow \mathbf{x}_0 + \mathbf{V_m} \mathbf{y}_m$

the search and constraint subspaces, the GMRES iteration may be formulated as follows:

We note here several properties of this formulation and how they may facilitate or hinder the solution of large-scale, sparse linear problems, also noting that these properties are common among many Krylov methods. First, from a memory perspective GMRES is efficient in that the operator $\mathbf{A}$ need not be explicitly stored. Rather, only the ability to compute the action of that operator on a vector of valid size is required. However, these savings in memory are balanced by the fact that the long recurrence relations used in the Arnoldi procedure require all vectors that span the Krylov space to be stored. If the size of these vectors becomes prohibitive, the Arnoldi procedure can be restarted at the cost of losing information in the orthogonalization process, creating the potential to generate new search directions that are not orthogonal to all previous search directions (and therefore less than optimal). From an implementation perspective, because the operator is not required to be formed, GMRES is significantly more flexible in its usage in that there are many instances where various processes serve to provide the action of that operator (e.g. radiation transport sweeps (Evans et al., 2010)) that normally may not be amenable to its full construction. In addition, the minimization problem is a straight-forward least-squares problem where $\mathbf{H}$ is an upper-Hessenberg matrix.

2.4 Parallel Projection Methods

Modern parallel implementations of projection methods on distributed memory architectures rely heavily on capabilities provided by general linear algebra frameworks. For methods like GMRES, this arises from the fact that Krylov methods require only a

handful of operation types in their implementation that can be efficiently programmed on these architectures. Per Saad's text (Saad, 2003) and as noted in Algorithm 2.1, these operations are preconditioning, matrix-vector multiplications, vector updates, and inner products. For the last three items, linear algebra libraries such as PETSc (Gropp and Smith, 1993) and Trilinos (Heroux et al., 2005) provide efficient parallel implementations for these operations. Depending on the type of preconditioning used, efficient parallel implementations may also be available for those operations. Due to their prevalence in modern numerical methods, parallel formulations these operations have warranted intense study (Tuminaro et al., 1998). In all cases, a series of scatter/-gather operations are required such that global communication operations must occur. Although the relative performance of such operations is bound to the implementation, asymptotically performance should be the same across all implementations.

We will look at the three primary parallel matrix/vector operations as preconditioning is not an immediate requirement for implementing the algorithms. We note here that variants are available that reduce the number of global communications required (consider (Sosonkina et al., 1998) as an example of reducing global operation counts using a different orthogonalization procedure than Arnoldi), however, we will only consider the basic algorithms here as this handful of operations can be generalized to fit more complicated algorithms. In all of these cases, we assume a general matrix/vector formulation that is distributed in parallel such that both local and global knowledge of their decomposition is available on request. Furthermore, it is assumed that these objects are partitioned in such a way that the parallel formulation of the operator and vectors in Eq (2.1) will be such that each parallel process contains only a subset of the global problem and that subset forms a local set of complete equations. The form of this partitioning is problem dependent and often has a geometric or graph-based aspect to its construction in order to optimize communication patterns. Libraries such as Zoltan (Devine et al., 2002), provide implementations of such algorithms.

Parallel Vector Update

Parallel vector update operations arise from the construction of the orthonormal basis and the application of the correction generated by the constraints to the solution vector. Vector update operations are embarrassingly parallel in that they require no communication operations to be successfully completed; all data operated on is local. These operations are globally of the form:

$$\mathbf{y}[n] \leftarrow \mathbf{y}[n] + a * \mathbf{x}[n], \ \forall n \in [1, N_g] \,, \tag{2.30}$$

and locally of the form:

$$\mathbf{y}[n] \leftarrow \mathbf{y}[n] + a * \mathbf{x}[n],\ \forall n \in [1, N_l]\ , \tag{2.31}$$

where $\mathbf{y}$ and $\mathbf{x}$ are vectors of global size N_g, local size N_l, and $a \in \mathbb{R}^N$. In order to avoid communication, the vectors $\mathbf{y}$ and $\mathbf{x}$ must have the same parallel decomposition where each parallel process owns the same pieces of each vector.

Parallel Vector Product

Vector product operations are used in several instances during a Krylov iteration including vector norm computations and the orthogonalization procedure. By definition, the vector product is a global operation that effectively collapses a set of vectors to a single value. Therefore, we cannot eliminate all global communications. Instead, vector product operations are formulated as *global reduction operations* that are efficiently supported by modern message passing libraries. For the dot product of two vectors $\mathbf{y}$ and $\mathbf{x}$, a single reduction is required such that:

$$d_l = \mathbf{y}_l \cdot \mathbf{x}_l,\ d_g = \sum_p d_l\ , \tag{2.32}$$

where the l subscript denotes a local quantity, d_l is the local vector dot product, and d_g is the global dot product generated by summing the local dot products over all p processes. Parallel norm operations can be conducted with the same single reduction. Consider the infinity norm operation:

$$||x||_{\infty,l} = \max_n \mathbf{y}[n],\ \forall n \in [1, N_l] \tag{2.33a}$$

$$||x||_{\infty,g} = \max_p ||x||_{\infty,l}\ . \tag{2.33b}$$

In this form, the local infinity norm is computed over the local piece of the vector. The reduction operation is then formed over all p processes such that the global max of the vector is computed and distributed to all processes.

Parallel Matrix-Vector Multiplications

We finally consider parallel matrix-vector multiplication operations using sparse matrices in a compressed storage format by considering Saad's outline as well as the more formal work of Tuminaro (Tuminaro et al., 1998). For these operations,

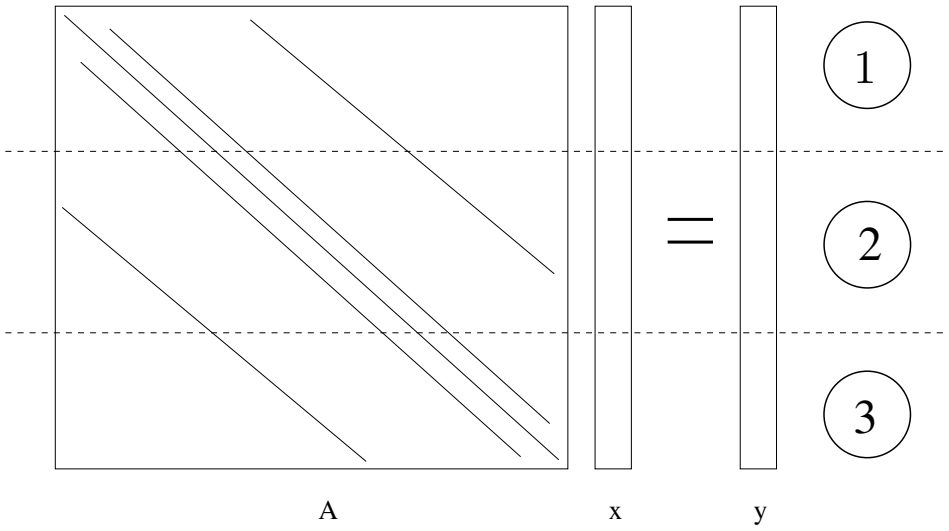

Figure 2.2: **Sparse matrix-vector multiply Ax = y operation partitioned on 3 processors.** *Each process owns a set of equations that correlates to its physical domain.*

more complex communication patterns will be required given that the entire global vector is required in order to compute a single element of the local product vector. Fortunately, the vast majority of the global vector components will be multiplied by zero due to the sparsity of the matrix and therefore much of the vector can be neglected. Instead we only require data from a handful of other processes that can be acquired through asynchronous/synchronous communications. Consider the sparse matrix-vector multiply in Figure 2.2 that is partitioned on 3 processors. Each process owns a set of equations that correlate to the physical domain of which it has ownership. We can break down the equations owned by each process in order to devise an efficient scheme for the multiplication. Consider the portion of the matrix-vector multiply problem owned by process 1 in Figure 2.2. As shown in Figure 2.3, the components of the matrix will be multiplied by pieces of the vector that are owned by all processors. For those pieces of the matrix that are owned by process 1 that act on the vector owned by process 1, we do these multiplications first as no communication is required. Next, process 1 gathers the components of the global vector owned by the other two processes that it requires to complete its part of the vector product. For this example, the components of matrix owned by process 1 that will operate on the global vector components owned by process 3 are zero, and therefore no vector elements are required to be scattered from process 3 to process 1. Those matrix elements owned by process 1 that will act on the piece of the vector owned by process 2 are not all non-zero,

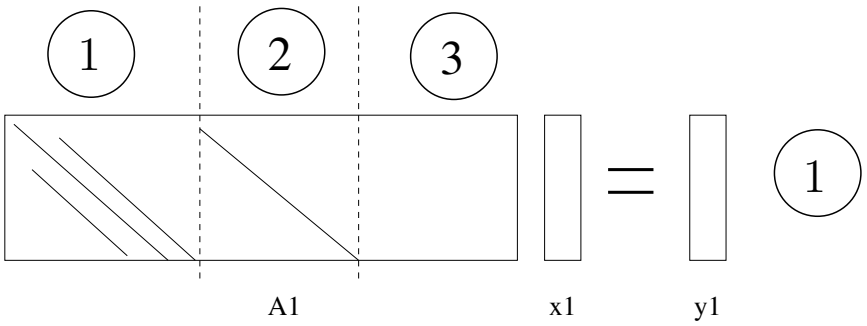

Figure 2.3: **Components of sparse matrix-vector multiply operation owned by process 1.** *The numbers above the matrix columns indicate the process that owns the piece of the global vector they are acting on. In order to compute its local components of the matrix-vector product, process 1 needs its matrix elements along with all elements of the global vector owned by processes 2 and 3. The piece of the matrix shown is* $\mathbf{A}_1$ *and it is acting locally on* $\mathbf{x}_1$ *to compute the local piece of the product,* $\mathbf{y}_1$.

and therefore we must gather the entire process 2 vector components onto process 1 to complete the multiplication. Conversely, processes 2 and 3 must scatter their vector components that are required by other processes (such as process 1) in order to complete their pieces of the product. This then implies that these domain connections for proper gather and scatter combinations must be constructed a priori. These data structures are typically generated by a data partitioning library. Mathematically, if we are performing a global matrix-vector multiply of the form $\mathbf{Ax} = \mathbf{y}$, then for this example on process 1 we have a sequence of local matrix-vector multiplications:n $\mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_1\mathbf{x}_2 = \mathbf{y}_1$, with the subscripts provided by notation in Figure 2.3. Here, some of the data is intrinsically local, and some must be gathered from other processes using the partitioning data structures.

Parallel Performance Implications for Krylov Methods

Knowing the parallel characteristics of the key operations we must perform in order to implement Krylov methods, we can make a few statements about parallel performance and implications for operation on machines of increasing size. Reconsider the matrix and vector operations required to implement Algorithm 2.1. For very large distributed machines, the global reduction operations required at several levels of Krylov algorithms stand to reduce scalability and performance. In the case of GMRES, these reductions include vector norm operations and the inner products required

for basis orthogonalization. Furthermore, communication between adjacent domains in matrix-vector multiply operations may also cause a bottleneck as the number of domains used in a simulation grows and the number of processors participating in the gather/scatter sequence requires a large communication bandwidth. The end result is that global data must be collected and communicated. For scaling improvement, we seek a reduction in these types of operations. In addition, these issues become more prominent as the Krylov iterations progress, causing the Krylov subspace to grow and the total number of operations needed to orthogonalize that subspace to increase.

As an example of these performance implications in practice, in a 2001 work, Gropp and colleagues presented results on fluid dynamic simulations that heavily leveraged Krylov methods in their solution schemes (Gropp et al., 2001). In this follow-on to their 1997 Bell Prize-winning work, part of their analysis included identifying parallel scalability bottlenecks generated by solver implementations in a strong scaling exercise where the number of processors was increased with respect to a fixed global problem size. Gropp's observations show that for their particular hardware, a distributed memory machine similar to modern architectures, that global reduction operations did not impede scalability, meaning that the global reduction operation occupied approximately the same percentage of compute time independent of the number of processors used. Rather, it was the gather/scatter operations required to communicate data to neighboring processors that reduced performance with an increasing percentage of compute time consumed by these operations as processor count was increased. Furthermore, it was noted that this reduction in scaling was a product of poor algorithmic strong scaling rather than hardware or implementation related issues as the algorithm requires more data to be scattered/gathered as the number of processors and therefore computational domains increased. In the case of a weak scaling exercise, we would instead expect this percentage to exhibit a more desirable behavior of remaining constant for gather/scatter operations as problem size would be scaled with the number of processors.

Chapter 3

Monte Carlo Solution Methods for Linear Systems

An alternative approach to approximate matrix inversion is to employ Monte Carlo methods that sample a distribution with an expectation value equivalent to that of the inverted operator. Such methods have been in existence for decades with the earliest reference noted here a manuscript published in 1950 by Forsythe and Leibler (Forsythe and Leibler, 1950). In their outline, Forsythe and Liebler in fact credit the creation of this technique to J. Von Neumann and S.M. Ulam some years earlier than its publication. In 1952 Wasow provided a more formal explanation of Von Neumann and Ulam's method (Wasow, 1952) and Hammersley and Handscomb's 1964 monograph (Hammersley and Handscomb, 1964) and Spanier and Gelbard's 1969 book (Spanier and Gelbard, 1969) present additional detail on this topic using a collection of references from the 1950's and early 1960's.

In this chapter, we will present the fundamentals of the Monte Carlo method for discrete linear systems. Both the forward and adjoint methods will be presented and analyzed including a brief variance analysis of several Monte Carlo estimators. The Sequential Monte Carlo and Monte Carlo Synthetic Acceleration methods will then be presented as a means of leveraging the basic Monte Carlo methods in an iterative refinement scheme that accelerates their convergence. Using these iterative schemes, a set of modest variance reduction techniques will be introduced and their effects on the Monte Carlo methods explored.

3.1 Preliminaries

We begin our discussion of Monte Carlo methods using these texts by seeking a solution to Eq (2.1). For a given linear operator $\mathbf{A}$, we can use diagonal splitting in a similar manner as the stationary method in Eq (2.15) to define the following operator[1]:

$$\mathbf{H} = \mathbf{I} - \mathbf{A}\,, \tag{3.1}$$

[1]It should be noted that non-diagonal splittings have been recently explored (Srinivasan, 2010) and have the potential to improve efficiency. However, it was observed in this work that this type of splitting did not improve performance in the asymptotic limit of $\rho(\mathbf{H}) \to 1$ for non-trivial problems.

such that we are solving the system:

$$\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{b} \,. \tag{3.2}$$

We can then form an alternative representation for $\mathbf{A}^{-1}$ by generating the *Neumann series*:

$$\mathbf{A}^{-1} = (\mathbf{I} - \mathbf{H})^{-1} = \sum_{k=0}^{\infty} \mathbf{H}^k \,, \tag{3.3}$$

which will converge if the spectral radius of $\mathbf{H}$ is less than 1. If we then apply this Neumann series to the right hand side of Eq (2.1) we acquire the solution to the linear problem:

$$\mathbf{A}^{-1}\mathbf{b} = \sum_{k=0}^{\infty} \mathbf{H}^k\mathbf{b} = \mathbf{x} \,. \tag{3.4}$$

An approximation of this summation by truncation will therefore lead to an approximation of the solution. If we expand the summation with a succession of matrix-vector multiply operations, we arrive at an alternative perspective of this summation by considering the i^{th} component of the solution vector:

$$x_i = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \dots \sum_{i_k}^{N} h_{i,i_1} h_{i_1,i_2} \dots h_{i_{k-1},i_k} b_{i_k} \,, \tag{3.5}$$

which can interpreted as a series of transitions between states,

$$\nu = i \to i_1 \to \dots \to i_{k-1} \to i_k \,, \tag{3.6}$$

in $\mathbf{H}$ where ν is interpreted as a particular random walk sequence permutation. We can generate these sequences of transitions through Monte Carlo random walks by assigning them both a probability and weight. As a reinterpretation of the iteration matrix, we then form the *Neumann-Ulam decomposition* of $\mathbf{H}$:

$$\mathbf{H} = \mathbf{P} \circ \mathbf{W} \,, \tag{3.7}$$

where $\circ$ denotes the Hadamard product operation[2], $\mathbf{P}$ denotes the transition probability matrix, and $\mathbf{W}$ denotes the transition weight matrix. This decomposition, a generalization of Dimov's work (Dimov et al., 1998), is an extension of the original Neumann-Ulam scheme in that now a weight cutoff can be used to terminate a random walk sequence and therefore truncate the Neumann series it is approximating. The

[2]The Hadamard product $\mathbf{A} = \mathbf{B} \circ \mathbf{C}$ is defined element-wise as $a_{ij} = b_{ij}c_{ij}$.

formulation of **P** and **W** will be dependent on whether we choose a direct or adjoint Monte Carlo sequence to estimate the state transitions in Eq (3.5). In the direct method, we will use the provided linear operator **A** to form the Neumann Ulam decomposition while the adjoint method will use the adjoint linear operator ($\mathbf{A}^T$ for real-valued systems) to form the decomposition.

3.2 Direct Monte Carlo Method

In the context of matrix inversion, a direct (forward) method resembles an adjoint Monte Carlo method in the reactor physics community where the solution state is sampled and the source terms that contribute to it are assembled. To achieve this, we build the direct method Neumann-Ulam decomposition per Dimov's approach by first choosing a probability matrix that is a row scaling of **H** such that its components are:

$$p_{ij} = \frac{|h_{ij}|}{\sum_j |h_{ij}|} \,. \tag{3.8}$$

From this, we then see that the probability of transitioning from a state i to a state j is implicitly linked to the original operator **A** in that those terms with large values, and therefore those that make the greatest contribution to the numerical solution, will be sampled with a higher probability than smaller terms. In addition, the row scaling provides a normalization over the state to which we are transitioning such that $\sum_j p_{ij} = 1$, meaning that we sample the probabilities over the rows of the matrix. The components of the weight matrix are then defined by Eq (3.7) as:

$$w_{ij} = \frac{h_{ij}}{p_{ij}} \,. \tag{3.9}$$

It should be noted here that if **A** is sparse, then **H**, **P**, and **W** must be sparse as well by definition. Additionally, we only compute **P** and **W** from the non-zero elements of **H** as those components that are zero will not participate in the random walk and thus produce identical sparsity patterns for all matrices.

Using these matrices, we can then form the expectation value of the direct solution. For a given random walk permutation ν, we define the weight of that permutation on the m^{th} step to be:

$$W_m = w_{i_0,i_1} w_{i_1,i_2} \cdots w_{i_{m-1},i_m} \,, \tag{3.10}$$

such that the weight of each transition event contributes to the total through multipli-

cation with $W_0 = 1$ as the starting weight. The contribution to the solution from a particular random walk permutation with k total events is then the *forward estimator*:

$$X_{i_0=i}(\nu) = \sum_{m=0}^{k} W_m b_{i_m} \,, \tag{3.11}$$

where $X_{i_0=i}(\nu)$ signifies that the solution state, i_0, in which the random walk ν started is also the state, i, in which we are tallying. We then define the probability that a particular random walk permutation of k events will occur:

$$P_\nu = p_{i,i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} \,. \tag{3.12}$$

Finally, we define the expectation value of X to be the collection of all random walk permutations and their probabilities:

$$E\{X_i\} = \sum_{\nu} P_\nu X_i(\nu) \,, \tag{3.13}$$

which, if expanded, directly recovers the exact solution by forming the Neumann series:

$$\begin{aligned} E\{X_i\} &= \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} p_{i,i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} w_{i,i_1} w_{i_1,i_2} \cdots w_{i_{k-1},i_k} b_{i_k} \\ &= x_i \,, \end{aligned} \tag{3.14}$$

therefore providing an unbiased Monte Carlo estimator.

In cases where we seek only approximate solutions, we need only to perform a predetermined number of random walks in order to generate an approximation for $\mathbf{x}$. If we are only to approximate the solution, we also need conditions by which we may terminate a random walk as the Neumann Ulam decomposition defined by Eqs (3.8) and (3.9) will create a random walk weight in Eq (3.10) that approaches, but never reaches zero. We do this by noticing that the factors added to Eq (3.10) will become diminishingly small due to their definition in Eq (3.9) and therefore their contributions to the solution estimate will become negligible. Using this, we choose terminate a random walk sequence with a *weight cutoff*, W_c, that is enforced when $W_m < W_c$ for a particular random walk permutation.

Forward Estimator Variance

We can compute the variance of the forward estimator through traditional methods by defining the variance, σ_i, for each component in the solution:

$$\sigma_i{}^2 = E\{X_i - (\mathbf{A}^{-1}\mathbf{b})_i\}^2 = E\{X_i^2\} - x_i^2 \,, \tag{3.15}$$

where the vector exponentials are computed element-wise. Inserting Eq (3.13) gives:

$$\sigma_i^2 = \sum_\nu P_\nu X_i(\nu)^2 - x_i^2 \,, \tag{3.16}$$

and applying Eq (3.11):

$$\sigma_i^2 = \sum_\nu P_\nu \Big(\sum_{m=0}^{k} W_m^2 b_{i_m}^2 + 2 \sum_{\substack{j=0 \\ m<j}}^{k} W_m W_j b_{i_m} b_{i_j} \Big) - x_i^2 \,. \tag{3.17}$$

We can distribute the random walk probability to yield an expanded form of the variance:

$$\sigma_i^2 = \sum_\nu P_\nu \sum_{m=0}^{k} W_m^2 b_{i_m}^2 + 2 \sum_\nu P_\nu \sum_{\substack{j=0 \\ m<j}}^{k} W_m W_j b_{i_m} b_{i_j} - x_i^2 \,. \tag{3.18}$$

In particular, we will isolate the first summation of the variance by expanding it:

$$\sum_\nu P_\nu \sum_{m=0}^{k} W_m^2 b_{i_m}^2 = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} p_{i,i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} w_{i,i_1}^2 w_{i_1,i_2}^2 \cdots w_{i_{k-1},i_k}^2 b_{i_k}^2 \,. \tag{3.19}$$

Using this expansion, we can arrive at a more natural reason for enforcing $\rho(\mathbf{H}) < 1$ for our Monte Carlo method to converge. Per the Hadamard product, we can concatenate the summation in Eq (3.19):

$$(\mathbf{P} \circ \mathbf{W} \circ \mathbf{W})^k = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} p_{i,i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} w_{i,i_1}^2 w_{i_1,i_2}^2 \cdots w_{i_{k-1},i_k}^2 \,. \tag{3.20}$$

If we assign $\mathbf{G} = \mathbf{P} \circ \mathbf{W} \circ \mathbf{W}$ as in Eq (3.7), we then have:

$$(\mathbf{P} \circ \mathbf{W} \circ \mathbf{W})^k = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} g_{i,i_1} g_{i_1,i_2} \cdots g_{i_{k-1},i_k} \,, \tag{3.21}$$

which is the general Neumann series for $\mathbf{G}$,

$$\mathbf{T} = \sum_{k=0}^{\infty} \mathbf{G}^k \,, \tag{3.22}$$

where $\mathbf{T} = (\mathbf{I} - \mathbf{G})^{-1}$. We can then insert T back into the variance formulation for a more concise definition:

$$\sigma_i^2 = (\mathbf{T}\mathbf{b}^2)_i + 2 \sum_{\nu} P_\nu \sum_{\substack{j=0 \\ m<j}}^{k} W_m W_j b_{i_m} b_{i_j} - x_i^2 \,. \tag{3.23}$$

We can relate $\mathbf{G}$ to $\mathbf{H}$ by noting that $\mathbf{G}$ simply contains an additional Hadamard product with the weight matrix. The Hadamard product has the property that:

$$|\mathbf{H} \circ \mathbf{W}| \geq |\mathbf{H}| \, |\mathbf{W}| \,. \tag{3.24}$$

Using the norm property of the Hadamard product and Eq (3.7), we can define the norm of $\mathbf{W}$ as:

$$\frac{|\mathbf{H}|}{|\mathbf{P}|} \geq |\mathbf{W}| \,. \tag{3.25}$$

Choosing the infinity norm of the operator as defined in Eq (2.10), the row normalized probability matrix will yield a norm of 1 giving the following inequality for relating $\mathbf{G}$ and $\mathbf{H}$:

$$|\mathbf{G}| \geq |\mathbf{H}|^2 \tag{3.26}$$

Using these relations to analyze Eq (3.23), we see that if $\rho(\mathbf{G}) > 1$, then the first sum in Eq (3.22) will not converge and an infinite variance will arise as the elements of $\mathbf{T}$ become infinite in Eq (3.23). We must restrict $\mathbf{G}$ to alleviate this and therefore restrict $\mathbf{H}$ due to Eq (3.26) with $\rho(\mathbf{H}) < 1$ so that our expectation values for the solution may have a finite variance.

Direct Method: Evolution of a Solution

As a means of visually demonstrating the direct Monte Carlo method, consider a 2-dimensional thermal diffusion problem with sources on the left and right hand sides of the domain and a uniform source throughout the domain of 1/5 the strength of the boundary sources as shown in Figure 3.1. For this problem, the number of histories used to compute the solution at each grid point (state) in the domain was increased from 1 to 1000 in order to demonstrate the effects on the solution and the statistical

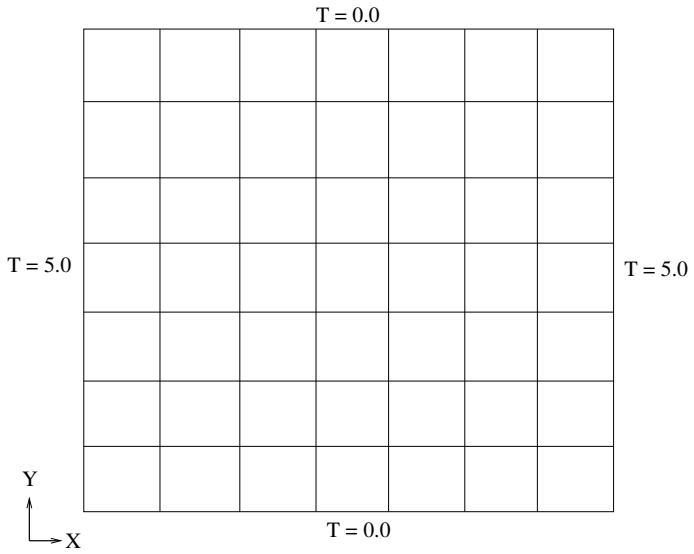

Figure 3.1: **Problem setup for 2D heat equation.** *Dirichlet conditions are set for the temperature on all 4 boundaries of the Cartesian grid. Background source of 1/5 the value of the boundary sources present.* 50×50 *grid.*

nature of the method. Figure 3.2 gives these results. As the number of histories used per state is increased, the statistical variance of the solutions is decreased as more tallies are made. Starting with single history at each state in the domain, the high variance prevents a precise solution from being obtained although we begin to see the solution take shape as expected. At 1000 histories per state, enough tallies have been made to generate a reasonable estimate for the structure of the solution. It is interesting to note here that as the statistical uncertainty is reduced at each grid point in the domain, the solution is resolved in a certain sense, analagous to the convergence of a traditional iterative method.

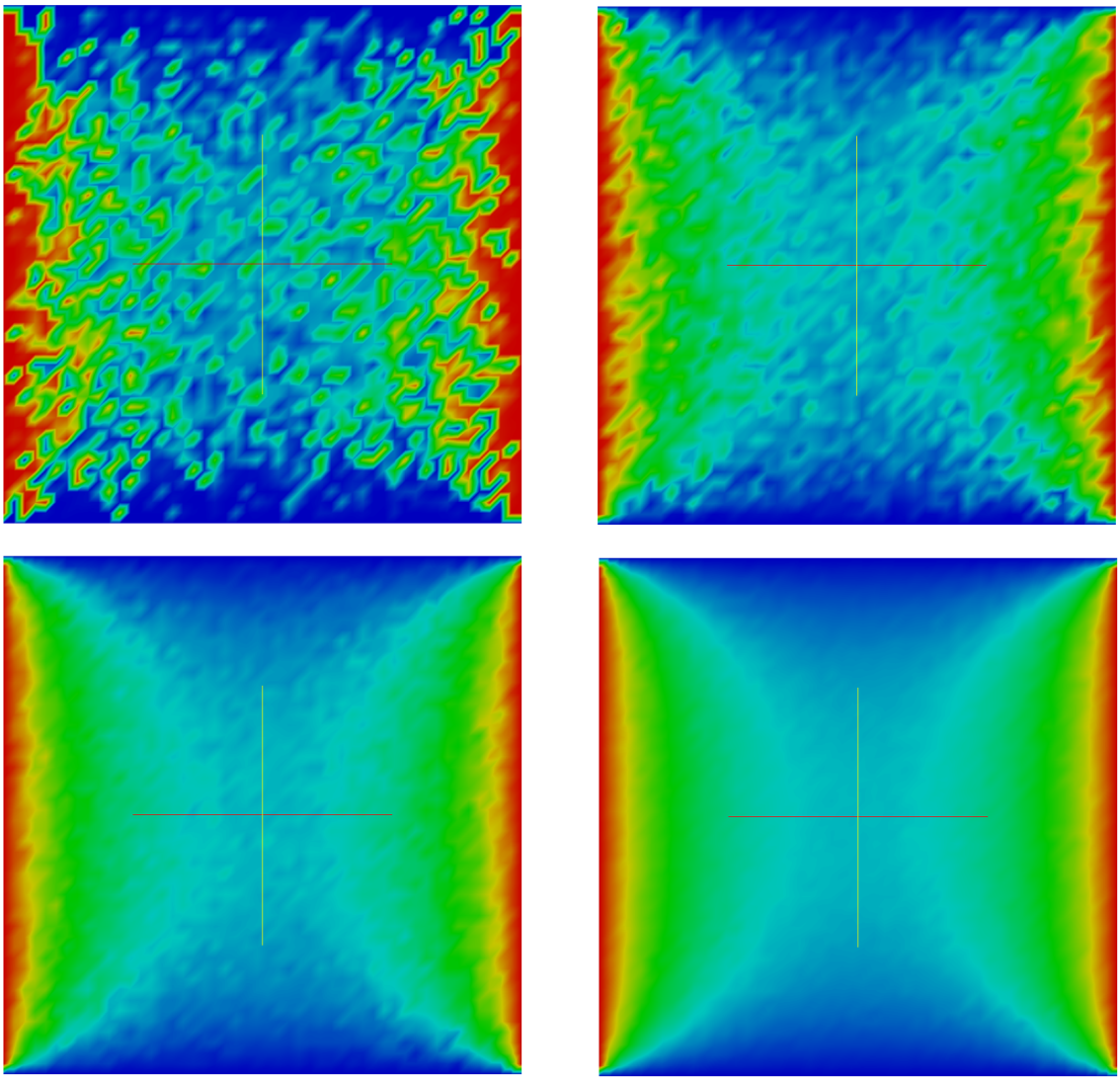

Figure 3.2: **Direct Monte Carlo solution to the heat equation with varying numbers of histories.** *Top left: 1 history per state. Top right: 10 histories per state. Bottom left: 100 histories per state. Bottom right: 1000 histories per state.*

3.3 Adjoint Monte Carlo Method

Often a more useful form, an alternative to forward Monte Carlo matrix inversion is the adjoint method. We begin by defining the linear system adjoint to Eq (2.1):

$$\mathbf{A}^T\mathbf{y} = \mathbf{d}\,, \tag{3.27}$$

where $\mathbf{y}$ and $\mathbf{d}$ are the adjoint solution and source vectors respectively and $\mathbf{A}^T$ is the adjoint operator for $\mathbf{A} \in \mathbb{R}^{N\times N}$. We can split this equation to mirror Eq (3.2):

$$\mathbf{y} = \mathbf{H}^T\mathbf{y} + \mathbf{d}\,. \tag{3.28}$$

As was required for convergence with the direct method using Eq (3.2), the spectral radius of $\mathbf{H}$ must remain less than 1 as $\mathbf{H}^T$ contains the same eigenvalues and therefore has the same spectral radius. By defining the following inner product equivalence (Spanier and Gelbard, 1969):

$$\langle \mathbf{A}^T\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, \mathbf{A}\mathbf{y}\rangle\,. \tag{3.29}$$

it follows that:

$$\langle \mathbf{x}, \mathbf{d}\rangle = \langle \mathbf{y}, \mathbf{b}\rangle\,. \tag{3.30}$$

Using these definitions, we can derive an estimator from the adjoint method that will also give the solution vector, $\mathbf{x}$. As with the direct method, we can acquire the adjoint solution by forming the Neumann series by writing Eq (3.28) as:

$$\mathbf{y} = (\mathbf{I} - \mathbf{H}^T)^{-1}\mathbf{d}\,, \tag{3.31}$$

which in turn yields the Neumann series using the adjoint operator:

$$\mathbf{y} = \sum_{k=0}^{\infty}(\mathbf{H}^T)^k\mathbf{d}\,. \tag{3.32}$$

We expand this summation to again yield a series of transitions that can be approximated by a Monte Carlo random walk sequence, this time forming the Neumann series in reverse order:

$$y_i = \sum_{k=0}^{\infty}\sum_{i_1}^{N}\sum_{i_2}^{N}\ldots\sum_{i_k}^{N} h_{i_k,i_{k-1}}\ldots h_{i_2,i_1}h_{i_1,i}d_{i_k}\,. \tag{3.33}$$

We can readily build an estimator for the adjoint solution from this series expansion, but we instead desire the solution to Eq (2.1). Here we have 2 unknowns, $\mathbf{y}$ and $\mathbf{d}$, and therefore we require two constraints to close the system. We use Eq (3.30) as the first constraint and as a second constraint we select:

$$\mathbf{d} = \boldsymbol{\delta}_j \,, \tag{3.34}$$

where $\boldsymbol{\delta}_j$ is one of a set of vectors in which the j^{th} component is the Kronecker delta function $\delta_{i,j}$. If we apply Eq (3.34) to our first constraint Eq (3.30), we get the following convenient outcome:

$$\langle \mathbf{y}, \mathbf{b} \rangle = \langle \mathbf{x}, \boldsymbol{\delta}_j \rangle = x_j \,, \tag{3.35}$$

meaning that if we compute the inner product of the original source and the adjoint solution using a delta function source, we recover one component of the original solution.

In terms of radiation transport, this adjoint method is equivalent to a traditional forward method where the initial state i_0 of the random walk is determined by sampling the source vector $\mathbf{b}$ with probabilities:

$$P_{i_0=i}(\nu) = \frac{|b_i|}{||\mathbf{b}||_1} \,. \tag{3.36}$$

The random walk starting weight will then be $W_0 = b_{i_0}$. As a result of using the adjoint system, we modify our probabilities and weights using the *adjoint Neumann-Ulam decomposition* of $\mathbf{H}$:

$$\mathbf{H}^T = \mathbf{P} \circ \mathbf{W} \,, \tag{3.37}$$

where now we are forming the decomposition with respect to the transpose of $\mathbf{H}$. We then follow the same procedure as the direct method for forming the probability and weight matrices in the decomposition. Using the adjoint form, probabilities should instead be column-scaled:

$$p_{ij} = \frac{|h_{ji}|}{\sum_j |h_{ji}|} \,, \tag{3.38}$$

such that we expect to select a new state, j, from the current state in the random walk, i, by sampling column-wise (or row-wise if an adjoint probability matrix is formed).

Per Eq (3.37), the transition weight is then defined as:

$$w_{ij} = \frac{h_{ji}}{p_{ij}} \, . \tag{3.39}$$

Using the decomposition we can then define an expectation value for the adjoint method. Given Eq (3.10) as the weight generated for a particular random walk permutation as in Eq (3.6) and our result from Eq (3.35) generated by applying the adjoint constraints, the contribution to the solution in state j from a particular random walk permutation of k events is then the *collision estimator*:

$$X_j(\nu) = \sum_{m=0}^{k} W_m \delta_{i_m,j} \, , \tag{3.40}$$

where the Kronecker delta indicates that the tally contributes only in the current state, i_m, of the random walk. Note here that the estimator in Eq (3.40) does not have a dependency on the source state as in Eq (3.40), providing a remedy for the situation in the direct method where we must start a random walk in each source state for every permutation if we want to compute a solution estimate for that state. In the adjoint method, we instead tally in all states and those of lesser importance will not be visited as frequently by the random walk. Finally, the expectation value using all permutations is:

$$E\{X_j\} = \sum_{\nu} P_\nu X_j(\nu) \tag{3.41}$$

which, if expanded in the same way as the direct method, directly recovers the exact solution:

$$\begin{aligned} E\{X_j\} &= \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \dots \sum_{i_k}^{N} b_{i_0} h_{i_0,i_1} h_{i_1,i_2} \dots h_{i_{k-1},i_k} \delta_{i_k,j} \\ &= x_j \, , \end{aligned} \tag{3.42}$$

therefore also providing an unbiased Monte Carlo estimate of the solution. Note that this expansion produces the effective sequence of matrix-vector multiplications with the b_{i_0} component of the source vector which will be selected at random for each walk permutation and the unbiasedness of the estimator relies on an unbiased sampling of the source. It should also be noted here that Eq (3.42) only computes a single component of our desired solution vector when really what we desire is the entire solution vector. In an adjoint Monte Carlo simulation using this estimator, the w_{ij} elements that are added into the tally for each state are only selected if the random

walk currently resides in that state. Much like a mesh tally in a particle transport simulation, we have N simultaneous tallies for $\mathbf{A} \in \mathbb{R}^{N\times N}$ that will yield the entire solution vector. Based on their current state in the system, the random walks will contribute tally j in this group.

Like the direct method, we also desire a criteria for random walk termination for problems where only an approximate solution is necessary. For the adjoint method, we utilize a *relative weight cutoff*:

$$W_f = W_c b_{i_0} \,, \tag{3.43}$$

where W_c is defined as in the direct method. The adjoint random walk will then be terminated after m steps if $W_m < W_f$ as tally contributions become increasingly small.

Collision Estimator Variance

We can compute the variance of the collision estimator in the same way as the direct estimator for each component in the solution where now:

$$\sigma_j^2 = \sum_\nu P_\nu \sum_{m=0}^{k} W_m^2 \delta_{i_m,j} + 2\sum_\nu P_\nu \sum_{\substack{l=0 \\ m<l}}^{k} W_m W_l \delta_{i_m,j}\delta_{i_l,j} - x_j^2 \,. \tag{3.44}$$

In this case, in the second summation $m \neq l$ for all states due to the form of the summation and therefore the delta functions, $\delta_{i_m,j}$ or $\delta_{i_l,j}$, will only be nonzero for random walks that are in the current solution state (when $i_m = i_l = j$) and other states visited do not contribute to the variance of the current state (when $i_m \neq i_l$). This is important as it shows a decoupling of the variance among the solution vector states, meaning that the variance in solution state i does not depend on the variance in solution state j.

Expanding the transition probabilities in the first sum again yields a Neumman series in the same form as that explored for the forward estimator, this time with the terms in reverse order and the introduction of the Kronecker delta:

$$\sigma_j^2 = \sum_{k=0}^{\infty}\sum_{i_1}^{N}\sum_{i_2}^{N}\cdots\sum_{i_k}^{N} p_{i_k,i_{k-1}}\cdots p_{i_2,i_1}p_{i_1,i_0} w_{i_k,i_{k-1}}^2 \cdots w_{i_2,i_1}^2 w_{i_1,i_0}^2 b_{i_0}^2 \delta_{i_k,j} + 2\sum_\nu P_\nu \sum_{\substack{l=0 \\ m<l}}^{k} W_m W_l \delta_{i_m,j}\delta_{i_l,j} - x_j^2 \,. \tag{3.45}$$

If we again define $\mathbf{G} = \mathbf{P} \circ \mathbf{W} \circ \mathbf{W}$, we then have:

$$\sigma_j^2 = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} g_{i_k,i_{k-1}} \cdots g_{i_2,i_1} g_{i_1,i_0} b_{i_0} \delta_{i_k,j} + 2 \sum_{\nu} P_\nu \sum_{\substack{l=0 \\ m<l}}^{k} W_m W_l \delta_{i_m,j} \delta_{i_l,j} - x_j^2 \,, \quad (3.46)$$

where now $\mathbf{G}$ in this case is the transpose of that used in Eq (3.21). The Kronecker delta again implies that the variance contribution from each random walk will only be in its current state and for many random walks, many starting weights of b_{i_0} will contribute to the variance in the same way as they contribute to the solution tally. As the transpose is formed and the Neumann series of $\mathbf{G}$ is in reverse order in Eq (3.46) relative to its formulation in the forward estimator, then Eq (3.46) and Eq (3.21) are equivalent and therefore the collision estimator is bound by the same restrictions on $\rho(\mathbf{G})$ and $\rho(\mathbf{H})$ to ensure that the variance is finite.

Expected Value Estimator

In addition to the collision estimator, an additional estimator is available due to the work of Okten (Ã–kten, 2005) that uses the method of expected values as a means to improve the Monte Carlo estimate. As outlined by Spanier and Gelbard (Spanier and Gelbard, 1969), the method of expected values is a deterministic averaging of events that may potentially occur in the Monte Carlo random walk sequence. Okten applied this principle directly to discrete Monte Carlo by forming the *expected value estimator* for a random walk of k events:

$$X_j(\nu) = b_j + \sum_{m=0}^{k} W_m h_{j,i_m} \quad (3.47)$$

where now the contribution of the iteration matrix is deterministically averaged at step m over all potential states j that may be reached from the current state i_m. Via Okten, the estimator can be shown to be unbiased through a comparison to the expected value estimator. We can first rewrite the summation in Eq (3.47):

$$X_j(\nu) = b_j + \sum_{m=0}^{k} \sum_{i=1}^{N} W_m \delta_{i_m,i} h_{ji} \quad (3.48)$$

where N is the number of states in the system. Immediately, we see the collision estimator as defined by Eq (3.40) and can therefore write the expectation value as:

$$E\{X_j\} = b_j + \sum_{i=1}^{N} E\{X_i\} h_{ji} \tag{3.49}$$

which is equivalently is the j^{th} component of Eq (3.2):

$$E\{X_j\} = b_j + \sum_{i=1}^{N} x_i h_{ji} \tag{3.50}$$

and is therefore and unbiased estimate. Compared to the collision estimator, the expected value estimator provides additional information at every step of the random walk, yielding potentially better statistics with the same amount of transport work. Conveniently, even if no Monte Carlo histories are computed, the expected value estimator still deterministically computes the first term of the Neumann Series, $\mathbf{H}^0\mathbf{b}$, whereas the collision estimator will provide no information.

Expected Value Estimator Variance

Following the collision estimator variance, the expected value estimator variance is given as:

$$\sigma_j^2 = \sum_{\nu} P_\nu b_j^2 + 2 \sum_{\nu} P_\nu b_j \sum_{m=0}^{k} W_m h_{j,i_m} + \sum_{\nu} P_\nu \sum_{m=0}^{k} W_m^2 h_{j,i_m}^2 + 2 \sum_{\nu} P_\nu \sum_{\substack{l=0 \\ m<l}}^{k} W_m W_l h_{j,i_m} h_{j,i_l} - x_j^2 \,. \tag{3.51}$$

As the delta functions are no longer present, the final summation in Eq (3.51) contains contributions for all valid state combinations in the same row of the iteration matrix and therefore the variance of each state in the solution tally is coupled to a group of other states as defined by the sparsity pattern of the iteration matrix. This coupling of variances is expected due to the deterministic averaging used to generate the expected value estimator.

Adjoint Method: Evolution of a Solution

As a means of visually demonstrating the adjoint Monte Carlo method, again consider the 2-dimensional thermal diffusion problem with sources on the left and right hand

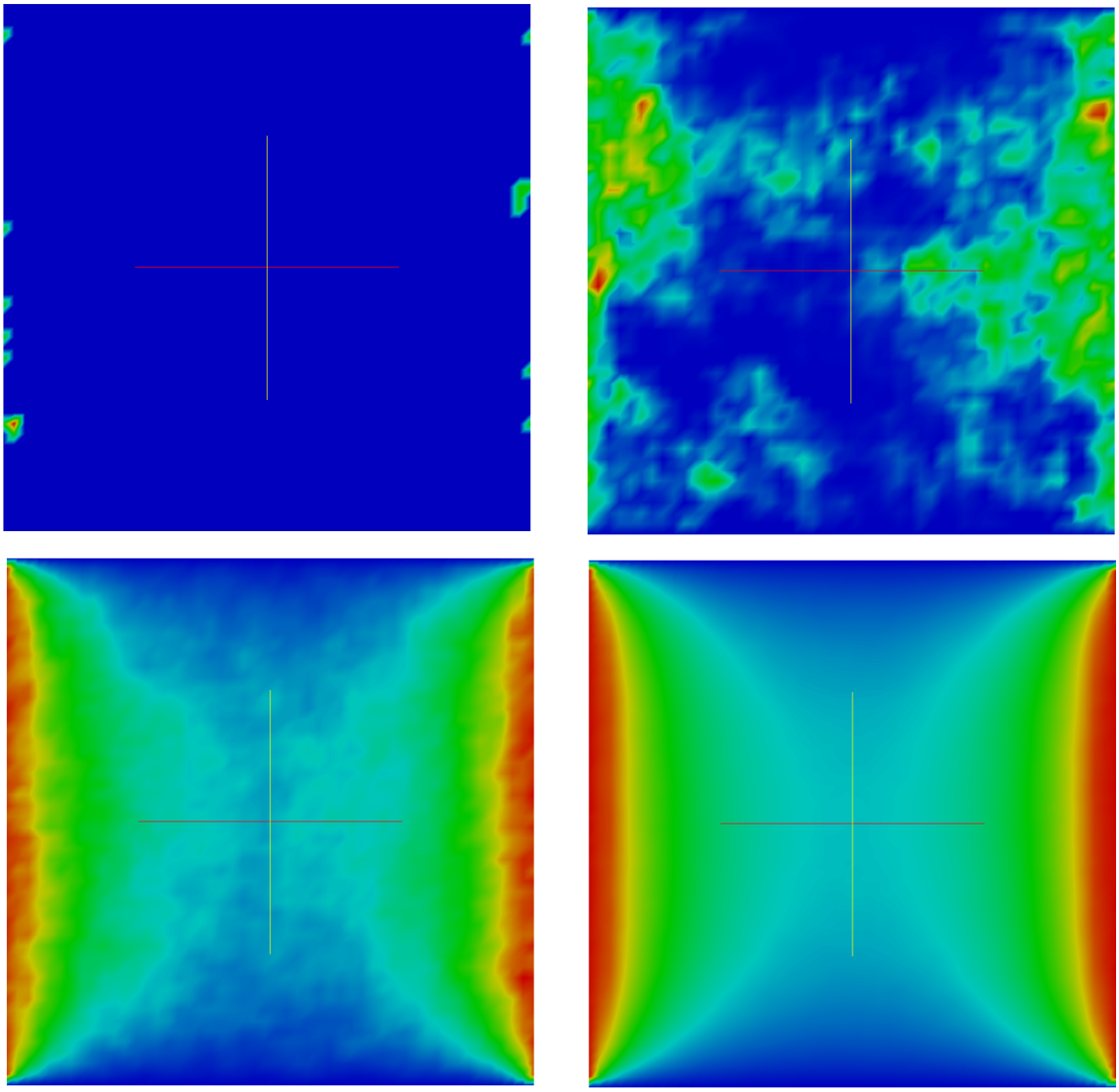

Figure 3.3: **Adjoint Monte Carlo solution to the heat equation with varying numbers of histories.** *Top left: 10 histories per state. Top right: 1,000 histories per state. Bottom left: 100,000 histories per state. Bottom right: 10,000,000 histories per state.*

sides of the domain and a smaller uniform source as shown in Figure 3.1. Using the adjoint method with the collision estimator, the number of histories sampled from the source was increased from 10 to 10,000,000 in order to show its effects on the solution and the statistical nature of the method. Figure 3.3 gives these results. As the number of histories used per state is increased, the statistical variance of the solutions is decreased as more tally contributions are made. At 10,000,000 histories per state, enough information has been tallied to generate a reasonable estimate for the structure

of the solution. The visual difference between Figures 3.2 and 3.3 is precisely that determined by their mathematics. As the adjoint solution evolves with the addition of histories, more histories emanate from the boundary and smaller uniform source with more penetrating from the boundary into the domain and making contributions to the tallies in those states.

3.4 Sequential Monte Carlo

The direct and adjoint Neumann-Ulam methods described are limited by a convergence rate of $1/\sqrt{N}$ by the Central Limit Theorem where N is the number of random walk permutations. In 1962, Halton presented a residual Monte Carlo method that moves towards exponential convergence rates (Halton, 1962) and further refined his work some years later (Halton, 1994). Applications of his work by the transport community have confirmed convergence rates on the order of of e^{-N} (Evans et al., 2003). In much the same way as projection methods, Halton's method, sequential Monte Carlo, utilizes the adjoint Monte Carlo solver as a means of directly reducing the elements residual vector. He proposed the following iterative scheme as a solution to Eq (2.1)

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k \,, \tag{3.52a}$$

$$\mathbf{A}\boldsymbol{\delta}^k = \mathbf{r}^k \,, \tag{3.52b}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \boldsymbol{\delta}^k \,, \tag{3.52c}$$

where the correction $\boldsymbol{\delta}^k$ is computed by the adjoint Monte Carlo method at each iteration. The merits of Halton's approach are immediately visible in that we have now broken the binding of the convergence rate to the Central Limit Theorem. Here, the Monte Carlo solver is used to produce a correction from the residual, analogous to using the residual to extract a correction from the search subspace in a projection method. By doing this, the Monte Carlo error is bound in the correction used to update the solution and therefore does not explicitly manifest itself in the overall convergence of the solution. The downside of such a method is that if the solution guess is poor, then many iterations are required in order to reach exponential converge as the Monte Carlo error (and therefore the Central Limit Theorem) does dominate in this situation.

3.5 Monte Carlo Synthetic Acceleration

Using the ideas of Halton, Evans and Mosher recently developed a Monte Carlo solution method that was not prohibited severely by the quality of the initial guess for the system (Evans and Mosher, 2009) and later applied it more rigorously as a solution mechanism for the radiation diffusion equation (Evans et al., 2012). With their new methods, they achieved identical numerical results as conventional Krylov solvers well as comparable performance in both number of iterations and CPU time.

Their approach was instead to use residual Monte Carlo as a synthetic acceleration for a stationary method. To derive this method, we begin by splitting the operator in Eq (2.1)

$$\mathbf{x} = (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b} \,. \tag{3.53}$$

With this we can then define the stationary method *Richardson's iteration* as:

$$\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}^{k} + \mathbf{b} \,, \tag{3.54}$$

which will converge if $\rho(\mathbf{I} - \mathbf{A}) < 1$. We then define the solution error at the k^{th} iterate relative to the true solution:

$$\delta\mathbf{x}^{k} = \mathbf{x} - \mathbf{x}^{k} \,. \tag{3.55}$$

Subtracting Eq (3.54) from Eq (3.53) we get:

$$\delta\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\delta\mathbf{x}^{k} \,. \tag{3.56}$$

Subtracting from this $(\mathbf{I} - \mathbf{A})\delta\mathbf{x}^{k+1}$ yields:

$$\begin{aligned} \mathbf{A}\delta\mathbf{x}^{k+1} &= (\mathbf{I} - \mathbf{A})(\mathbf{x}^{k+1} - \mathbf{x}^{k}) \\ &= \mathbf{r}^{k+1} \,. \end{aligned} \tag{3.57}$$

Using this, we define the following scheme that will converge in one iteration if $\mathbf{A}$ is inverted exactly:

$$\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}^{k} + \mathbf{b} \,, \tag{3.58a}$$

$$\mathbf{A}\delta\mathbf{x}^{k+1} = \mathbf{r}^{k+1} \,, \tag{3.58b}$$

$$\mathbf{x} = \mathbf{x}^{k+1} + \delta\mathbf{x}^{k+1} \,. \tag{3.58c}$$

However, $\mathbf{A}$ is only approximately inverted by our numerical methods and therefore we instead pose an iterative scheme in which the Monte Carlo solvers are used to invert the operator. The *Fixed-Point Monte Carlo Synthetic-Acceleration* (MCSA)

method is defined as:

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + \mathbf{r}^{k} \,, \tag{3.59a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \,, \tag{3.59b}$$

$$\mathbf{A}\delta\mathbf{x}^{k+1/2} = \mathbf{r}^{k+1/2} \,, \tag{3.59c}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2} \,, \tag{3.59d}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \,, \tag{3.59e}$$

where the adjoint Monte Carlo method is used to generate the solution correction from the residual and Richardson's iteration in the first step has been rewritten as a residual correction. Using Monte Carlo in this way achieves the same effect as Halton's method, decoupling its convergence rate from the overall convergence rate of the method. Here, the approximate Monte Carlo solution is not driven to a particular convergence as it merely supplies a correction for the initial guess generated by Richardson's iteration. Rather, only a set number of histories are required using the adjoint method to generate the correction. In addition, the fact that the scheme in Eq (3.58) will converge in one iteration if $\mathbf{A}$ is inverted exactly means that as more and more stochastic histories are used to compute the correction and the error is reduced towards zero, the number of iterations required for MCSA to converge should decrease accordingly, thus accelerating the solution.

In addition to the Monte Carlo solver parameters dictating the number of histories and weight cutoff, the outer MCSA iterations also have the following stopping criteria:

$$||\mathbf{r}||_{\infty} < \epsilon \, ||\mathbf{b}||_{\infty} \,, \tag{3.60}$$

where ϵ is a user-defined parameter. As with any iterative method, other stopping criteria using other vector norms could be computed, however, for this work we will only use Eq (3.60). We therefore have 3 parameters to tune in an MCSA implementation: the number of Monte Carlo histories computed in the adjoint solve during each MCSA iteration, the weight cutoff for those histories, and the total MCSA convergence tolerance as specified by ϵ.

Preconditioning MCSA

In most cases, at least a minimal amount of *preconditioning* of the linear system will be required in order to use the class of stochastic methods described. Although these

methods have no symmetry requirements for convergence, they do require that the spectral radius of the iteration matrix be less than one. Preconditioning serves as a means of achieving this by altering the eigenvalue spectrum of the iteration matrix.

Basic Preconditioning

As an example, to achieve a spectral radius of less than one for diagonally dominant matrices point Jacobi preconditioning can be used such that the preconditioning matrix $\mathbf{M}$ is:

$$\mathbf{M} = diag(\mathbf{A}) \,, \tag{3.61}$$

which may be trivially inverted. With the application of this preconditioner we are instead solving the following scaled linear system:

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \,. \tag{3.62}$$

Next, we can apply MCSA to solve Eq (3.62):

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + \mathbf{M}^{-1}\mathbf{r}^{k} \,, \tag{3.63a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \,, \tag{3.63b}$$

$$\mathbf{M}^{-1}\mathbf{A}\delta\mathbf{x}^{k+1/2} = \mathbf{M}^{-1}\mathbf{r}^{k+1/2} \,, \tag{3.63c}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2} \,, \tag{3.63d}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \,, \tag{3.63e}$$

where the adjoint Monte Carlo solve now has a preconditioned operator from which to build weights and probabilities for transport and a preconditioned source vector to sample.

Choosing point Jacobi preconditioning with MCSA is advantageous for several reasons. First, $\rho(\mathbf{I} - \mathbf{M}^{-1}\mathbf{A}) < 1$ is true for all $\mathbf{A}$ that is diagonally dominant and is easy to formulate because the inversion of $\mathbf{M}$ is trivial. Second, because the adjoint Monte Carlo method used within MCSA to compute the correction operates on a linear problem with the preconditioned operator, then $\mathbf{H}$ in the adjoint solver will have a zero term in each of its diagonal elements, thereby eliminating all in-state transitions during the random walk sequence. Because of this, point Jacobi preconditioning should be considered for many classes of problems, regardless of any other preconditioning that is applied to the system.

General Preconditioning Strategies

It is possible to use general left, right, and left/right preconditioning with MCSA by carefully considering the underlying Monte Carlo problem that will be solved with the Neumann-Ulam method. We consider here the general left/right preconditioned method as the left or right preconditioned methods can be inferred from its formulation. We consider a left preconditioner $\mathbf{M_L}$ and a right preconditioner $\mathbf{M_R}$. The left/right preconditioned linear problem is then:

$$\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\mathbf{M}_R\mathbf{x} = \mathbf{M}_L^{-1}\mathbf{b}\,. \tag{3.64}$$

To handle the right preconditioning, the system is written with a substitution of variables:

$$\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\mathbf{u} = \mathbf{M}_L^{-1}\mathbf{b}\,, \tag{3.65}$$

with

$$\mathbf{x} = \mathbf{M}_R^{-1}\mathbf{u}\,. \tag{3.66}$$

To apply such a method to MCSA, we solve for the substituted variable $\mathbf{u}$ during the iteration sequence:

$$\mathbf{u}^{k+1/2} = \mathbf{u}^k + \mathbf{r}^k\,, \tag{3.67a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{M}_L^{-1}(\mathbf{b} - \mathbf{A}\mathbf{M}_R^{-1}\mathbf{u}^{k+1/2})\,, \tag{3.67b}$$

$$\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\delta\mathbf{u}^{k+1/2} = \mathbf{r}^{k+1/2}\,, \tag{3.67c}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^{k+1/2} + \delta\mathbf{u}^{k+1/2}\,, \tag{3.67d}$$

$$\mathbf{r}^{k+1} = \mathbf{M}_L^{-1}(\mathbf{b} - \mathbf{A}\mathbf{M}_R^{-1}\mathbf{u}^{k+1})\,, \tag{3.67e}$$

and then recover the original solution vector with Eq (3.66). For the Monte Carlo problem, we isolate the generation of the correction:

$$\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\delta\mathbf{u}^{k+1/2} = \mathbf{r}^{k+1/2}\,, \tag{3.68}$$

and note that the precondtioned residual of the substituted variable is now serving as the source and the new iteration matrix is:

$$\mathbf{H} = \mathbf{I} - \mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\,. \tag{3.69}$$

As we require (i, j) element-wise access to the iteration matrix in order to construct probabilities and weights for the Monte Carlo procedure from the Neumann-Ulam decomposition, the *composite operator*, $\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}$, must be formed via matrix-matrix multiplication.

Several possible shortcomings of this preconditioning approach are readily observed. First, the matrix-matrix multiplication operation for sparse, parallel distributed matrices is significantly more expensive than a matrix-vector multiplication operation. Second, each preconditioner must be explicitly inverted, an operation in itself that may be expensive and which prohibits the use of any preconditioners which provide no mechanism to extract their inverse. Third, for many modern preconditioning methods, this inversion may yield dense matrices, destroying sparsity and further impeding the performance of a matrix-matrix multiplication operation. It is also interesting to note that the Monte Carlo problem in the general left/right preconditioned scheme given by Eq (3.68) is not fully left/right preconditioned (meaning that we do not recover $\mathbf{x}$), but instead part of a sequence for finding the substituted variable $\mathbf{u}$. We do, however, gain the benefits of this general preconditioning by building the iteration matrix in Eq (3.69) from the fully preconditioned linear operator. In addition, for MCSA to apply to increasingly difficult problems, more advanced preconditioning techniques that require the generation of the composite operator may be necessary for convergence.

Alternative Fixed Point Iterations

3.6 Monte Carlo Method Selection

p The MCSA method defined in Eq. (3.59) uses the adjoint method to estimate the error in a residual Monte Carlo solve instead of the direct method outlined in §3.2. To demonstrate the effectiveness of the adjoint method over the direct method within the context of MCSA, we choose the two-dimensional time-dependent Poisson equation as a simple model problem:

$$\frac{\partial \mathbf{u}}{\partial t} = \nabla^2 \mathbf{u}\,. \tag{3.70}$$

For all comparisons, a single time step is computed with backwards Euler time integration. The Laplacian is differenced on a square Cartesian grid with a second-order five-point stencil,

$$\nabla_5^2 = \frac{1}{\Delta^2}[u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}]\,, \tag{3.71}$$

and a fourth-order nine-point stencil,

$$\nabla_9^2 = \frac{1}{6\Delta^2}[4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1} + u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{i,j}] , \quad (3.72)$$

both assuming a grid size of Δ in both the i and j directions. For a single time step solution, we then have the following sparse linear system to be solved with the MCSA method:

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{u}^n . \quad (3.73)$$

Both the stencils will be used to vary the size and density of the sparse linear system in Eq. (3.73).

A timing and convergence study is used to demonstrate the effectiveness of the adjoint method with the collision estimator as compared to the direct method. To assess both the CPU time and number of iterations required to converge to a solution, a problem of constant Δ was used with varying values of the number of mesh elements, fixing the spectral radius of the system at a constant value for each variation. Both the five-point and nine-point stencils were used with both the direct and adjoint solvers. For each case, $N \times N$ total random walk permutations were computed per MCSA iteration where $N \times N$ is the number of discrete grid points in the system. Solver parameters were set to a weight cutoff of 1×10^{-4} for the stochastic linear solver and a convergence tolerance of 1×10^{-8} for the MCSA iterative solver. Figure 3.4 gives the CPU time needed for each case to converge in seconds and Figure 3.5 gives the number of iterations needed for each case to converge to the specified tolerance as a function of the problem size. All computations presented in this section and §3.7 were completed on a 3.0 GHz Intel Core 2 Quad Q9650 CPU machine with 16 GB 1067 MHz DDR3 memory.

We see clearly in Figure 3.4 that the using the adjoint solver with MCSA results in a speedup over the direct solver while the number of iterations required to converge is also reduced as shown in Figure 3.5. We expect this for several reasons. First, with an equivalent number of histories specified for both solvers per MCSA iteration and a system of size $N \times N$, the direct solver will compute a single random walk for each state in the system per iteration to acquire a solution in that state, regardless of the size of the residual in that state. This is necessary in the direct method to ensure a contribution from each state as the random walk sequence will only contribute to the starting state. For the adjoint method, a total of $N \times N$ random walk events will

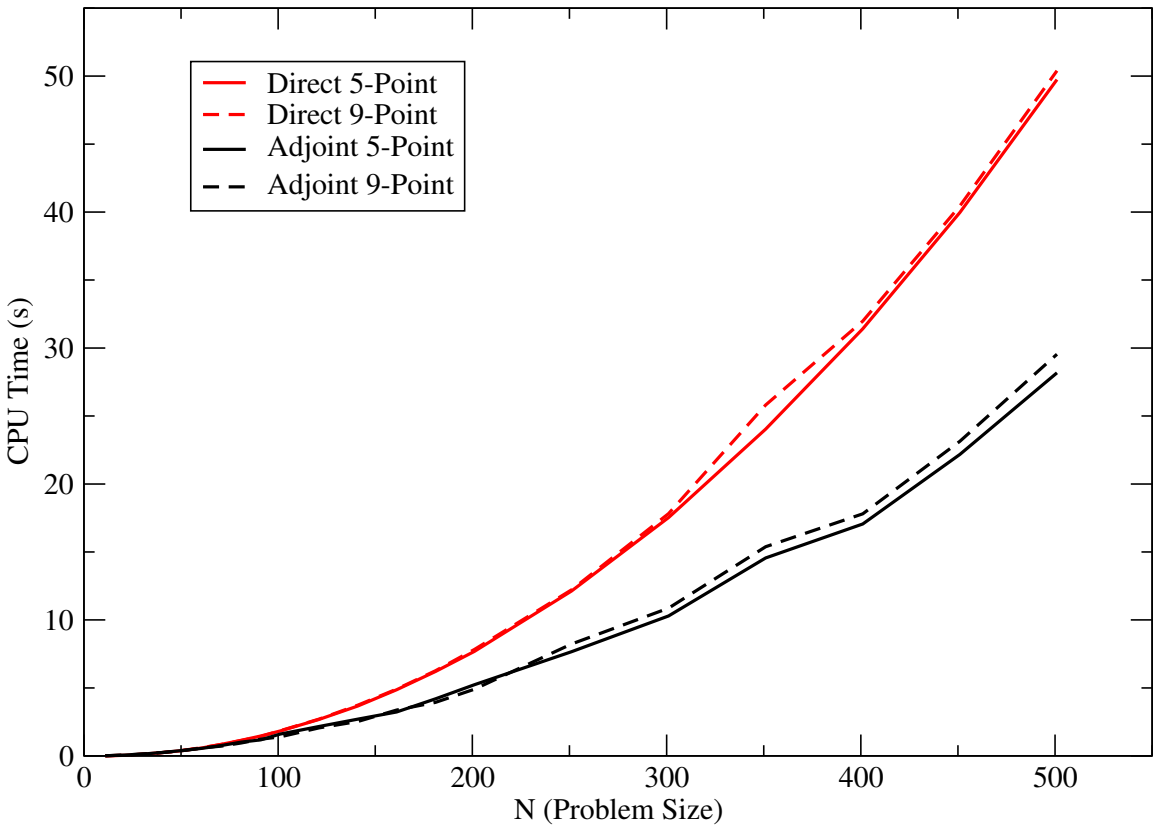

Figure 3.4: **CPU Time (s) to converge vs. Problem Size (N for an $N \times N$ square mesh).** *Both the adjoint and direct solvers are used with the five point and nine point stencils. A CPU time speedup is noted with the adjoint method due to the higher density of random walk events in regions with a large residual.*

have their starting state determined by sampling the residual vector. Because the random walk sequence contributes to the state in which it currently resides, sampling the residual vector as the Monte Carlo source gives a higher density of random walk events in regions with a high residual, thus giving a more accurate correction in that region due to reduced statistical error. From an iteration perspective, Figure 3.5 shows that using the direct method yields a roughly unchanging number of iterations required to converge as the problem size increases. Again, if we desire a correction value for all states in the problem, then we must start a random walk in each state in the system which does not reduce the number of iterations need as the problem size grows. Conversely, as the problem size grows in the adjoint method, the additional stochastic histories that will be computed are concentrated in regions with a large residual, further reducing the stochastic error in the correction in those regions and subsequently reducing the required number of iterations to converge.

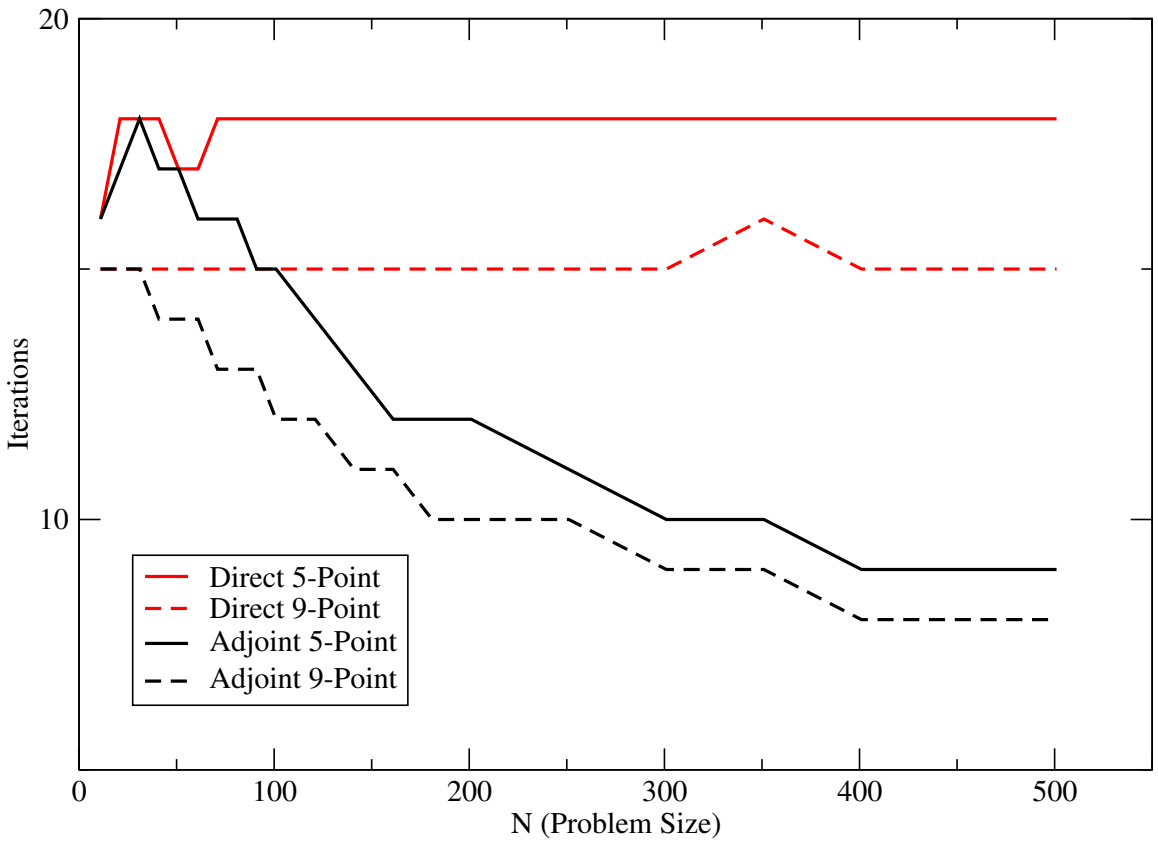

Figure 3.5: **Iterations to converge vs. Problem Size (N for an $N \times N$ square mesh).** *Both the adjoint and direct solvers are used with the five-point and nine-point stencils.*

As an additional comparison, the convergence behavior of MCSA can be analyzed using both the adjoint and direct solvers to detect any performance benefits. To assess the convergence properties of MCSA using each solver and stencil, the infinity norm of the residual computed in Eq. (3.59) was collected at each iteration for a fixed problem size of $N = 500$. Figure 3.6 gives the results of these computations. First, it is worthy to note on the semilog plot that we are indeed achieving the expected exponential convergence from MCSA with both Monte Carlo solvers. Second, we note that using the adjoint method with the same number of stochastic histores per MCSA iteration gives a faster rate of converge for the same reasons as above. We also note here that fewer iterations are required for convergence when the 9-point stencil is used to discretize the Laplacian operator (although at no gain in speed as given by the results in Figure 3.4). This is due to the fact that the smaller discretization error directly corresponds to a more well defined residual source generated by the

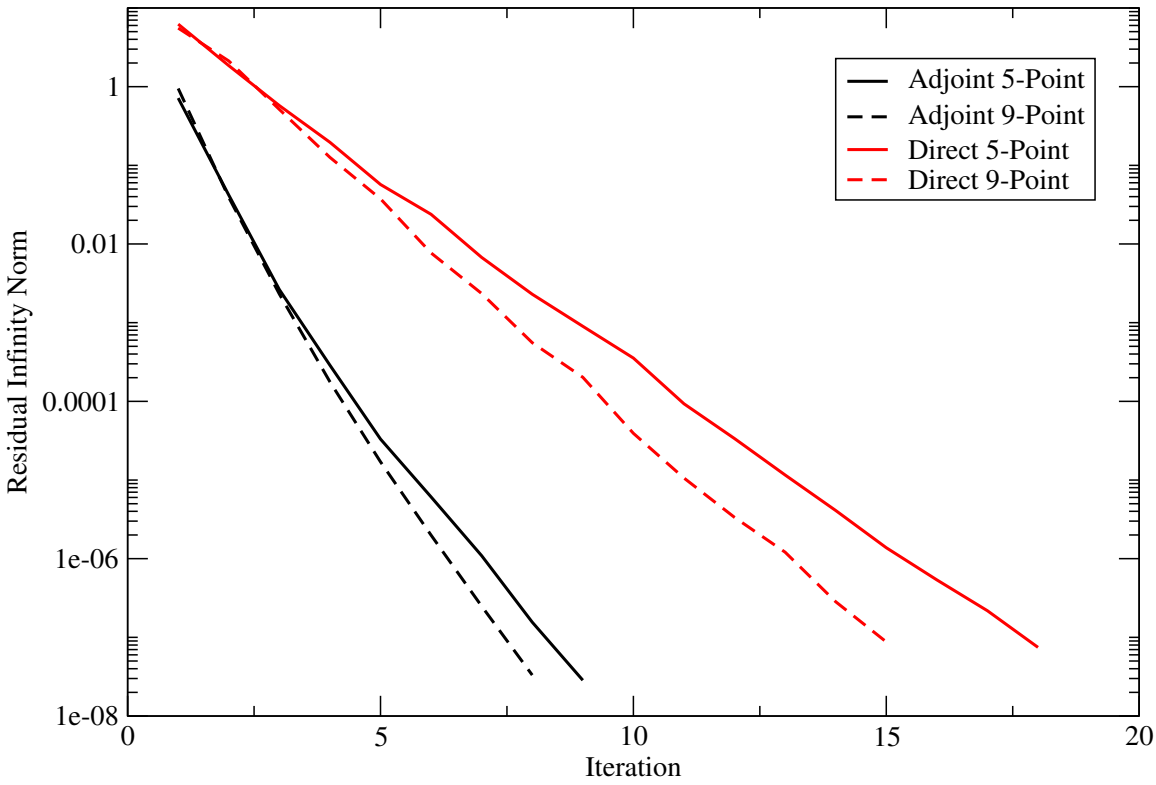

Figure 3.6: **Infinity norm of the solution residual vs. iteration number for a problem of size** $N = 500$**.** *Both the adjoint and direct solvers are used with the five point and nine point stencils. A higher rate of convergence is observed for MCSA using the adjoint Monte Carlo solver as compared to the direct method when both solvers compute the same number of random walks per iteration.*

Richardson extrapolation for the Monte Carlo calculation. In addition, the better defined source is transported through a domain described more accurately by the 9-point stencil, thus yielding a more accuracte correction vector from the Monte Carlo calculation.

3.7 MCSA Comparison to Sequential Monte Carlo

To further motivate using Monte Carlo Synthetic Acceleration, we compare its performance to Halton's Sequential Monte Carlo method on which previous work in this area was based. For this comparison, we use the same transient Poisson problem as described in the previous section and choose only the 5-point stencil to discretize the Laplacian operator as the previous results yielded little qualitative difference between the discretizations. Both MCSA and Halton's method are used with the adjoint Monte Carlo solver and the collision estimator. In order to complete the same study as in the previous section, the number of histories computed by the Monte Carlo solver at each iteration had to be doubled to $2 \times N \times N$ in order to ensure convergence in Sequential Monte Carlo Method. For the majority of the problems in the previous section, the Sequential method used with $N \times N$ histories would not converge. Figure 3.7 gives the CPU time results for this comparison as a function of problem size while Figure 3.8 gives the number of iterations to converge as a function of problem size with a convergence tolerance of 1×10^{-8}. In both cases, using the Monte Carlo solver as a synthetic acceleration rather than in a pure residual Monte Carlo scheme resulted in a reduction in both CPU time and iterations required to converge. The additional Richardson extrapolation between each Monte Carlo solve in the MCSA method gives a better converged residual source to use with the Monte Carlo calculation while the Sequential method requires more iterations to achieve the same level of convergence in the residual.

The benefits of using a synthetic acceleration scheme are also noted when the infinity norm of the residual computed at each iteration for both methods was collected at each iteration for a fixed problem sizes of $N = 100$ and $N = 500$ as shown in figures Figure 3.9 and 3.10 respectively. In both cases, the Sequential method is subject to two regimes of exponential convergence with high frequency error modes removed in the first regime leaving lower frequency and slower converging error modes in the second. Using MCSA we a see a single rate of exponential convergence observed to be much higher than that computed by Halton's method due to the fact that the extra Richardson iteration is providing a smoothing effect to alleviate the error mode variations. Even with the doubling of the number of stochastic histories computed per time step in order to ensure convergence for the Sequential method, we still see robustness issues with a non-monotonically decreasing residual observed for the

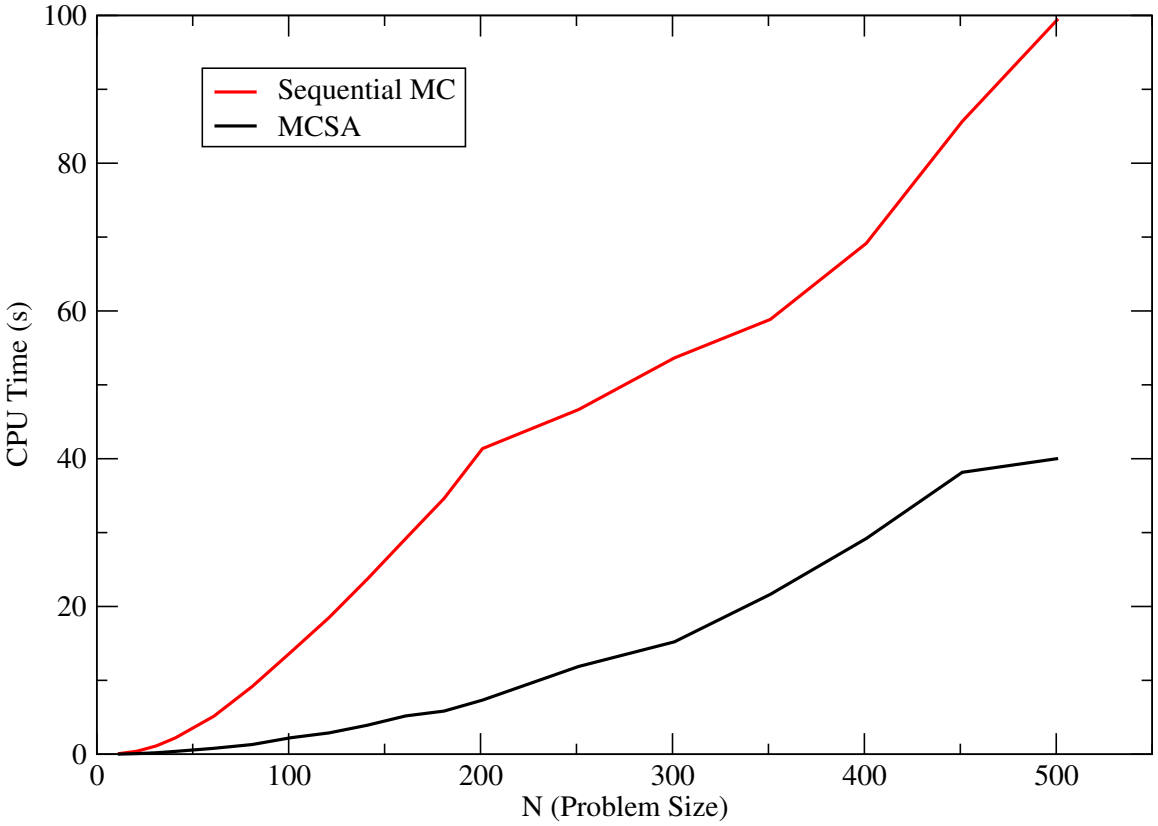

Figure 3.7: **CPU Time (s) to converge vs. Problem Size (N for an $N \times N$ square mesh).** *Both the Sequential Monte Carlo and MCSA solvers are used with the five point stencils and the adjoint Monte Carlo solver. The number of random walks was twice the number of discrete states in the system in order to ensure convergence in the Sequential Monte Carlo method.*

$N = 100$ case. In both cases the MCSA solver is observed to be robust with a monotonically decreasing residual.

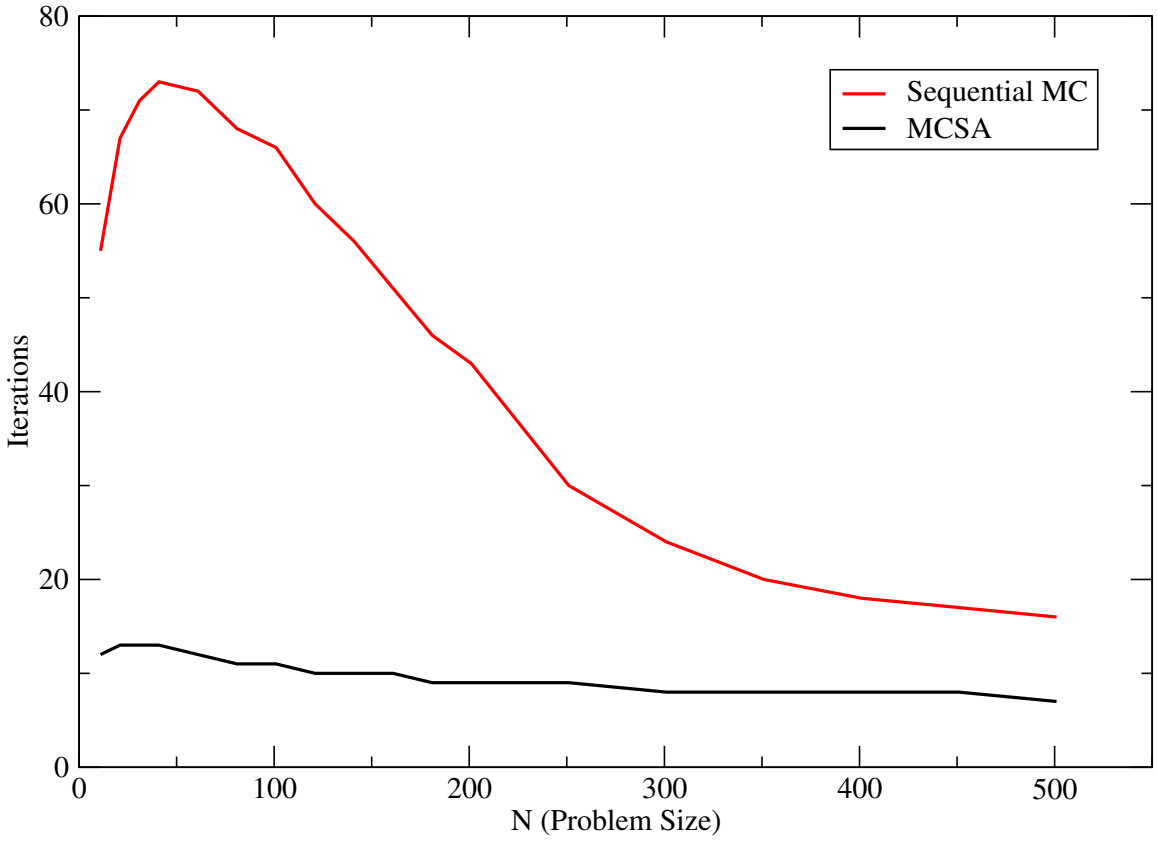

Figure 3.8: **Iterations to converge vs. Problem Size (N for an $N \times N$ square mesh).** *Both the Sequential Monte Carlo and MCSA solvers are used with the five point stencils and the adjoint Monte Carlo solver.*

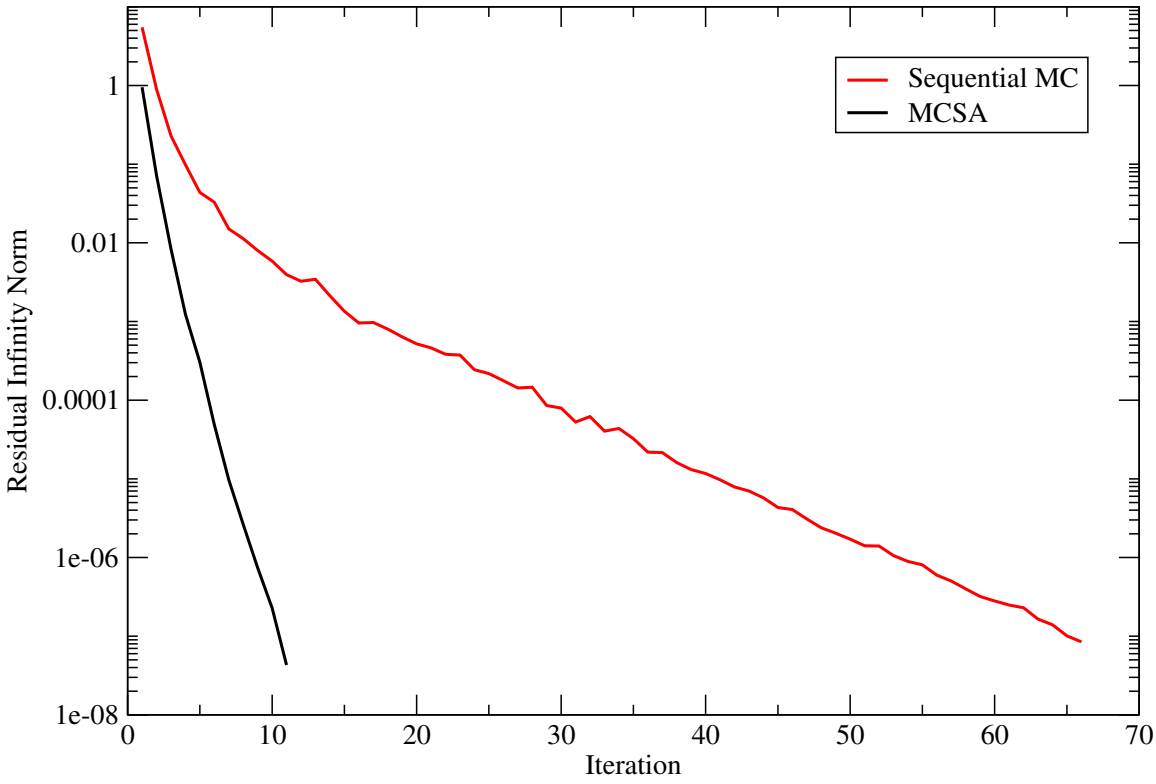

Figure 3.9: **Infinity norm of the solution residual vs. iteration number for a problem of size** $N = 100$**.** *Both the Sequential Monte Carlo and MCSA solvers are used with the five point stencils and the adjoint Monte Carlo solver.*

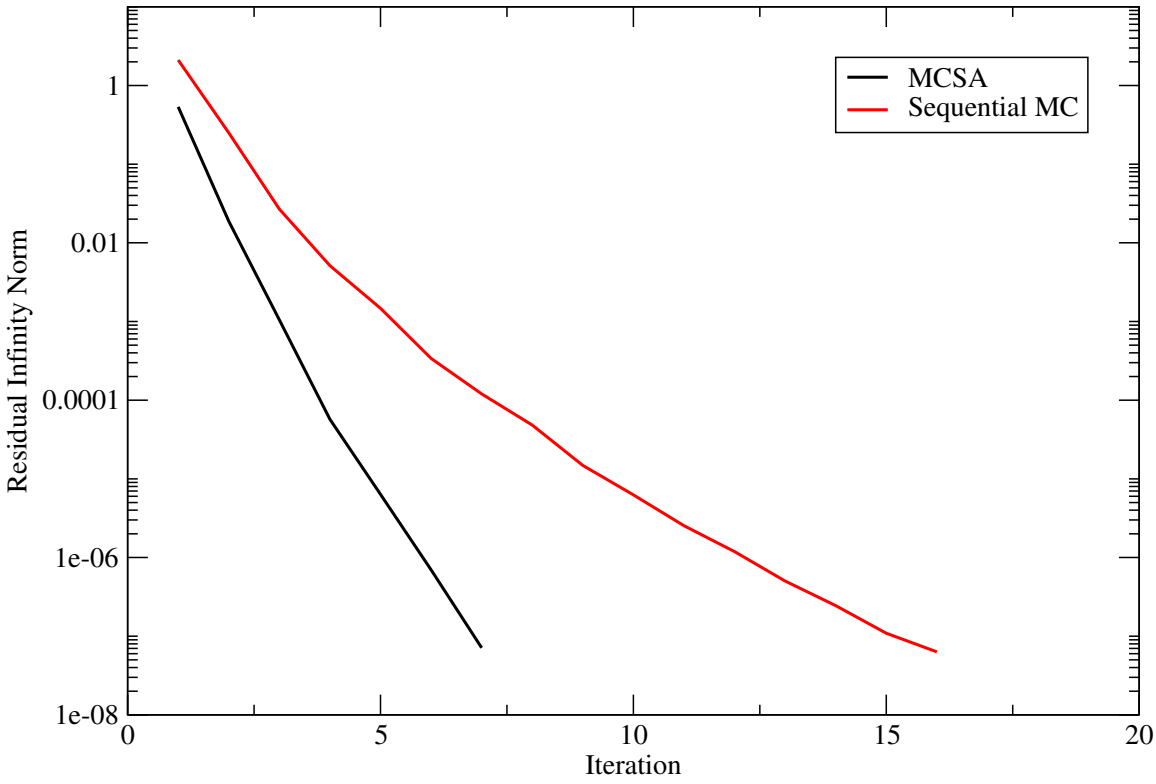

Figure 3.10: **Infinity norm of the solution residual vs. iteration number for a problem of size** $N = 500$**.** *Both the Sequential Monte Carlo and MCSA solvers are used with the five point stencils and the adjoint Monte Carlo solver.*

3.8 Monte Carlo Parameter and Estimator Analysis

With the adjoint method shown to be more effective than the direct method and MCSA to have better iterative and timing performance than sequential Monte Carlo, we now aim to study the effects of the adjoint Monte Carlo parameters, weight cutoff and number of histories, on MCSA performance with both the collision and expected value estimators. With the same Poisson problem, we will use a 200×200 grid and a convergence tolerance of 1×10^{-8} for all calculations. To study the effects of the number of Monte Carlo histories per MCSA iteration, the weight cutoff was fixed at 1×10^{-4} while the number of histories per iteration was varied from 6,000 to 100,000. It was observed that MCSA would not converge for this problem using the collision estimator with less than 6,000 histories while the expected value estimator permitted convergence with only 2,000 histories. Figure 3.11 gives the number of iterations required to converge for both estimators as a function of the number of histories per iteration. For smaller numbers of histories, the performance of the expected value estimator is significantly better than the collision estimator. This result is valuable in that less transport is required to achieve the same MCSA iterative performance with the expected value estimator, important for situations where transport is expensive (i.e. in domain decomposed calculations). Interstingly, as the number of histories per iteration are increased, the iterative performance with the collision estimator approaches that of the expected value estimator. However, given the performance of the expected value estimator at a fractional number of histories, one should strongly consider its use over using the collision estimator with more histories. The CPU time required to converge is presented in Figure 3.12 and reflects the results of the iterative performance. In general, using the collision estimator is slightly slower overall, but the time per iteration is faster due to the fact that the estimator does not have to cycle through multiple states during the tally procedure.

For the weight cutoff study, the number of histories per iteration was fixed at 40,000 and the weight cutoff varied from 5×10^{-1} down to 1×10^{-10}. Surprisingly, the number of iterations to converge given by Figure 3.13 is effectively invariant to the weight cutoff, only seeing detrimental effects on the number of iterations at a very large weight cutoff. This suggests that a fairly large weight cutoff can be used in practice with both estimators and that the preliminary components of the random walk are more important to MCSA convergence than those that occur later and with lower

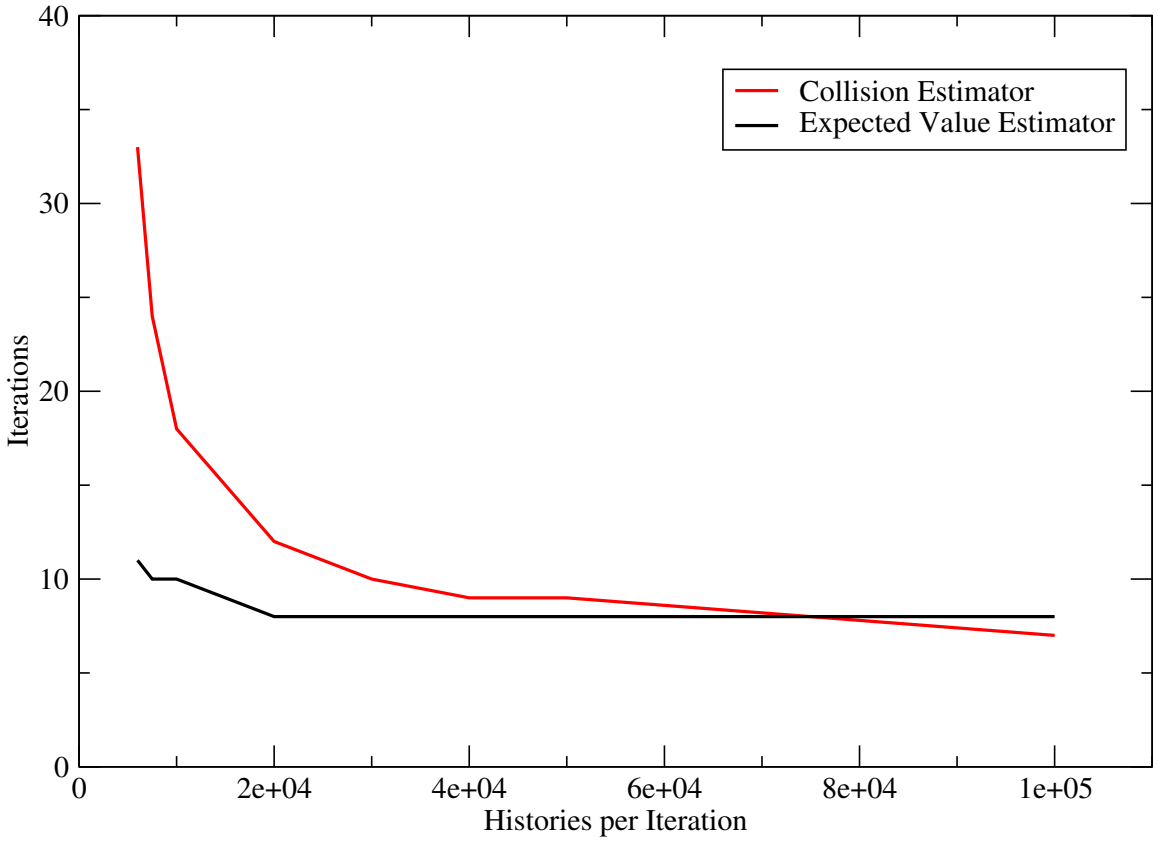

Figure 3.11: **Iterations (s) to converge vs. Monte Carlo histories per MCSA iteration for a 200×200 square mesh and a weight cutoff of 1×10^{-4}.** *For low numbers of histories, the expected value estimator performance is significantly better than the collision estimator. At higher numbers of histories, the estimators become roughly equivalent.*

weight contributions. To further motivate using a larger weight cutoff, Figure 3.14 gives the CPU time need to converge as a function of weight cutoff. As expected, lowering the weight cutoff lengthens the random walk lengths and increases CPU time at no gain of iterative performance. In general, these results suggest using the expected value estimator over the collision estimator with MCSA as well as using a larger weight cutoff.

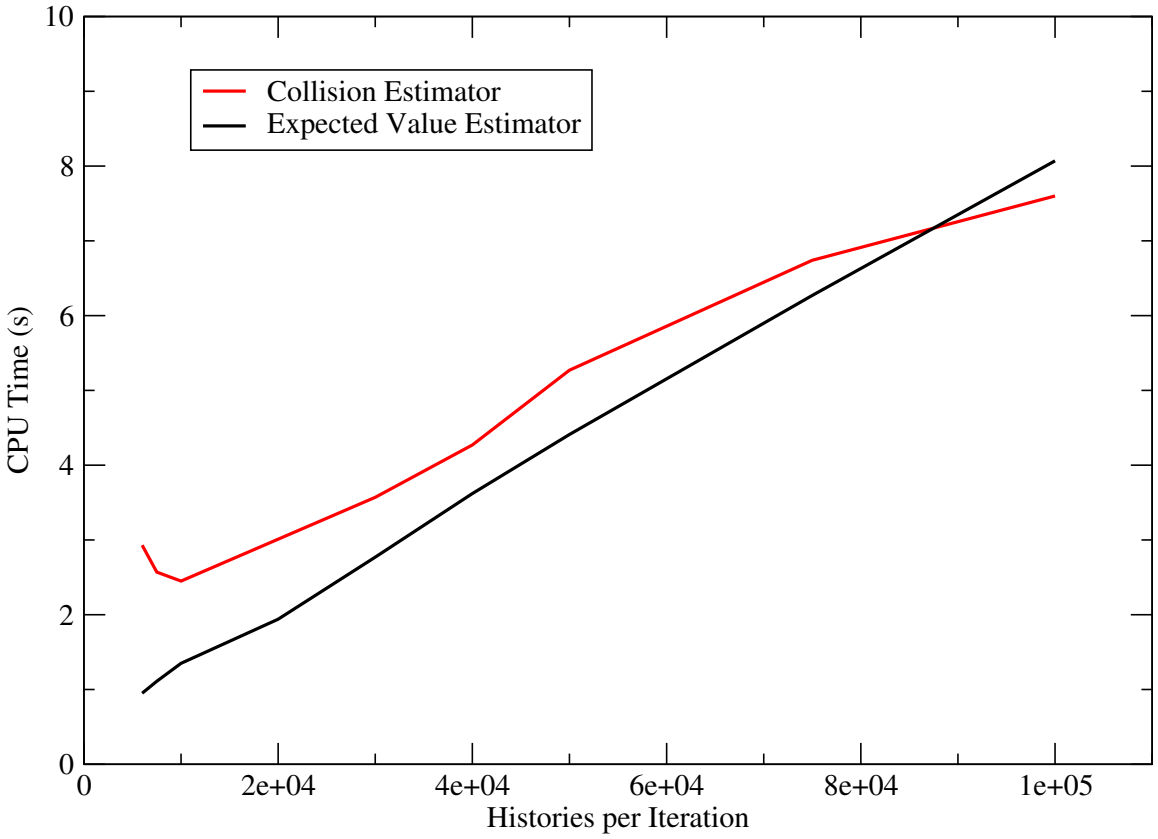

Figure 3.12: **CPU Time (s) to converge vs. Monte Carlo histories per MCSA iteration for a 200×200 square mesh and a weight cutoff of 1×10^{-4}.** *For low numbers of histories, the expected value estimator performance is better than the collision estimator due to a lower iteration count while the actual compute time per iteration is higher.*

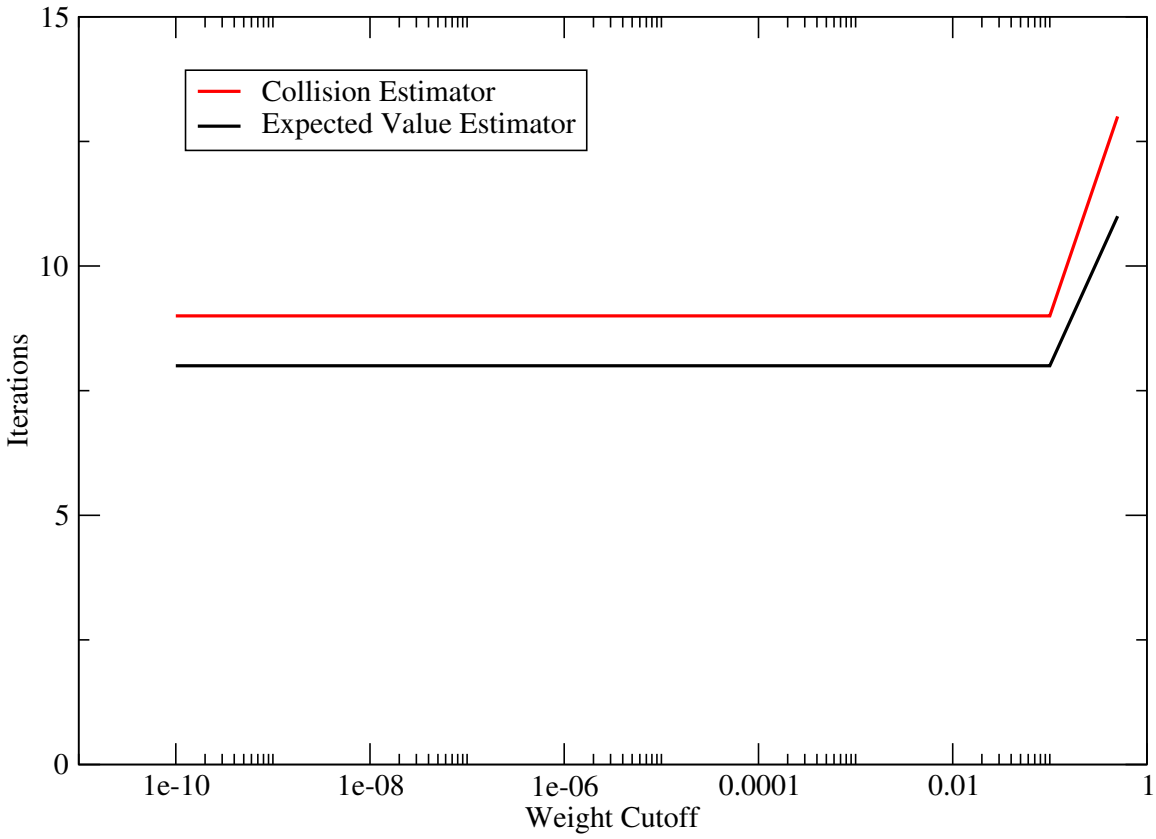

Figure 3.13: **Iterations (s) to converge vs. history weight cutoff for a** 200×200 **square mesh and 40,000 histories.** *For low numbers of histories, the expected value estimator performance is significantly better than the collision estimator. At higher numbers of histories, the estimators become roughly equivalent.*

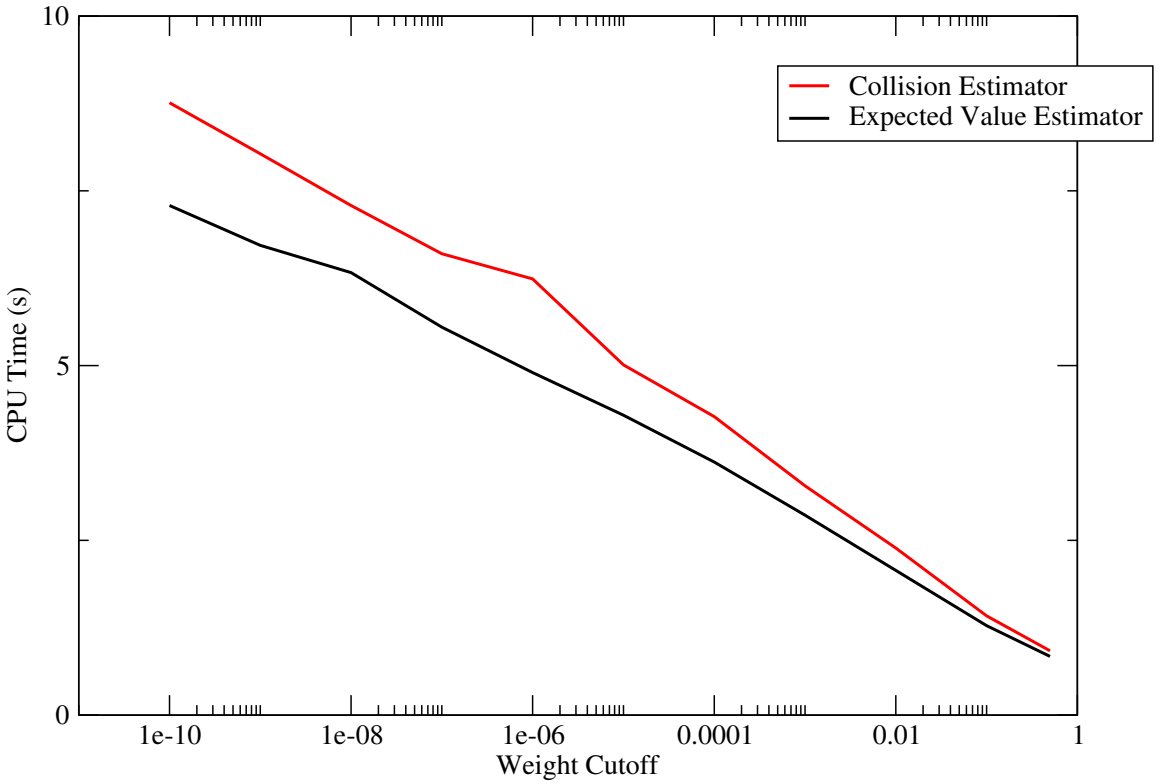

Figure 3.14: **CPU Time (s) to converge vs. history weight cutoff for a 200×200 square mesh and 40,000 histories.** *For low numbers of histories, the expected value estimator performance is better than the collision estimator due to a lower iteration count while the actual compute time per iteration is higher.*

3.9 Variance Reduction Techniques

Variance reduction is a mechanism by which the probabilities and weights of the Monte Carlo problem are modified to improve the performance of a given estimator. In many cases, such modifications will introduce bias into the system. However, using Monte Carlo within an MCSA iterative scheme provides a potential buffer for the solution from this bias as the correction generated by the Monte Carlo solve will contain large statistical error even without variance reduction.

Artificial Absorption

The random walk sequences generated by the adjoint Neumann Ulam decomposition in Eqs (3.38) and (3.39) will continue on indefinitely without the implementation of a weight cutoff procedure (which in itself is effectively a form of biased variance reduction). This is due to the fact that all states to which a stochastic history may move in a given transition exist within the system and have a non-zero weight. A preliminary variance reduction scheme is to introduce *artifical absorption* into the system such that at each transition event, a stochastic history will now have some finite probability of being terminated through absorption as well as transition to other states in the system. This probability, p_{abs}, modifies the probabilities and weights in the adjoint random walk sequence as:

$$p_{ij} = \frac{|h_{ji}|}{\sum_j |h_{ji}|}(1 - p_{abs}) \,, \tag{3.74}$$

and

$$w_{ij} = \frac{sign(h_{ji}) \sum_j |h_{ji}|}{(1 - p_{abs})} \,. \tag{3.75}$$

At each transition step, a history at state i will then sample the probability distribution function generated by Eq (3.74) with an additional absorption state added to the distribution. If the sampling procedure results in a transition to the absorption state, the history is immediately terminated. The advantages of this for residual Monte Carlo are that increasing the absorption probability decreases the length of each random walk (resulting in speedup), while the random walk has a higher weight contribution at every step. This higher weight means that each history is depositing more information near its birth site which will on average be located where the system residual is the largest. An additional advantage of this formulation is that the full Neumann Ulam decomposition is maintained.

To observe the effects of this variance reduction on MCSA solutions, the transient Poisson problem was again solved with the adjoint method, this time with varying values of artificial absorption. A 200×200 grid was used with 40,000 histories at every iteration with the expected value estimator converged to a tolerance of 1×10^{-8}. To reduce the effect of the weight cutoff on this study, the weight cutoff was set to 1×10^{-12} such that it is signicantly more likely that a history will be terminated by absorption rather than weight cutoff. Figure 3.15 gives the number of iterations to converge as a function of the artificial absorption probability. Up to a probability of roughly 0.5, the iterative performance is not significanltly affected with the information lost due to shortened random walks causing a small increase in the number of iterations. At a probability of 0.6, the number of iterations required grows rapidly as the bias creates an MCSA correction that pushes the solution in the wrong direction. Convergence was observed to be lost at probabilities of 0.7 and higher. The random walk length has a strong effect on CPU time as well. Figure 3.16 gives the CPU time in seconds required to converge as a function of artificial absorption probability. Initially, small absorption probabilities not only maintain iterative performance, but also drastically reduced the time required to converge as absorption was more frequent than even a large weight cutoff but the weights themselves were not significantly modified. As the number of iterations grow rapidly, so does the time required to converge to a solution. Based on these results, using a small amount of artificial absorption should provide some CPU speedup while maintaining iterative performance. For the problem presented here, an absorption probability of around 0.25 may be the best choice as it minimizes CPU time.

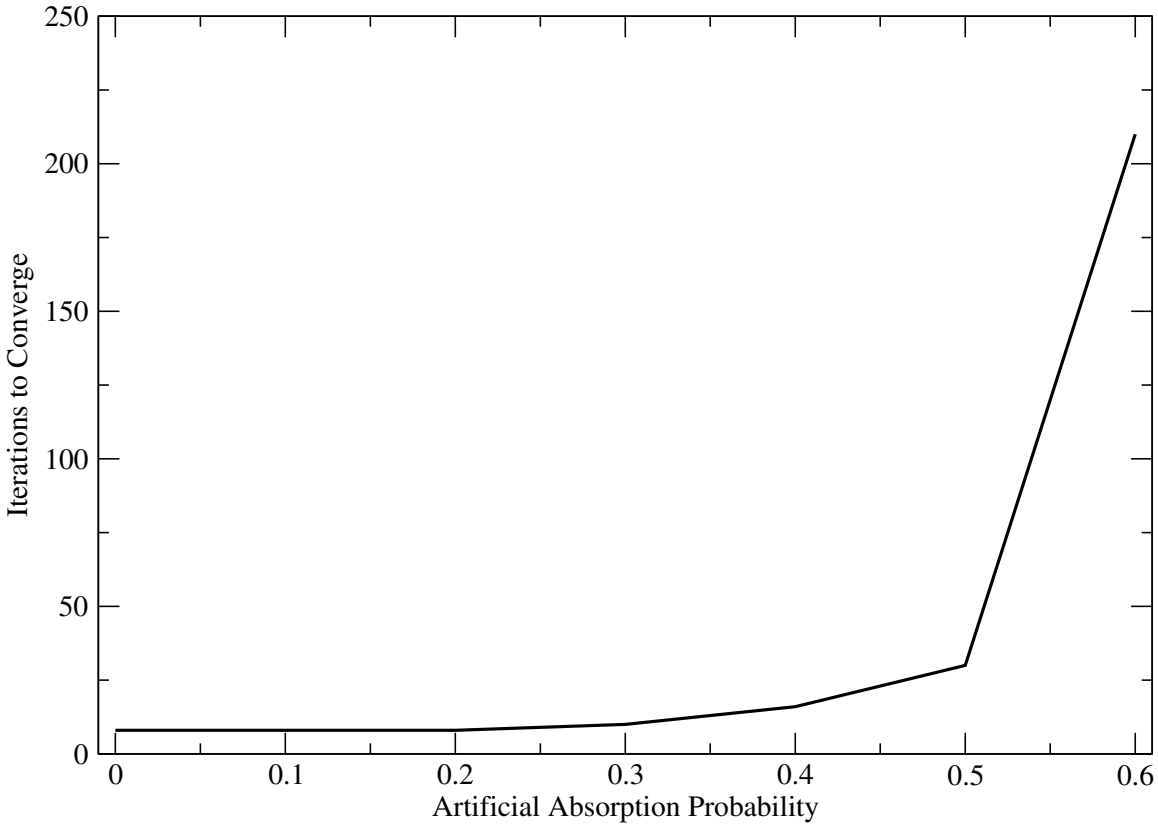

Figure 3.15: **Iterations (s) to converge vs. artificial absorption probability for a 200×200 square mesh and 40,000 histories.** *The addition of absorption is detrimental to iterative performance due to both the shortening of the random walks as well as the modification of the transition weight.*

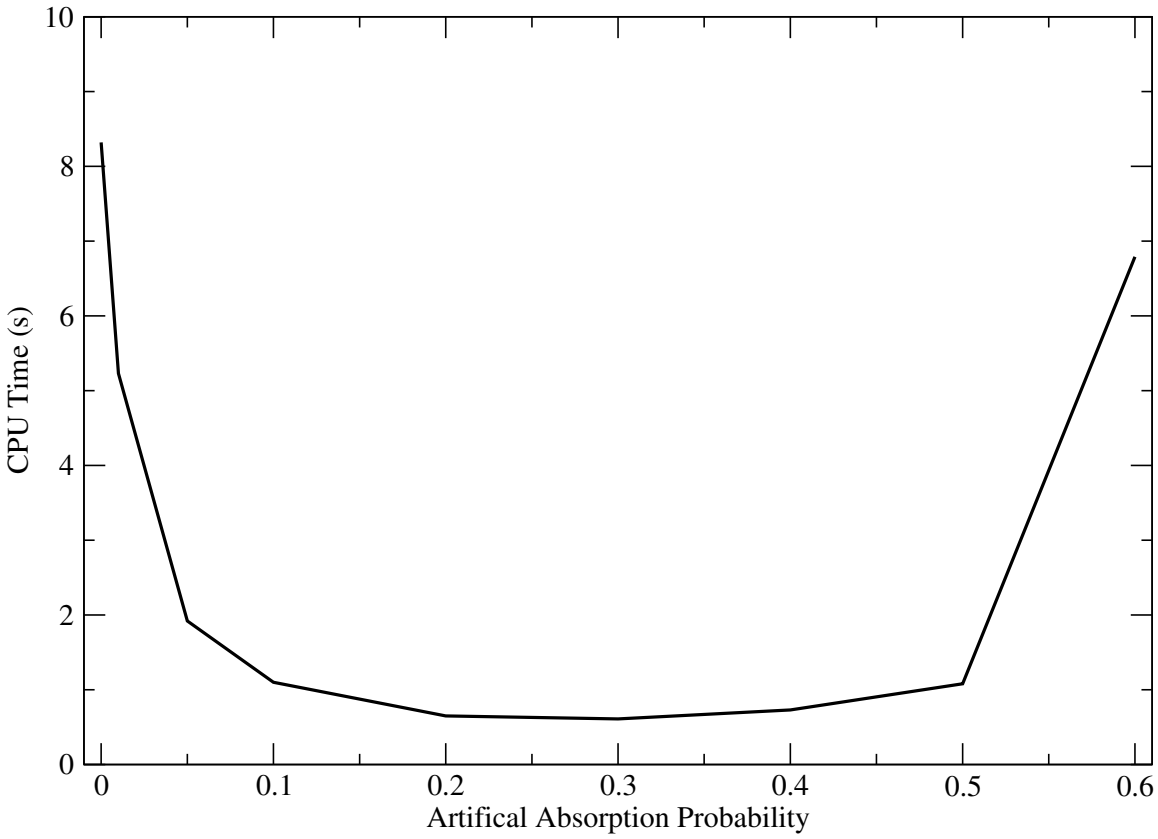

Figure 3.16: **CPU Time (s) to converge vs. artificial absorption probability for a** 200×200 **square mesh and 40,000 histories.** *The reduced random walk length speeds up the calculation with a fixed number of histories. Too much artificial absorption increases iterations rapidly giving an increasing CPU time.*

Reduced Domain Approximation

For scalable parallel implementations and efficient Monte Carlo transport, the domain is required to be sparse. We may find through general preconditioning operations as outlined by the Monte Carlo problem in Eq (3.68) that this sparsity is lost. To alleviate this, we propose modifying the domain sampled by the Monte Carlo method to generate the correction within MCSA by removing terms in the iteration matrix through some predetermined criteria in order to recover sparsity. As the Neumann-Ulam scheme is a stochastic realization of multiple matrix-vector multiplies, the criteria we choose should be based on maintaining those elements that will make the largest contributions to the multiplication and therefore recover as much of the original product as possible. For each row in the preconditioned system, we can then apply the reduced domain approximation to eliminate those elements that either fall below a specified tolerance level, keep the N largest elements in each row, or both giving:

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + \mathbf{r}^{k} \,, \tag{3.76a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \,, \tag{3.76b}$$

$$\hat{\mathbf{A}}\delta\mathbf{x}^{k+1/2} = \mathbf{r}^{k+1/2} \,, \tag{3.76c}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2} \,, \tag{3.76d}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \,, \tag{3.76e}$$

where $\hat{\mathbf{A}}$ in Eq (3.76c) means that the reduced domain approximation has been applied to the iteration matrix in the Monte Carlo method. If enough of the terms in the original iteration matrix have been maintained, the character of the original iteration matrix should be retained and continue to accelerate the iterative scheme. The effects of the reduced domain approximation on systems where it is required due to the explicit preconditioning strategy chosen will be presented in Chapter 4.

Chapter 4

Monte Carlo Solution Methods for the Simplified P_N Equations

The neutron transport problem is complicated. Solutions cover a large phase space and the problems of interest are often geometrically complex, very large, or both, requiring tremendous computational resources to generate an adequate solution. Modern deterministic methods for large scale problems are commonly variants on the discrete ordinates (S_N) method (Evans et al., 2010). For full reactor core neutronics simulations, the S_N method requires potentially trillions of unknown angular flux moments to be computed to achieve good accuracy for the responses of interest (Slaybaugh, 2011). Other forms of the transport problem, including the P_N method, take on a simpler form than the more common S_N methods but lack in accuracy when compared while still requiring considerable computational resources for solutions in multiple dimensions.

In the 1960's, Gelbard developed an ad-hoc multidimensional extension of the simple single dimension planar P_N equations that created a system of coupled, diffusion-like equations known as the simplified P_N (SP_N) equations. Up until around the 1990's, the SP_N method was either widely unknown, widely unused, or combination of both even though numerical studies showed promising results with better solutions than diffusion theory and a significant reduction in computational time over more accurate methods such as discrete ordinates. Why did this happen? A significant problem, pointed out by Brantley and Larsen (Brantley and Larsen, 2000), was that little rigor had been applied to the formulation of the SP_N equations since their derivation through primarily heuristic arguments. Instead, studies at that time focused on simply comparing the results of the method to other contemporary transport solution strategies. In addition, many problems of interest from the literature at the time were either solved using nodal-type methods for reactor-sized problems or S_N-type methods for benchmark problems with intricate material configurations and potentially large flux gradients over small spatial domains.

So why reconsider the SP_N equations? Starting in the 1990's and primarily due to Larsen and his colleagues, the SP_N equations have been given a more rigorous treatment with both variational and asymptotic derivations performed as a means

of verification. In addition, these equations have been more rigorously studied as solution methods to MOX fuel problems and have been shown to provide accurate solutions. With this mathematical literature to provide a solid numerical footing for the method, we look at its application to today's challenge problems in full reactor core transport. The reduction in numerical complexity of current full core deterministic solution methods using the S_N approximation could mean significant savings in both compute time and memory required. In addition, the characteristics of the solution to the transport problem for a steady state reactor core permit diffusion theory to be used; a staple of the nuclear industry since its inception. Therefore, if diffusion theory is applicable, then finer grained solutions that capture more of the physics contained in the transport equation should be possible with the SP_N method. In doing so, we also expect from the literature to obtain computed responses on the order of accuracy we would expect from an appropriately discretized S_N method at a fraction of the cost.

To further motivate moving in this direction, recent developments in the Exnihilo neutronics package at Oak Ridge National Laboratory have permitted generation of the SP_N system of equations for detailed full core reactor models (Evans, 2013). By fully forming these equations and formulating them as a linear algebra problem instead of using the explicit iterative methods of the past, we now have access to all of the modern advancements in computational linear algebra including Krylov solvers for asymmetric systems and preconditioning methods such as algebraic multigrid. This leads us to then explore the applicability of our work in discrete Monte Carlo methods for linear systems as a possible solution method for the SP_N equations. If formulated correctly, we hypothesize that significant improvement in usage of computational resources may be observed compared to modern solution techniques such as those suggested due to the form of the matrices generated by the discretization. In addition, solving the SP_N equations in this way also breaks away from the S_N forms of parallelism where spatial parallelism is achieved by an efficient parallel sweep, angular efficiency achieved by pipe-lining, and energy parallelism achieved by decoupling the groups. With the SP_N equations as a full matrix system, we now can parallelize the problem as prescribed by the linear solver, which may be significantly more scalable than current S_N transport practices.

In this chapter, we derive the SP_N equations, closely following the work of Evans[1], in order to gain full understanding of the underlying system and its potential behavior

[1]The SP_N derivations in this chapter are heavily based on those presented by Evans in (Evans, 2013)

in a Monte Carlo context. From the P_N equations, we apply a set of approximations to yield the multi-dimensional, multi-group SP_N equations for fixed source problems. Using the fully-formed matrix for the transport problem, we explore solutions to the SP_N equations using Monte Carlo Synthetic Acceleration using a light water reactor fuel assembly criticality calculation as the driving problem.

4.1 The Neutron Transport Equation

As a starting point we define the time-independent neutron transport equation (Lewis, 1993):

$$\hat{\Omega} \cdot \vec{\nabla}\psi(\vec{r},\hat{\Omega},E) + \sigma(\vec{r},E)\psi(\vec{r},\hat{\Omega},E) = \iint \sigma_s(\vec{r},E' \rightarrow E, \hat{\Omega}' \cdot \hat{\Omega})\psi(\vec{r},\hat{\Omega}',E')d\Omega' dE' + q(\vec{r},\hat{\Omega},E)\,, \quad (4.1)$$

with the variables defined as:

- $\vec{r}$ - neutron spatial position
- $\hat{\Omega}$ - neutron streaming direction with radial component μ and azimuthal component ω
- $\hat{\Omega}' \cdot \hat{\Omega} = \mu_0$ is the angle of scattering
- E - neutron energy
- $\psi(\vec{r},\hat{\Omega},E)$ - angular flux
- $\sigma(\vec{r},E)$ - total interaction cross section
- $\sigma_s(\vec{r},E' \rightarrow E,\hat{\Omega}')$ - probability of scattering from direction $\hat{\Omega}'$ into an angular domain $d\hat{\Omega}'$ about the direction $\hat{\Omega}$ and from energy E' to an energy domain dE' about energy E
- $q(\vec{r},\hat{\Omega},E)$ - external source of neutrons.

For this work, it is sufficient to formulate Eq (4.1) in 1-dimensional Cartesian geometry:

$$\mu\frac{\partial}{\partial x}\psi(x,\mu,E) + \sigma(x,E)\psi(x,\mu,E) = \iint \sigma_s(x,E' \rightarrow E, \hat{\Omega}' \cdot \hat{\Omega})\psi(x,\hat{\Omega}',E')d\Omega' dE' + \frac{q(x,E)}{4\pi}\,, \quad (4.2)$$

where the angular component of the solution is no longer dependent on the azimuthal direction of travel and an isotropic source of neutrons is assumed. In addition, for fission systems the eigenvalue form of the transport equation is:

$$\hat{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\hat{\Omega},E)+\sigma(\vec{r},E)\psi(\vec{r},\hat{\Omega},E)= \iint\sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega})\psi(\vec{r},\hat{\Omega}',E')d\Omega'dE'+ \frac{1}{k}\chi(E)\iint\nu\sigma_f(\hat{r},E')\psi(\vec{r},\hat{\Omega}',E')d\Omega'dE'+q(\vec{r},\hat{\Omega},E)\,, \quad (4.3)$$

with the additional variables defined as

- k - multiplication factor
- $\chi(E)$ - fission neutron energy spectrum
- ν - average number of neutrons per fission
- $\sigma_f(r,E')$ - fission cross section .

In 1-dimensional Cartesian geometry, Eq (4.3) becomes:

$$\mu\frac{\partial}{\partial x}\psi(x,\mu,E)+\sigma(x,E)\psi(x,\mu,E)= \iint\sigma_s(x,E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega})\psi(x,\hat{\Omega}',E')d\Omega'dE'+ \frac{1}{k}\chi(E)\iint\nu\sigma_f(x,E')\psi(x,\hat{\Omega}',E')d\Omega'dE'+\frac{q(x,E)}{4\pi}\,. \quad (4.4)$$

4.2 Derivation of the Monoenergetic SP_N Equations

The P_N equations as derived in Appendix A give $N+1$ coupled first-order equations capturing the spatial and angular-dependence of the solution. In multiple dimensions, the equation set becomes large and coupled not only through angular moments but also through the spatial variables. As a simpler alternative to multidimensional P_N solutions, Gelbard recognized in 1960 that the planar P_N equations could be simplified and applied an ad-hoc method to extend them to multiple dimensions, yielding the SP_N equations. These equations are not only fewer in number, but also take on a diffusion-like form while maintaining the angular character of the flux, making them amenable to solutions with modern diffusion methods.

First, the P_N equations can be simplified to $(N+1)/2$ second-order equations by solving for the n^{th} Legendre flux moment in the odd-order equations:

$$\phi_n = \frac{1}{\Sigma_n}\left[q\delta_{no} - \frac{\partial}{\partial x}\left(\frac{n}{2n+1}\phi_{n-1} + \frac{n+1}{2n+1}\phi_{n+1}\right)\right], \tag{4.5}$$

for $n = 1, 3, \cdots, N$ and $\delta_{no} = 0\ \forall n \neq 0$. We can insert the odd moments into Eq (A.31) to get a reduced group of equations for the even moments:

$$\begin{aligned} -\frac{\partial}{\partial x}\Bigg[\frac{n}{2n+1}\frac{1}{\Sigma_{n-1}}\frac{\partial}{\partial x}\left(\frac{n-1}{2n-1}\phi_{n-2} + \frac{n}{2n-1}\phi_n\right) \\ + \frac{n+1}{2n+1}\frac{1}{\Sigma_{n+1}}\frac{\partial}{\partial x}\left(\frac{n+1}{2n+3}\phi_n + \frac{n+2}{2n+3}\phi_{n+2}\right)\Bigg] \\ + \Sigma_n\phi_n = q\delta_{n0} \qquad n = 0, 2, 4, \cdots, N\,. \end{aligned} \tag{4.6}$$

Immediately, we note the diffusion-like nature of Eq (4.6) as compared to the original P_N equations. To extend these equations to multiple dimensions, Gelbard simply replaced the planar spatial derivatives in the reduced set of equations with general multidimensional gradient operators:

$$\begin{aligned} -\nabla\cdot\Bigg[\frac{n}{2n+1}\frac{1}{\Sigma_{n-1}}\nabla\left(\frac{n-1}{2n-1}\phi_{n-2} + \frac{n}{2n-1}\phi_n\right) \\ + \frac{n+1}{2n+1}\frac{1}{\Sigma_{n+1}}\nabla\left(\frac{n+1}{2n+3}\phi_n + \frac{n+2}{2n+3}\phi_{n+2}\right)\Bigg] \\ + \Sigma_n\phi_n = q\delta_{n0} \qquad n = 0, 2, 4, \cdots, N\,, \end{aligned} \tag{4.7}$$

yielding a multidimensional set of $(N+1)/1$ angular coupled equations defined as the SP_N equations. As with the P_N equations, we provide closure to this set of equations

with $\phi_{N+1} = 0$. As a concrete example, we will consider the SP_7 equations:

$$-\nabla \cdot \frac{1}{3\Sigma_1}\nabla(\phi_0 + 2\phi_2) + \Sigma_0\phi_0 = q \tag{4.8a}$$

$$-\nabla \cdot \left[\frac{2}{15\Sigma_1}\nabla(\phi_0 + 2\phi_2) + \frac{3}{35\Sigma_3}\nabla(3\phi_2 + 4\phi_4)\right] + \Sigma_2\phi_2 = 0 \tag{4.8b}$$

$$-\nabla \cdot \left[\frac{4}{63\Sigma_3}\nabla(3\phi_2 + 4\phi_4) + \frac{5}{99\Sigma_5}\nabla(5\phi_4 + 6\phi_6)\right] + \Sigma_4\phi_4 = 0 \tag{4.8c}$$

$$-\nabla \cdot \left[\frac{6}{143\Sigma_5}\nabla(5\phi_4 + 6\phi_6) + \frac{7}{195\Sigma_7}\nabla(7\phi_6)\right] + \Sigma_6\phi_6 = 0\,. \tag{4.8d}$$

To further modify these equations, we can use a change of variables to create a new group of equations such that the gradients are operating on a single vector:

$$u_1 = \phi_0 + 2\phi_2 \tag{4.9a}$$

$$u_2 = 3\phi_2 + 4\phi_4 \tag{4.9b}$$

$$u_3 = 5\phi_4 + 6\phi_6 \tag{4.9c}$$

$$u_4 = 7\phi_6\,, \tag{4.9d}$$

or equivalently:

$$\phi_0 = u_1 - \frac{2}{3}u_2 + \frac{8}{15}u_3 - \frac{16}{35}u_4 \tag{4.10a}$$

$$\phi_2 = \frac{1}{3}u_2 - \frac{4}{15}u_3 + \frac{8}{35}u_4 \tag{4.10b}$$

$$\phi_4 = \frac{1}{5}u_3 - \frac{6}{35}u_4 \tag{4.10c}$$

$$\phi_6 = \frac{1}{7}u_4\,. \tag{4.10d}$$

When substituted into Eq (4.8), these terms give:

$$-\nabla\cdot\frac{1}{3\Sigma_1}\nabla u_1+\Sigma_0\left[u_1-\frac{2}{3}u_2+\frac{8}{15}u_3-\frac{16}{35}u_4\right]=-q \tag{4.11a}$$

$$-\nabla\cdot\left[\frac{2}{15\Sigma_1}\nabla u_1+\frac{3}{35\Sigma_3}\nabla u_2\right]+\Sigma_2\left[\frac{1}{3}u_2-\frac{4}{15}u_3+\frac{8}{35}u_4\right]=0 \tag{4.11b}$$

$$-\nabla\cdot\left[\frac{4}{63\Sigma_3}\nabla u_2+\frac{5}{99\Sigma_5}\nabla u_3\right]+\Sigma_4\left[\frac{1}{5}u_3-\frac{6}{35}u_4\right]=0 \tag{4.11c}$$

$$-\nabla\cdot\left[\frac{6}{143\Sigma_5}\nabla u_3+\frac{7}{195\Sigma_7}\nabla u_4\right]+\Sigma_6\left[\frac{1}{7}u_4\right]=0\,. \tag{4.11d}$$

If we rearrange Eq (4.11) such that only one divergence operation is present in each equation, we can formulate this as a matrix system of 4 equations in the case of the SP_7 approximation:

$$-\nabla\cdot D_n\nabla u_n+\sum_{m=1}^{4}A_{nm}u_m=q_n\qquad n=1,2,3,4\,, \tag{4.12}$$

with $\mathbf{u}$ the vector of solution variables:

$$\mathbf{u}=(u_1\quad u_2\quad u_3\quad u_4)^T\,, \tag{4.13}$$

$\mathbf{D}$ the vector of effective diffusion coefficients:

$$\mathbf{D}=\left(\frac{1}{3\Sigma_1}\quad\frac{1}{7\Sigma_3}\quad\frac{1}{11\Sigma_5}\quad\frac{1}{15\Sigma_7}\right)^T\,, \tag{4.14}$$

$\mathbf{q}$ the vector of source terms where the 0^{th} moment source has now been distributed through the system:

$$\mathbf{q}=(q\quad-\frac{2}{3}q\quad\frac{8}{15}q\quad-\frac{16}{35}q)^T\,, \tag{4.15}$$

and $\mathbf{A}$ a matrix of angular scattering terms:

$$\mathbf{A}=\begin{bmatrix}(\Sigma_0) & (-\frac{2}{3}\Sigma_0) & (\frac{8}{15}\Sigma_0) & (-\frac{16}{35}\Sigma_0)\\ (-\frac{2}{3}\Sigma_0) & (\frac{4}{9}\Sigma_0+\frac{5}{9}\Sigma_2) & (-\frac{16}{45}\Sigma_0-\frac{4}{9}\Sigma_2) & (\frac{32}{105}\Sigma_0+\frac{8}{21}\Sigma_2)\\ (\frac{8}{15}\Sigma_0) & (-\frac{16}{45}\Sigma_0-\frac{4}{9}\Sigma_2) & (\frac{64}{225}\Sigma_0+\frac{16}{45}\Sigma_2+\frac{9}{25}\Sigma_4) & (-\frac{128}{525}\Sigma_0-\frac{32}{105}\Sigma_2-\frac{54}{175}\Sigma_4)\\ (-\frac{16}{35}\Sigma_0) & (\frac{32}{105}\Sigma_0+\frac{8}{21}\Sigma_2) & (-\frac{128}{525}\Sigma_0-\frac{32}{105}\Sigma_2-\frac{54}{175}\Sigma_4) & (\frac{256}{1225}\Sigma_0+\frac{64}{245}\Sigma_2+\frac{324}{1225}\Sigma_4+\frac{13}{49}\Sigma_6)\end{bmatrix}. \tag{4.16}$$

Note that the term $\sum_{m=1}^{4} A_{nm}u_m$ in Eq (4.12) couples the moments in each equation while the diffusive term in each equation is only for a single 'pseudo-moment' u_n. As noted by Evans (Evans, 2013), lower order SP_N approximations can be generated by setting higher order even moments in this system to zero (e.g. $\phi_6 = \phi_4 = 0$ yields the SP_3 equations). Boundary conditions for these equations are provided in Appendix C. The multigroup form is presented in Appendix **??** and the spatial discretization in Appendix D.

Eigenvalue Form of the SP_N Equations

For criticality problems, the space, angle, and energy discretizations for the SP_N equations can be applied to the general eigenvalue from of the transport equation given by Eq (4.3). We can readily replace the fixed source in Eq (B.7) with the fission source derived in § A.4:

$$-\nabla\cdot\left[\frac{n}{2n+1}\boldsymbol{\Sigma}_{\mathbf{n-1}}{}^{-1}\nabla\left(\frac{n-1}{2n-1}\boldsymbol{\Phi}_{\mathbf{n-2}}+\frac{n}{2n-1}\boldsymbol{\Phi}_{\mathbf{n}}\right)\right.$$
$$\left.+\frac{n+1}{2n+1}\boldsymbol{\Sigma}_{\mathbf{n+1}}{}^{-1}\nabla\left(\frac{n+1}{2n+3}\boldsymbol{\Phi}_{\mathbf{n}}+\frac{n+2}{2n+3}\boldsymbol{\Phi}_{\mathbf{n+2}}\right)\right]$$
$$+\boldsymbol{\Sigma}_{\mathbf{n}}\boldsymbol{\Phi}_{\mathbf{n}}=\frac{1}{k}\mathbf{F}\boldsymbol{\Phi}_{\mathbf{n}}\delta_{n0}\qquad n=0,2,4,\cdots,N\,. \quad (4.17)$$

with the fission matrix $\mathbf{F}$ given by Eq (A.46). We again apply the change of variables to yield the set of pseudo-moments $\mathbb{U}_n$ where now the application of the fission matrix to the moment vectors will be expanded into a set of block matrices in identical fashion to the scattering matrices.

It should be noted here that with respect to a general MCSA scheme, the introduction of the fission in the system will only affect the source vector in the linear system. The linear operator, and therefore overall MCSA performance will be dictated by streaming and scattering as defined on the left-hand side.

4.3 Spectral Analysis of the SP_N Equations

Before we can move on to solving the SP_N equations with both conventional and Monte Carlo methods, we must first understand their spectral character to ensure that MCSA will be applicable. As presented in Chapter 3, the spectral radius of the iteration matrix must be less than unity to ensure convergence. Therefore, we will

compute the spectral radius of the iteration matrix formed by the SP_N equations while parametrically varying the SP_N order, the P_N order, the number of energy groups, and upscattering/downscattering between energy groups. For each problem in the parameter study, the values given in Table 4.1 were fixed for each spatial location and energy group. Mesh element sizes were the same in all cardinal directions with

Parameter	Value
Mesh Element Size	1.0
Mesh Elements	$4 \times 4 \times 4$
Reflecting Boundaries	True
Materials	1
Source Strength	1.0
Total Cross Section	5.0
In-Group Cross Section	0.25
Downscatter Cross Section	1.0
Upscatter Cross Section	0.1
Eigenvalue Solver Tolerance	1.0×10^{-8}

Table 4.1: **Fixed parameters for the SP_N spectral radius parameter study.** *The parameters were fixed for each spatial location and energy group.*

reflecting conditions on each boundary. For the energy parameter, 1 and 10 energy groups were used with upscatter/downscatter varied for the 10 group case. For the downscatter cases, all groups could scatter to all lower energy groups with the cross sections given in Table 4.1 while for the upscatter case all groups could scatter to all other energy groups with different cross sections for upscatter and downscatter used. A uniform isotropic source of neutrons is given in each group as well.

The matrices generated by the discretization of this problem are sparse with a block-based form as dictated by Eq (B.8). Figure 4.1 gives the sparsity pattern for a monoenergetic SP_7 discretization while Figures 4.2 and 4.3 give the sparsity patterns for the same problem for 10 energy groups without and with upscatter respectively. For these figures, the parameters in Table 4.1 with the number of spatial elements reduced to $2 \times 2 \times 2$ to show the blocks generated by the discretization. We note a few key features of the sparsity plots. The first is that for multigroup problems without full upscatter and downscatter (i.e. Figure 4.2), the resulting matrix is asymmetric and therefore a linear solver that can handle asymmetric linear systems is required. Nearly all problems of interest will not have full upscattering or downscattering. Second we note the largely diagonal character of these systems, although the blocks from

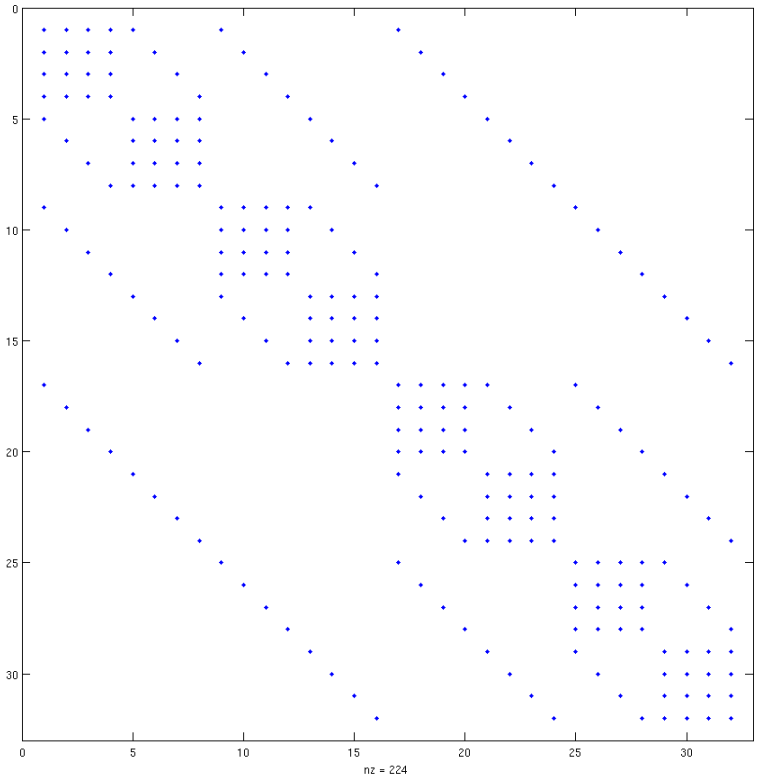

Figure 4.1: **Sparsity pattern for 1-group SP_7 discretization.** *A $2\times2\times2$ element mesh was used to show detail of the blocks formed by the discretization.*

Eq (B.9) are readily apparent. Our first attempt at preconditioning this system will be to use the point Jacobi preconditioning from §3.5 due to this diagonal form.

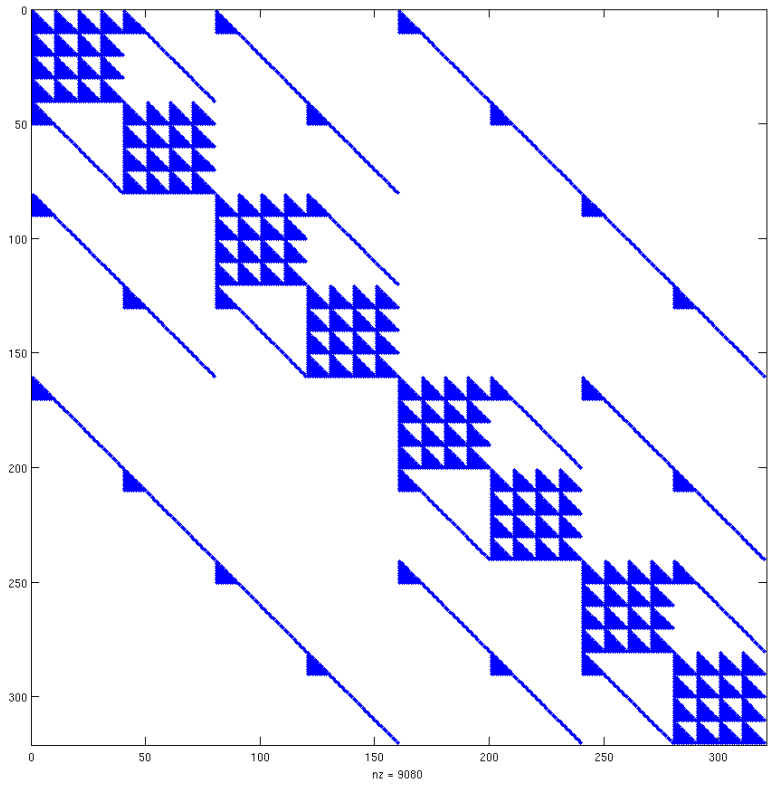

Figure 4.2: **Sparsity pattern for 10-group SP_7 discretization with downscatter only.** *A $2 \times 2 \times 2$ element mesh was used to show detail of the blocks formed by the discretization.*

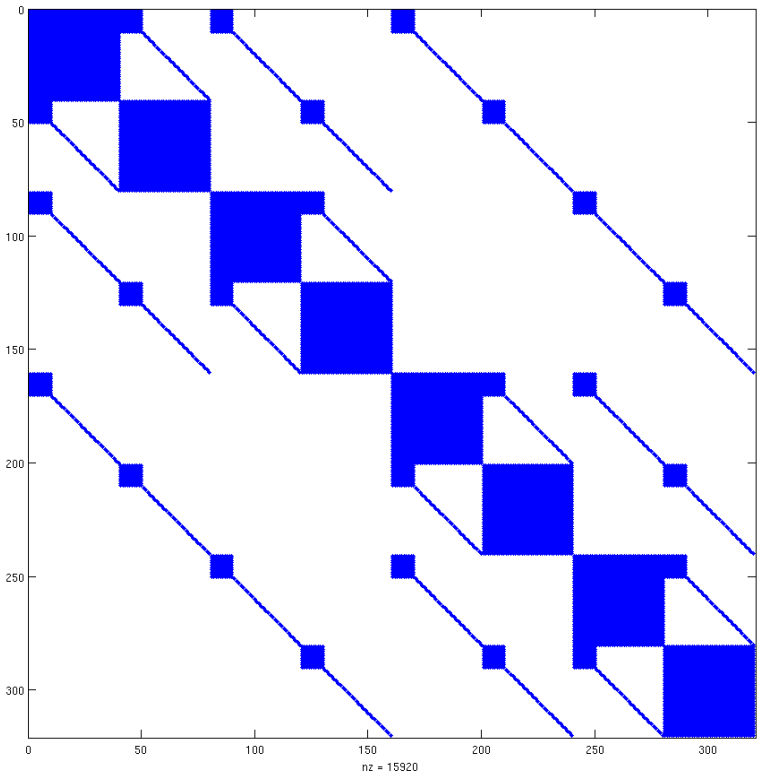

Figure 4.3: **Sparsity pattern for 10-group SP_7 discretization with downscatter and upscatter.** *A $2 \times 2 \times 2$ element mesh was used to show detail of the blocks formed by the discretization.*

Point Jacobi Spectral Analysis Results

Spectral radius computations were performed for the cases described above for the point Jacobi preconditioned iteration matrix $\mathbf{H} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{A}$ with $\mathbf{M} = diag(\mathbf{A})$. Table 4.2 gives the results for the 1-group case, Table 4.3 for the 10-group case with full downscatter only and Table 4.4 for the 10-group case with full downscatter and full upscatter. It is readily apparent from the tabulated data that point Jacobi

		SP_N Order			
		1	**3**	**5**	**7**
	0	0.0635	0.6722	1.3144	1.976
	1	0.0666	0.6728	1.3141	1.9755
P_N Order	**3**	0.0666	0.6822	1.3141	1.9755
	5	0.0666	0.6822	1.3278	1.9847
	7	0.0666	0.6822	1.3278	1.9917

Table 4.2: **Spectral radius results for the point Jacobi preconditioned iteration matrix with 1 energy group.**

		SP_N Order			
		1	**3**	**5**	**7**
	0	0.0655	0.677	1.32	1.982
	1	0.071	0.6777	1.319	1.982
P_N Order	**3**	0.071	0.687	1.327	1.9872
	5	0.071	0.687	1.336	1.997
	7	0.071	0.687	1.336	1.9995

Table 4.3: **Spectral radius results for the point Jacobi preconditioned iteration matrix with 10 energy groups and full downscatter.**

preconditioning is insufficient. For problems of order SP_5 and SP_7, the method will not converge at all and for the upscatter case with SP_3 the spectral radius is still quite large for large P_N orders and therefore convergence is expected to be slow. These eigenvalues signal a need for a better preconditioning strategy to both ensure and improve convergence for Monte Carlo methods.

Block Jacobi Preconditioning

If Monte Carlo methods are to be used to solve the SP_N system of equations, a different preconditioning strategy is required in order to ensure convergence for systems of all

		SP_N Order			
		1	**3**	**5**	**7**
	0	0.7283	0.81	1.47	2.1446
	1	0.7317	0.8	1.46	2.1368
P_N Order	**3**	0.7317	0.91	1.526	2.2274
	5	0.7317	0.91	1.5344	2.2562
	7	0.7317	0.91	1.5345	2.2842

Table 4.4: **Spectral radius results for the point Jacobi preconditioned iteration matrix with 10 energy groups, full downscatter and full upscatter.**

SP_N and P_N orders with arbitrary energy group structures. As another means of achieving this, we look back to the sparsity plots we generated in Figures 4.1, 4.2 and 4.3 as well as the multigroup SP_N equations. Initially, the diagonal character of the system led us to try point Jacobi preconditioning with only marginal results. From the sparsity plots we note the block structure that ultimately arises from the multigroup scattering matrices and their insertion into Eq (B.9). When full upscatter and downscatter are used the resulting blocks are completely dense while only downscatter gives a lower triangular scattering matrix and the block structure shown in Figure 4.2.

Based on this both block and diagonally dominant structure for matrices formed by the general multigroup SP_N equations, we instead choose *block Jacobi* preconditioning as a left preconditioner for the system. Like point Jacobi preconditioning, block Jacobi preconditioning extracts the diagonal elements of the matrix as the preconditioner where now the elements extracted are the blocks on the diagonal as shown on the left side of Figure 4.4. Shown on the right side of Figure 4.4, inversion of the preconditioner is trivial with each diagonal block inverted separately. For the SP_N equations, Eq (B.9) gives a block size of $N_g \times (N+1)/2$. The inversion of this preconditioner is trivial as shown on the right side of Figure 4.4. Each block can be inverted individually and combined to form the inverse. In addition, in the limit of a block size of one, the block Jacobi method reduces to the point Jacobi method. For high performance implementations this has several attractive properties. First, the blocks in the matrix come from the energy/angle discretization of the transport equation as given by Eq (B.9). Each block on the diagonal is bound to a mesh element in the system (note there are 8 blocks on the diagonal in each of the sparsity patterns with a mesh of $2 \times 2 \times 2$) and therefore we expect the matrix elements forming the block to be entirely local. Second, these blocks are typically dense and nearly lower triangular for many

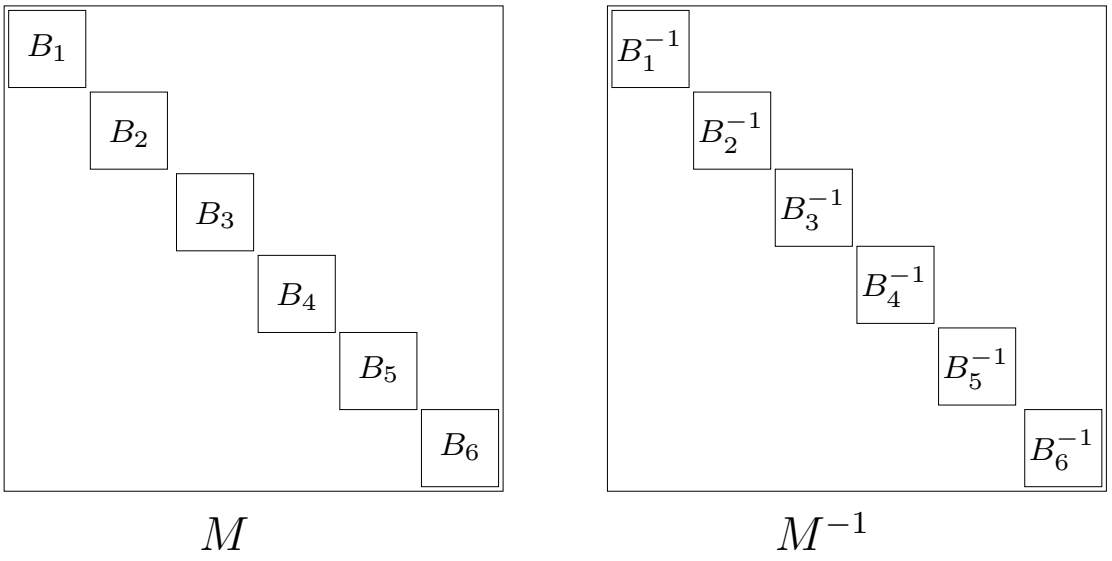

Figure 4.4: **Block Jacobi preconditioning strategy used for the SP_N equations.** *Left: The preconditioner is formed by the diagonal blocks of the matrix. Right: Inversion of the preconditioner is trivial and decoupled by block.*

transport problems meaning that established dense matrix methods can be used for fast inversion.

Block Jacobi Spectral Analysis Results

Spectral radius computations for the block Jacobi preconditioned iteration matrix were performed for the same problems as the point Jacobi preconditioned case. Table 4.5 gives the results for the 1-group case, Table 4.6 for the 10-group case with full downscatter only and Table 4.7 for the 10-group case with full downscatter and full upscatter. A block size of $N_g \times (N+1)/2$ was used for the preconditioner in all cases.

From the tabulated block Jacobi data it is clear that this is a viable preconditioning choice for all SP_N problems in term of Monte Carlo solution methods. All cases were observed to have a spectral radius below unity and often significantly smaller than that, greatly improving convergence rates over the point Jacobi preconditioned problem. Based on these results, the block Jacobi method should be the first preconditioning strategy applied to more complicated problems representative of physical systems.

		SP_N Order			
		1	**3**	**5**	**7**
	0	0.0635	0.1269	0.1444	0.1513
	1	0.0666	0.1315	0.1474	0.1534
P_N Order	**3**	0.0666	0.1365	0.154	0.1592
	5	0.0666	0.1365	0.1562	0.163
	7	0.0666	0.1365	0.1562	0.164

Table 4.5: **Spectral radius results for the block Jacobi preconditioned iteration matrix with 1 energy group.**

		SP_N Order			
		1	**3**	**5**	**7**
	0	0.0647	0.1275	0.1449	0.1514
	1	0.0686	0.1338	0.1484	0.1547
P_N Order	**3**	0.0687	0.1399	0.1582	0.1625
	5	0.0692	0.1399	0.1582	0.1657
	7	0.0678	0.1393	0.1624	0.166

Table 4.6: **Spectral radius results for the block Jacobi preconditioned iteration matrix with 10 energy groups and full downscatter.**

		SP_N Order			
		1	**3**	**5**	**7**
	0	0.1887	0.2267	0.2285	0.2286
	1	0.4535	0.5044	0.5045	0.5045
P_N Order	**3**	0.4535	0.6453	0.6506	0.6506
	5	0.4535	0.6453	0.6802	0.6818
	7	0.4535	0.6453	0.6802	0.6927

Table 4.7: **Spectral radius results for the block Jacobi preconditioned iteration matrix with 10 energy groups, full downscatter and full upscatter.**

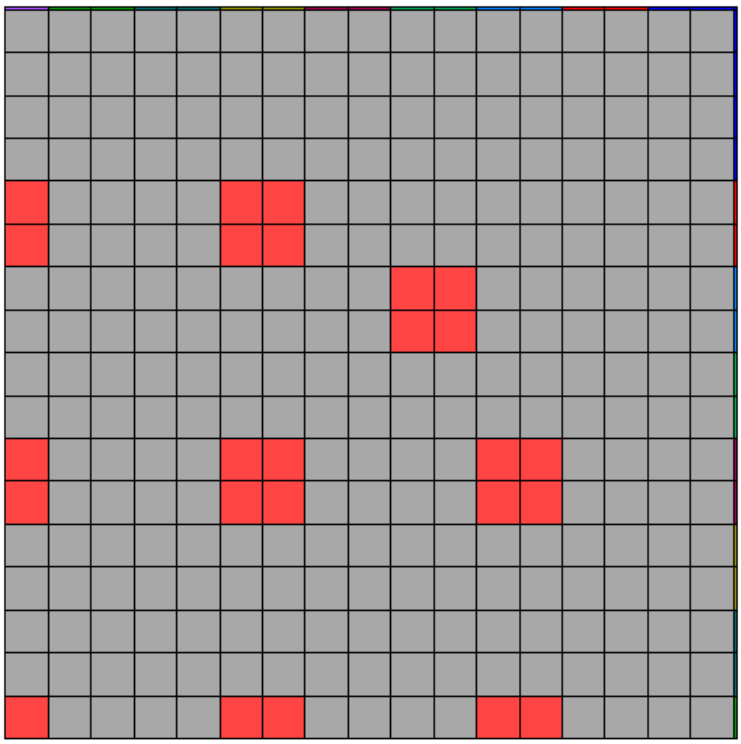

Figure 4.5: **Fuel assembly mesh and geometry cross section.** *Reflecting boundaries are used on the left and lower boundaries to create a complete* 17×17 *assembly geometry. Gray regions are homogenized fuel and red regions are homogenized moderator. Each fuel pin is resolved by a* 2×2 *mesh.*

4.4 Fuel Assembly Criticality Calculations

Fuel assembly calculations are a critical piece of nuclear engineering infrastructure for reactor core analysis and design. At this level, individual fuel pins may be resolved at fine resolution in a variety of configurations. As a sophisticated problem of interest to push the limits of MCSA, a hot zero-power 17×17 pin assembly will be used with varying energy group structure and SP_N discretization in a criticality calculation. A cross section along the vertical axis showing homogenized fuel pin materials and the associated grid is given by Figure 4.5 while a cross section of the materials configuration along the horizontal axis is given by Figure 4.6. A detailed view of the assembly bottom is given in Figure 4.7. On the top and bottom of the assembly, vacuum conditions are used as well as on the top and right boundaries in Figure 4.5. Reflecting conditions are used on the left and bottom boundaries of Figure 4.5, effectively giving a representation of one quarter of the assembly. Significant geometric details are

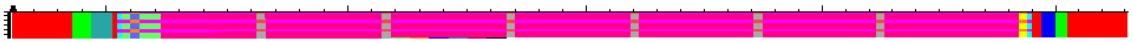

Figure 4.6: **Fuel assembly geometry cross section.** *The geometry is subdivided along the axial direction into 50 zones spaced to capture material boundaries. Important details include spacer grids along the length of the fuel pins and reactor core structures on the top and bottom of the assembly. Lighter purple material in the center of the assembly is moderator and darker purple/red material is fuel.*

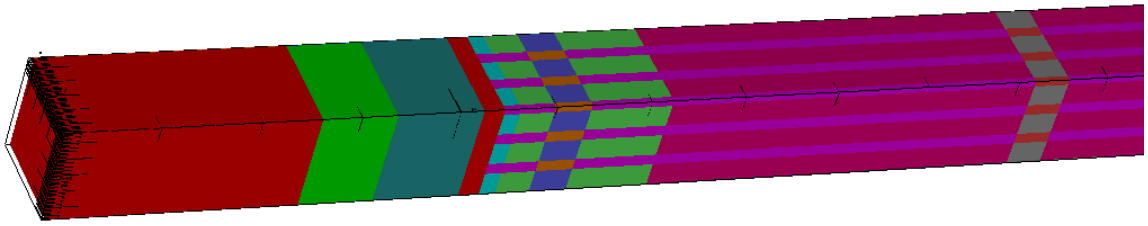

Figure 4.7: **Fuel assembly geometry end detail.** *Reactor core structure including spacer grids and plenum has been included. Lighter purple material on the right of the figure is moderator and darker purple/red material is fuel.*

contained in the model including spacer grids, fuel pins with homogenized cladding and gas gap, core plenum, and moderator with boron. Group cross sections and other discrete nuclear data are generated as needed by a cross-section processing module dependent on the meshing parameters used to discretize the geometry and single-dimension pin-cell calculations for initial flux spectrum generation. Table 4.8 gives the primary design parameters for the fuel assembly calculations.

To generate the multiplication factor and steady-state flux distribution for this problem, at every eigenvalue iteration MCSA is used to solve the resulting SP_N problem using the provided fission source. Algorithm 6.1 presents the use of MCSA within a power iteration strategy to find the multiplication factor. Here, $\mathbf{M}$ is the transport operator generated on the left-hand side of the SP_N discretization, $\mathbf{F}$ is the fission matrix, and $\mathbf{\Phi}$ the multigroup neutron flux. This problem is significantly more complicated than the simple test problem used for the previous spectral analysis. Fission has been introduced into the set of equations and the addition of moderator into the system will increase the amount of scattering, creating a significantly more difficult problem manifesting itself in an iteration matrix with a larger spectral radius. When using MCSA, the linear operator applied to $\mathbf{\Phi}^{n+1}$ at each eigenvalue iteration

Parameter	**Value**
Power Level	0 MW
Inlet Temperature	326.85C
Fuel Temperature	600C
Boron Concentration	1300 ppm
Moderator Density	0.743 g/cc
Helium Density	1.79×10^{-4} g/cc
Zirconium Density	6.56 g/cc
Stainless Steel Density	8.0 g/cc
Inconel Density	8.19 g/cc
UO2 Density	10.257 g/cc
Fuel Pin Radius (w/o clad)	0.4096 cm

Table 4.8: **Design parameters for the** 17×17 **pin fuel assembly criticality calculation.**

Algorithm 4.1 Power Iteration MCSA Scheme

$k_0 =$ initial guess
$\mathbf{\Phi}_0 =$ initial guess
$n = 0$
while $|\frac{k^n - k^{n-1}}{k^n}| < \epsilon$ **do** ▷ Iterate until convergence of the eigenvalue
  $\mathbf{M\Phi}^{n+1} = \frac{1}{k^n}\mathbf{F\Phi}^n$ ▷ Solve for the new flux state with MCSA
  $k^{n+1} = k^n \frac{\int \mathbf{F\Phi}^{n+1} d\mathbf{r}}{\int \mathbf{F\Phi}^n d\mathbf{r}}$ ▷ Update the multiplication factor
  $n = n + 1$
end while

will dictate convergence and remain unchanged throughout the computation[2] while the addition of fission to the system will only modify the source of neutrons and the multiplication factor while not affecting Monte Carlo transport.

Preliminary Jacobi Preconditioned Calculations

Based on the success of block Jacobi preconditioning with the test problem used for the spectral radius parameter study, we use it first to solve the 17×17 fuel assembly problem. A single energy group problem was first solved with SP_1 discretization, effectively giving the one-speed neutron diffusion system for the fuel assembly resulting in 20,088 degrees of freedom in the problem. Figure 4.8 gives the residual infinity norm as a function of iteration for the MCSA linear solve in the first eigenvalue iteration

[2]The operator will change if, for example, physics feedback through temperature or potentially burnup is considered. These additional physics will modify the cross sections used to assemble $\mathbf{M}$. The calculations presented here will consist of a single eigenvalue calculation with a static $\mathbf{M}$.

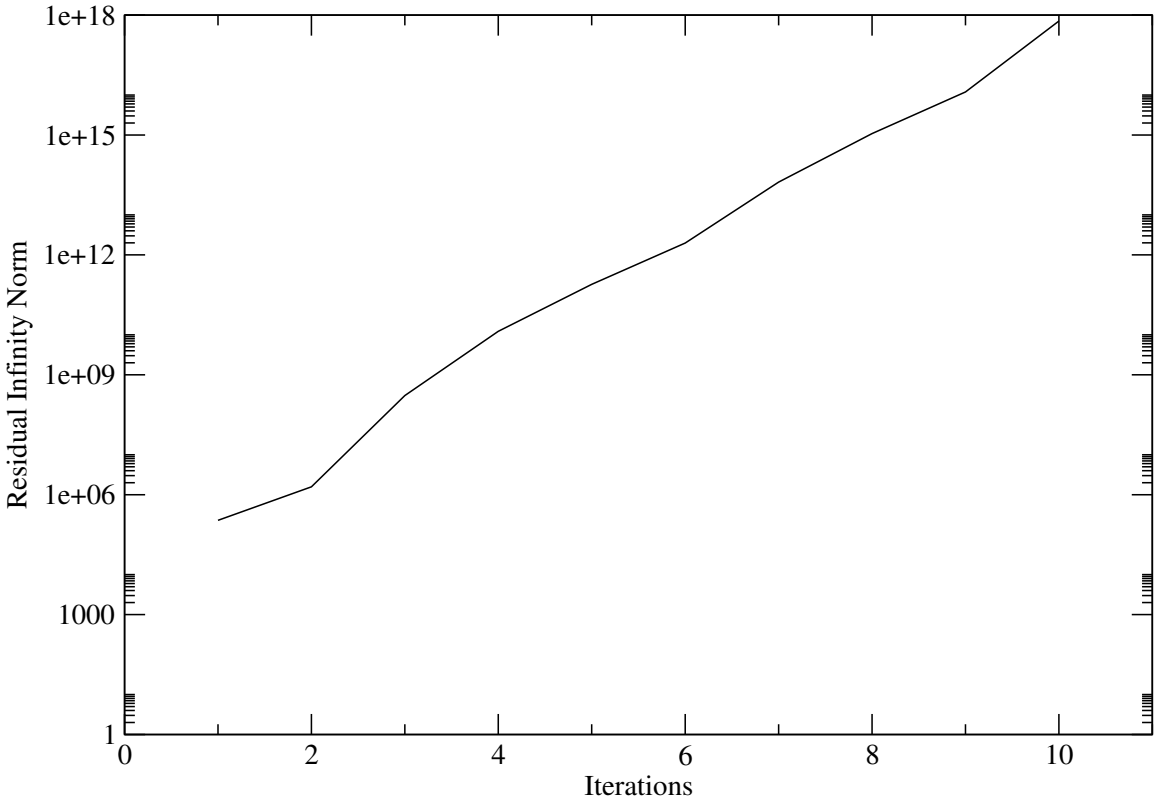

Figure 4.8: **Residual infinity norm vs. iteration for the block Jacobi preconditioned MCSA solve during the first eigenvalue iteration of the 1-group 17×17 fuel assembly problem.** *Convergence was not achieved with the block Jacobi preconditioned method.*

using 25,000 stochastic histories at every iteration for the adjoint Neumann-Ulam solve. Convergence was not achieved as noted by the rapid rise in the residual over a few iterations. Based on the spectral radius computations performed, these results are not in line with expectations for this problem. Additional computations performed with 1×10^6 histories per iteration exhibited the same divergent behavior at a significant computational cost and compute time. Even if the problem may be ill-conditioned, we do expect convergence of MCSA. To investigate further, a block Jacobi preconditioned Richardson iteration was used to solve the same problem. Figure 4.9 gives the residual infinity norm as a function of iteration for the Richardson linear solve in the first eigenvalue iteration. Poor converge is observed for the Richardson iteration, however, convergence is achieved meaning that the preliminary criteria needed to satisfy MCSA convergence has also been met. The number of iterations required for the Richardson iteration to converge will give an approximation for the spectral radius via Eq (2.22).

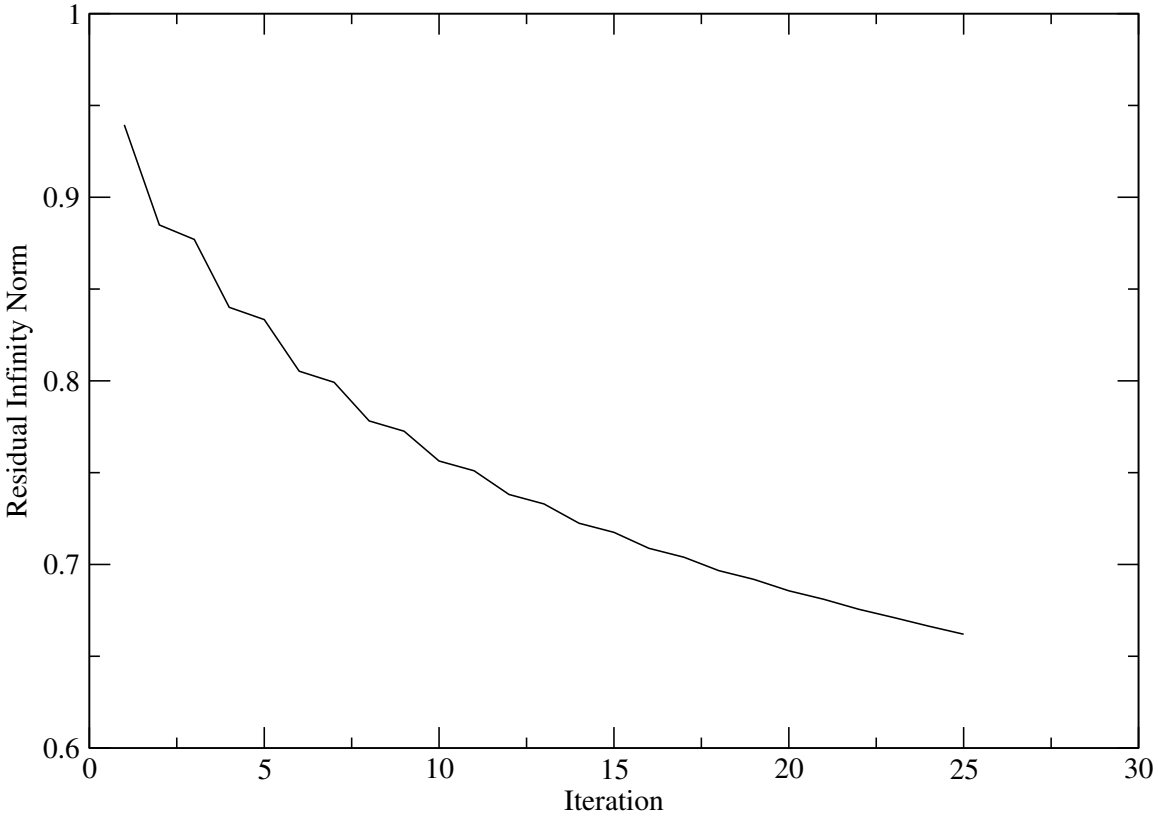

Figure 4.9: **Residual infinity norm vs. iteration for the block Jacobi preconditioned Richardson solve during the first eigenvalue iteration of the 1-group 17×17 fuel assembly problem.** *A spectral radius near 1 was observed for the iteration matrix.*

With 7291 iterations required to converge to a tolerance of 1×10^{-6}, $\rho(\mathbf{H}) \approx 0.998$, nearing the limits of MCSA applicability and well beyond the spectral radii generated in the initial spectral analysis.

Based on these results, it then appears that even if the simple criteria of a spectral radius of less than one is met and the Richardson iteration will converge with the same preconditioning, MCSA still may not converge. We then expect the issue to reside with the Neumann-Ulam solve providing the correction as the Richardson iteration is known to provide the correct result. Furthermore, the fact that the spectral radius is less than one means that the stochastic histories in the block Jacobi preconditioned Neumann-Ulam method are eventually being terminated by the weight cutoff as no artificial absorption was used for these preliminary calculations. Based on this, the correction being generated by the Neumann-Ulam solve is the component of MCSA causing a divergent solution.

MCSA Breakdown

For the fuel assembly problem, the initial block Jacobi preconditioned[3] calculations used a one-speed SP_1 discretization, effectively giving a diffusion system. To study the breakdown of MCSA at iteration matrix spectral radii near one, we will use a simpler homogeneous 2-dimensional one-speed neutron diffusion system to isolate this behavior. In this system, we can vary the cross sections while maintaining a fixed grid in order to achieve varying spectral radii. For these studies, we neglect fission as MCSA behavior is dictated by the transport operator $\mathbf{M}$ in an eigenvalue scheme with the fission matrix used to generate a fixed source.

Model Diffusion Problem

For our breakdown study, we choose the one-speed, two-dimensional neutron diffusion equation as a model problem (Duderstadt and Hamilton, 1976):

$$-\boldsymbol{\nabla}\cdot D\boldsymbol{\nabla}\phi+\Sigma_a\phi=S\,, \tag{4.18}$$

where ϕ is the neutron flux, Σ_a is the absorption cross section, and S is the source of neutrons. In addition, D is the diffusion coefficient defined as:

$$D=\frac{1}{3(\Sigma_t-\bar{\mu}\Sigma_s)}\,, \tag{4.19}$$

where Σ_s is the scattering cross section, $\Sigma_t=\Sigma_a+\Sigma_s$ is the total cross section, and $\bar{\mu}$ is the cosine of the average scattering angle. For simplicity, we will take $\bar{\mu}=0$ for our analysis giving $D=(3\Sigma_t)^{-1}$. In addition, to further simplify we will assume a homogeneous domain such that the cross sections remain constant throughout. Doing this permits us to rewrite Eq (4.18) as:

$$-D\boldsymbol{\nabla}^2\phi+\Sigma_a\phi=S\,. \tag{4.20}$$

We choose a finite difference scheme on a square Cartesian grid to discretize the problem. For the Laplacian, we choose the 9-point stencil shown in Figure 4.10 over a

[3]For 1-group SP_1 problems the block size is 1, giving a point Jacobi preconditioner.

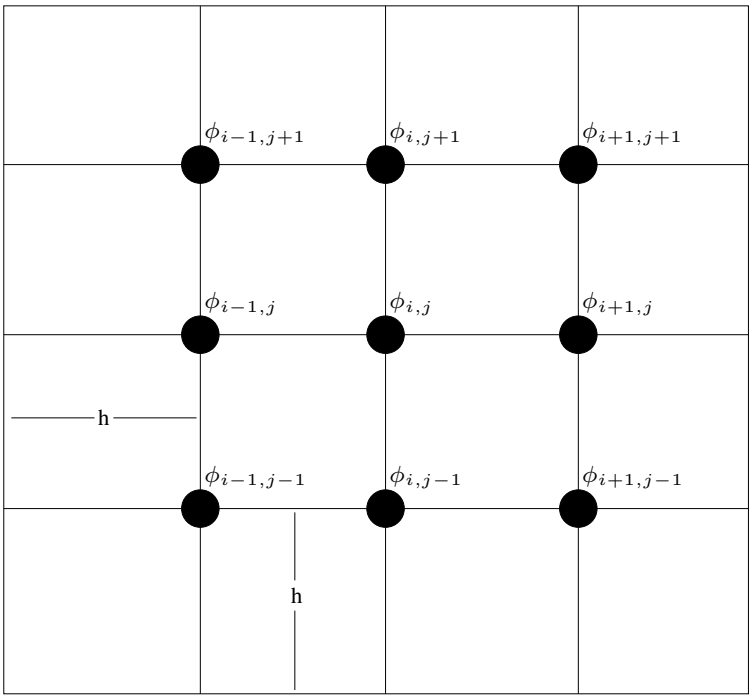

Figure 4.10: **Nine-point Laplacian stencil.**

grid of size h (LeVeque, 2007):

$$\nabla_9^2\phi = \frac{1}{6h^2}[4\phi_{i-1,j} + 4\phi_{i+1,j} + 4\phi_{i,j-1} + 4\phi_{i,j+1} + \phi_{i-1,j-1} + \phi_{i-1,j+1} + \phi_{i+1,j-1} + \phi_{i+1,j+1} - 20\phi_{i,j}] \,. \quad (4.21)$$

We then have the following linear system to solve:

$$-\frac{1}{6h^2}[4\phi_{i-1,j} + 4\phi_{i+1,j} + 4\phi_{i,j-1} + 4\phi_{i,j+1} + \phi_{i-1,j-1} + \phi_{i-1,j+1} + \phi_{i+1,j-1} + \phi_{i+1,j+1} - 20\phi_{i,j}] + \Sigma_a\phi_{i,j} = s_{i,j} \,, \quad (4.22)$$

and in operator form:

$$\mathbf{D}\boldsymbol{\phi} = \mathbf{s} \,, \quad (4.23)$$

where $\mathbf{D}$ is the diffusion operator, $\mathbf{s}$ is the source in vector form and $\boldsymbol{\phi}$ is the vector of unknown fluxes.

To close the system, a set of boundary conditions is required. In the case of a non-reentrant current condition applied to all global boundaries of the domain, we choose the formulation of Duderstadt by assuming the flux is zero at some ghost point

beyond the grid. Consider for example the equations on the $i = 0$ boundary of the domain:

$$-\frac{1}{6h^2}[4\phi_{-1,j} + 4\phi_{1,j} + 4\phi_{0,j-1} + 4\phi_{0,j+1} + \phi_{-1,j-1} + \phi_{-1,j+1} + \phi_{1,j-1} + \phi_{1,j+1} - 20\phi_{0,j}] + \Sigma_a\phi_{0,j} = s_{0,j}\,. \quad (4.24)$$

Here we note some terms where $i = -1$ and therefore are representative of grid points beyond the boundary of the domain. We set the flux at these points to be zero, giving a valid set of equations for the $i = 0$ boundary:

$$-\frac{1}{6h^2}[4\phi_{1,j} + 4\phi_{0,j-1} + 4\phi_{0,j+1} + \phi_{-1,j+1} + \phi_{1,j-1} + \phi_{1,j+1} - 20\phi_{0,j}] + \Sigma_a\phi_{0,j} = s_{0,j}\,. \quad (4.25)$$

We repeat this procedure for the other boundaries of the domain. For reflecting boundary conditions, the net current across a boundary is zero.

MCSA Breakdown Analysis

For each solver and estimator combination, the spectral radius of the iteration matrix generated by the diffusion problem was varied by changing the absorption cross section from 0.25 to 100 while fixing the grid size at 100×100 with $h = 0.1$ and a fixed scattering cross section of unity. For each parameter variation, a minimum of one stochastic history per degree of freedom (DOF) in the problem was used to compute the Monte Carlo correction. If the solver could not converge in less than 100 iterations, the number of histories was increased by increments of 5,000 until convergence was achieved in less than 100 iterations. The number of iterations required to converge MCSA and the time to converge was recorded as a means to capture the breakdown.

Figure 4.11 gives the number of iterations required to converge for the chosen number of histories per iteration given by Figure 4.12 using the adjoint solver with the collision and expected value estimators and the forward solver with the collision estimator. For spectral radii less than 0.97, all MCSA problems converged with 1 history per DOF (10,000 for this problem) with the number of iterations required to converge increasing as a function of spectral radius. Near a spectral radius of 0.97, the number of histories required to converge MCSA in less than 100 iterations takes a dramatic rise that exhibits neither exponential nor power law behavior. As the spectral radius approaches 1, the number of histories required becomes significant

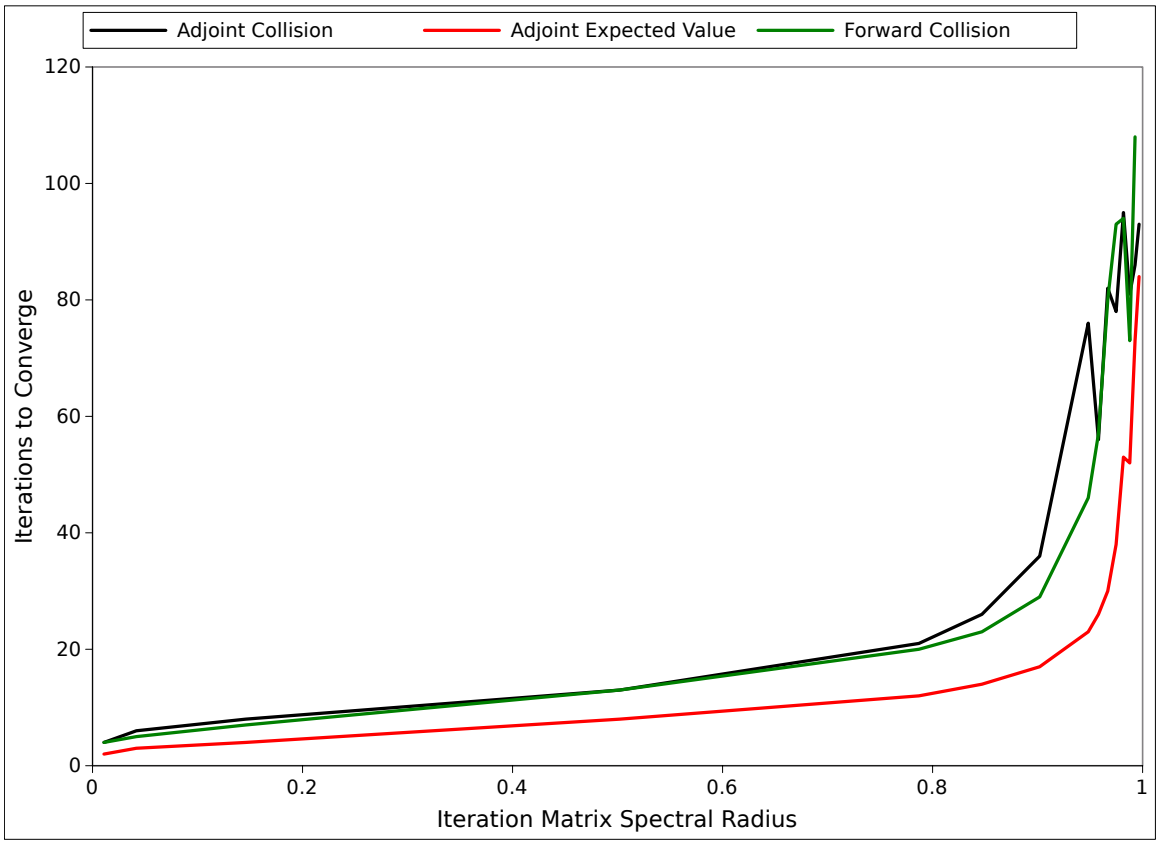

Figure 4.11: **Iterations required to converge as a function of spectral radius for the neutron diffusion problem.** *The number of histories was increased to achieve convergence in less than 100 iterations. At least 10,000 histories were used for each calculation.*

and effectively impractical to compute. Even with this simple diffusion problem, the behavior is consistent with that observed for the fuel assembly problem with SP_1 discretization. In that case, we estimated a spectral radius of ≈ 0.998, larger than any of the spectral radii that could be computed within even 90 minutes of compute time for this simple two dimensional problem. For that problem, even 1,000,000 histories (≈ 50 per DOF) was not enough to provide convergence. Even if single solve times of an hour can be tolerated, dozens of solves are typically required to compute the multiplication factor and flux spectrum within the SP_N eigenvalue scheme for more difficult problems like the fuel assembly, making the method unusable.

In addition to the significantly larger number of histories required for to achieve convergence for ill-conditioned problems another penalty is paid due to histories that take longer to compute. Figure 4.13 gives the CPU time in seconds required to compute a single random walk averaged over the entire set of histories run in the

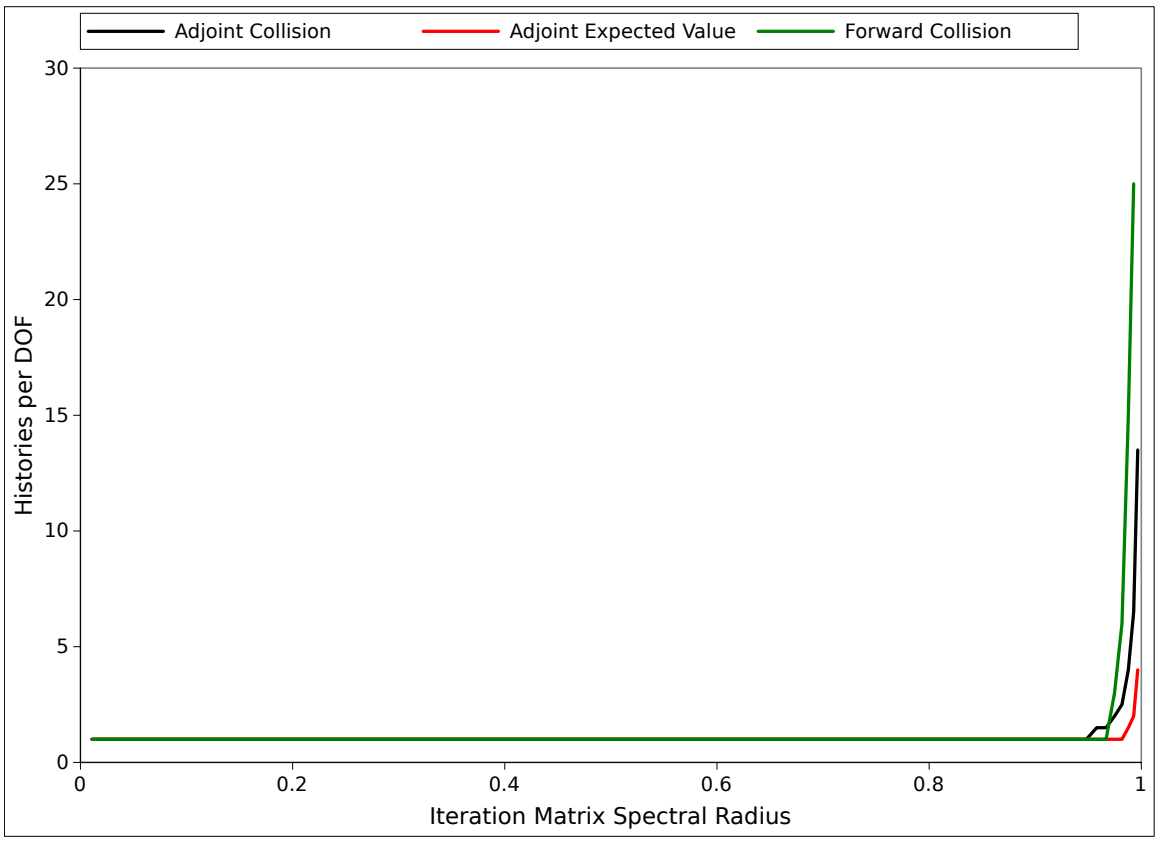

Figure 4.12: **Histories per DOF required to converge as a function of spectral radius for the neutron diffusion problem.** *The number of histories was increased to achieve convergence in less than 100 iterations. At least 10,000 histories were used for each calculation with 10,000 DOFs in the problem.*

calculation over all iterations. As the spectral radius increases (correlating to a higher ratio of scattering in the system) the random walk lengths increase, using more CPU time to finish the computation. Compared to spectral radii of 0.5, larger spectral radii over 0.97 have histories that require two orders of magnitude more computation time. This significant increase in computation time per history coupled with the significant increase in the number of histories required to converge is evidence that for problems with spectral radii above ≈ 0.97, using MCSA to solve any problems of interest is entirely ineffective and not practical. We therefore require a more expansive set of preconditioning techniques to move the eigenvalue spectrum of the SP_N problem into a regime in which MCSA is more applicable and in which performance is improved.

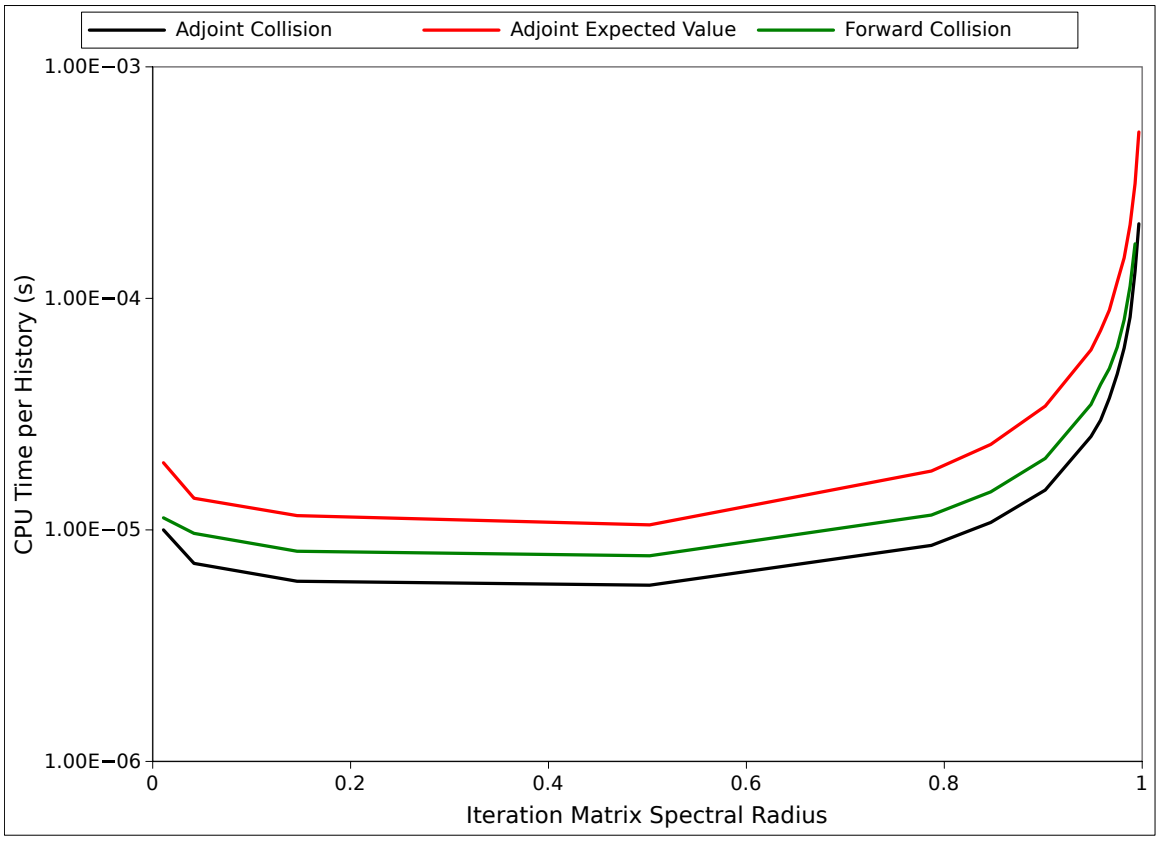

Figure 4.13: **CPU time per history as a function of spectral radius for the neutron diffusion problem.** *As the spectral radius grows, so do the length of the random walks. Longer random walks require more CPU time to compute.*

4.5 Advanced Preconditioning Strategies

For the fuel assembly criticality problem presented in the previous section, the spectral radius was near one using Jacobi preconditioning and in a regime in which MCSA breakdown is observed. In this regime the number of stochastic histories required to converge MCSA increases rapidly and the resulting poor performance is compounded by those histories being increasingly expensive to compute. To overcome this difficulty, advanced preconditioning strategies for the SP_N problem are required beyond simple Jacobi methods that can reduce the spectral radius into a region of better MCSA performance. Several modern algebraic preconditioning strategies will be presented here and used within the explicit preconditioning framework given by Eq (3.67). Data showing their effects on MCSA solutions of the fuel assembly criticality problem will be presented.

ILUT Preconditioning

Incomplete lower-upper (ILU) factorizations of the linear operator can be used a simple mechanism to form an approximate inverse of a preconditioner. To build the factorization, the sparse upper and lower triangular factors, $\mathbf{L}$ and $\mathbf{U}$, are computed such that the residual matrix formed by the factorization (Saad, 2003):

$$\mathbf{R} = \mathbf{LU} - \mathbf{A}\,, \tag{4.26}$$

has a specified sparsity pattern and element magnitude. When a magnitude threshold is used, elements generated in the factors below that magnitude are dropped, resulting in ILU threshold (ILUT) preconditioning. The sparsity pattern in this case is determined from the input matrix to be preconditioned and the number of elements maintained in the factor is specified by a fill level parameter. A fill level of 1 will generate the same number of elements as the sparsity pattern of the input matrix while a fill level of 2 will contain twice as many elements in the factorization, resulting in a better representation of the true LU factorization. The inverse of the lower and upper triangular factors may then be easily inverted by means of simple elimination to produce the preconditioner. For MCSA, $\mathbf{L}^{-1}$ will be used on the left and $\mathbf{U}^{-1}$ on the right to precondition the system.

For modern subspace methods, only the action to the preconditioner on a vector is required for efficient implementations and therefore a triangular solve can be used for this purpose. For the explicit preconditioning scheme presented in Eq (3.67), the

fully inverted operator must be generated in order to build the set of probabilities and weights required for Monte Carlo sampling. Therefore, for a linear system of size N, N triangular solves will be required in order to extract the inverse matrices from production ILU implementations. In addition, parallel implementations typically generate lower and upper factors that are only triangular locally, providing an easily parallelizable mechanism to generate the action of the preconditioner inverse. A consequence of this choice for parallel scalability is a degradation of the preconditioning quality as the size of the parallel system is increased. For serial computations, the triangular factorization is potentially exact depending on the parameters chosen while at thousands of parallel tasks, the global triangular factors differ significantly from the true factorization. As a result, more iterations are required at higher levels of parallelism to converge the system and will ultimately degrade overall scalability of the system with respect to total wall time to converge.

MCSA preconditioned with ILUT was used to solve the fuel assembly problem presented in the previous section for a 1-group SP_1 discretization. Unlike the Jacobi preconditioned strategy, convergence was achieved with ILUT preconditioning. To study convergence sensitivity to ILUT parameters, the fill level and drop tolerance were parametrically varied with the number of iterations to converge an eigenvalue iteration for the fuel assembly problem recorded along with the maximum number of non-zero entries observed in all matrix rows for the left/right preconditioned composite linear operator. To provide some sparsity to the factorization, the ILUT drop tolerance was used to drop elements in the extracted inverse triangular factors. For each calculation, the number of iterations required to converge reported was for a single eigenvalue iteration with 3×10^4 histories at each MCSA iteration to compute the Monte Carlo correction using the adjoint collision estimator. All ILUT calculations reported here were performed on a single CPU and therefore this data does not take into account the aforementioned effects of parallel decomposition on the quality of the ILUT preconditioning.

Figure 4.14 gives the number of iterations to converge the fixed source problem to a tolerance of 1×10^8. As expected, the higher the fill level chosen for the ILUT factorization, the fewer iterations are required to converge the problem (only 9 MCSA iterations were required to converge the problem for the fill level of 3 and drop tolerance of 1×10^{-5} case). For higher levels of fill, the iterations needed to converge were not as sensitive, signaling that a larger drop tolerance can perhaps be used without a significant degradation in iterative performance. At smaller fill levels, the sensitivity to the ILUT drop tolerance is more significant. For all fill levels used, convergence was

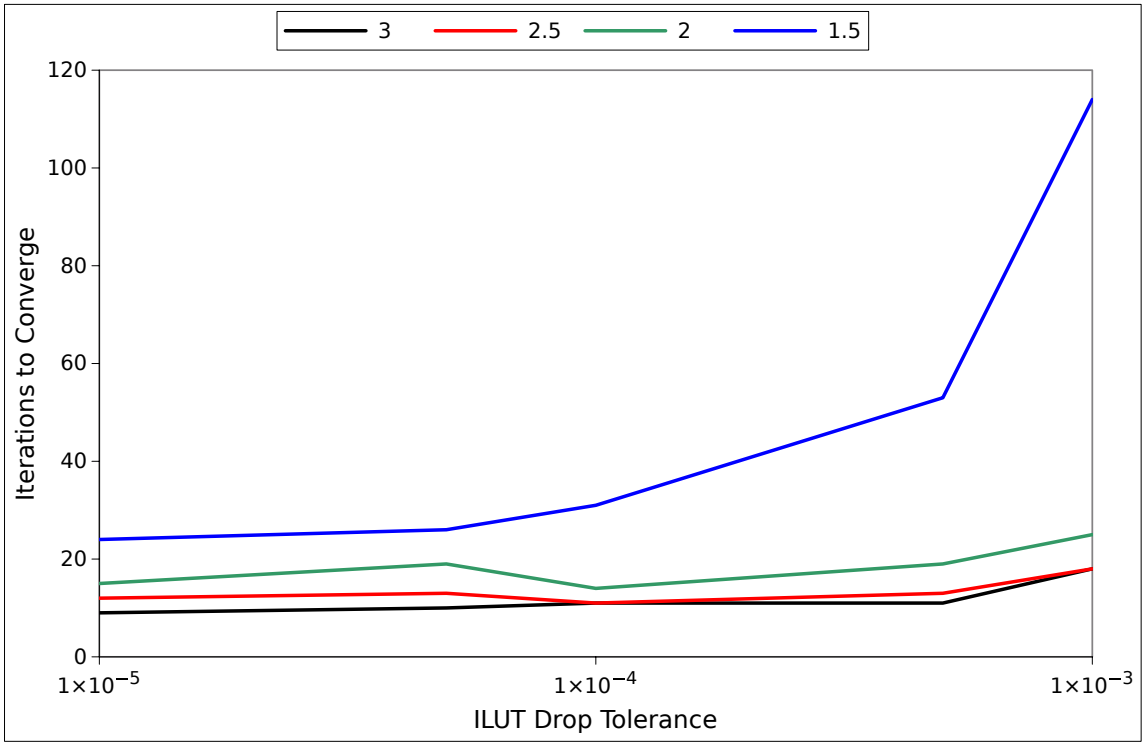

Figure 4.14: **Number of MCSA iterations required to converge an eigenvalue iteration for the fuel assembly problem with ILUT preconditioning as a function of ILUT drop tolerance.** *Each colored curve represents the iteration behavior for a different ILUT fill level. Fill levels of 1.5, 2.0, 2.5, and 3.0 were used.*

not achieved for the fuel assembly problem with a drop tolerance larger than 1×10^{-3}.

Unfortunately, gaining convergence (and excellent iterative performance) with MCSA for the SP_N fuel assembly problem comes at an immediate cost. Figure 4.15 gives the maximum number of non-zero entries in the composite linear operator generated by preconditioning as a function ILUT drop tolerance for varying values of fill level. For the 1-group SP_1 discretization, the original linear operator will contain only a maximum of 7 non-zero entries per row in the system. As observed in Figure 4.15, performing the explicit preconditioning yields composite linear operators with $O(1,000)$ elements in a row for all combinations of fill level and drop tolerance, over 10% of the total row size for this particular problem. This large number of row entries, observed for a significant fraction of rows in the system, creates several problems. First, sparsity is completely destroyed with each state in the system now coupled to over 10% of the total states in the system through the composite

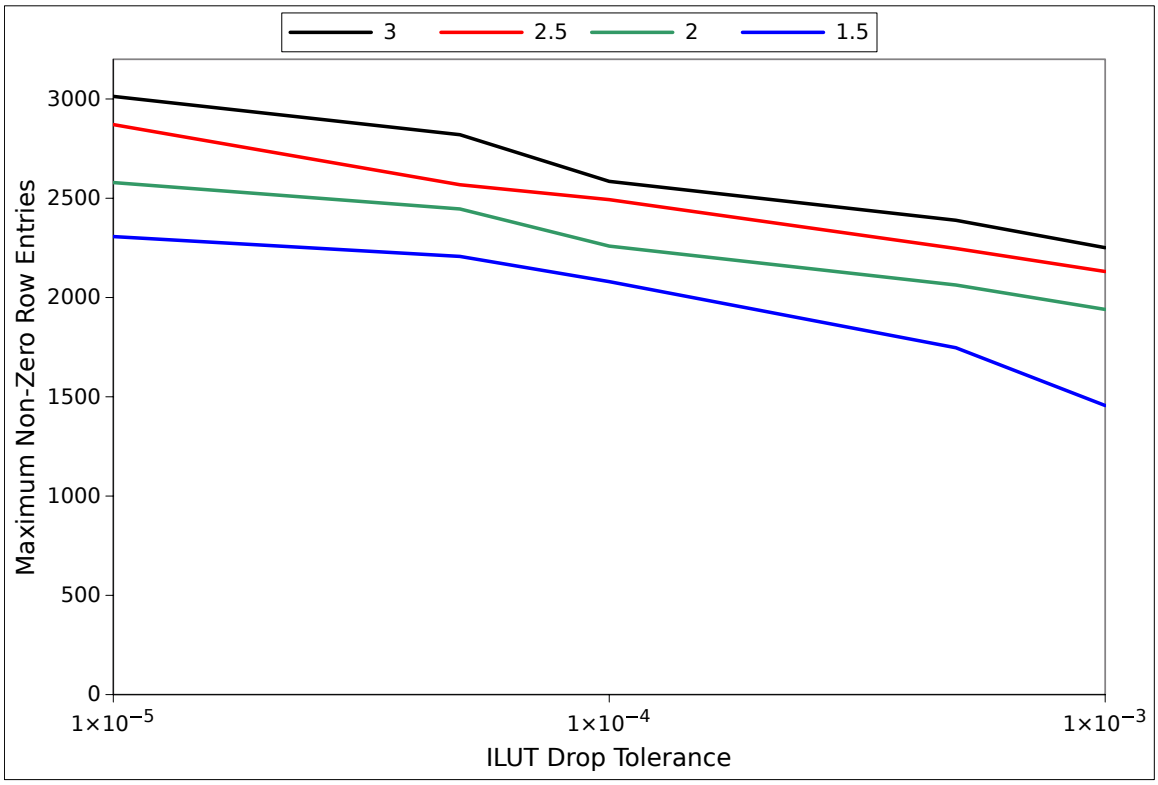

Figure 4.15: **Maximum number of non-zero entries observed for all rows in the composite linear operator for the fuel assembly problem with ILUT preconditioning given as a function of ILUT drop tolerance.** *Each colored curve represents the row size for a different ILUT fill level. Fill levels of 1.5, 2.0, 2.5, and 3.0 were used.*

iteration matrix. This means that Monte Carlo sampling tables will be large, requiring significant memory to store them and a substantial overhead in the sampling procedure during transport. In addition, the matrix-matrix multiply operations required to build the composite operator see significant performance losses due to the amount of data that must be handled. In parallel, losing sparsity severely inhibits performance where now parallel operations require communications amongst orders of magnitude more processors for nearest-neighbor type algorithms.

As a means to assess the quality of the preconditioning, a simple metric is developed to address these concerns and allow comparison to future developments. For our studies, our core performance metric will be iterative performance with the minimum number of MCSA iterations required to converge the problem desired. For Monte Carlo calculations, this improved performance is balanced by the creation of a dense composite system and we seek to reduce the number of non-zero entries to a minimum

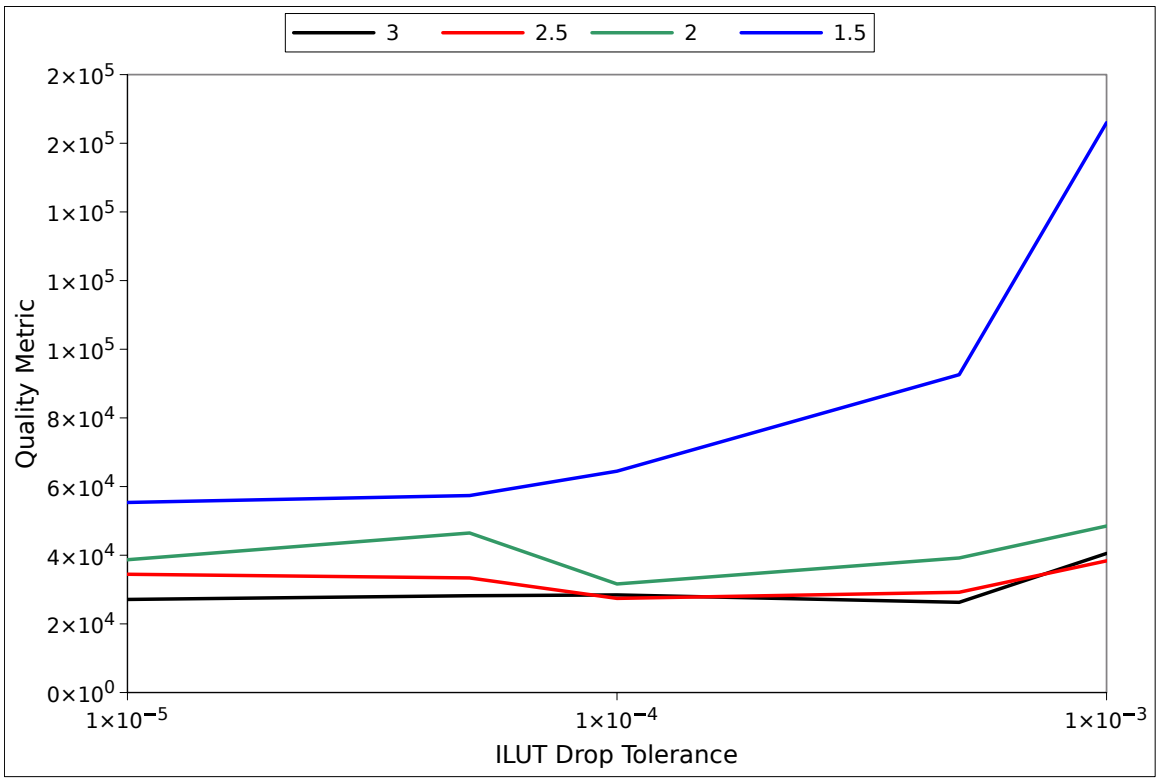

Figure 4.16: **ILUT preconditioning quality metric for the fuel assembly problem given as a function of ILUT drop tolerance.** *Each colored curve represents the quality metric behavior for a different ILUT fill level. Fill levels of 1.5, 2.0, 2.5, and 3.0 were used.*

value in order to lower the amount of coupling amongst states in the system and reduce the amount of memory used along with the potential latency overhead. Given these two objectives, the following metric is proposed:

$$\text{Quality Metric} = (\#\text{ iterations}) \times (\text{maximum }\#\text{ of non-zero values}) \,, \tag{4.27}$$

where the highest quality preconditioning is one that minimizes this metric. For the ILUT preconditioned data provided, Figure 4.16 provides the computed metric as a function of ILUT drop tolerance for each fill level used. In general, the metric is similar to the data observed for the number of iterations required to converge as the non-zero row entries changed by a smaller fraction over drop tolerances tested.

It should be noted here that although only the behavior of the linear solve during a single eigenvalue iteration is reported, the behavior of the linear solver was observed to be consistent throughout the eigenvalue iterations (25 total eigenvalue iterations

were required to converge the 1-group SP_1 problem). We expect this as the linear operator (which is unchanging in the fuel assembly criticality problem) dictates the convergence of MCSA. At each iteration the fission source provided to the linear problem is changing, however, we expect the same convergence behavior for all source vectors $\mathbf{b} \in \mathbb{R}^N$ where $||\mathbf{b}||_1 \neq 0$.

Sparse Approximate Inverse Preconditioning

Explicit formation of the preconditioner inverse is a requirement of the current MCSA preconditioning strategy. As a result, although ILUT preconditioning provided excellent iterative performance, a significant time penalty is paid to extract the inverse of the triangular factors through N triangular solves. In addition, this explicit inversion yielded a composite linear operator that was dense, requiring modification of the system in order to achieve better scalability and Monte Carlo performance. Sparse approximate inverse (SPAINV) preconditioning is a technique that may potentially alleviate both of these constraints at the cost of reduced iterative performance. The method produces a sparse approximation of the inverse matrix directly by minimizing the Frobenius norm of the residual matrix (Saad, 2003):

$$||\mathbf{I} - \mathbf{AM}||_F^2 = \sum_{j=1}^{N} ||\mathbf{e}_j - \mathbf{Am}_j||_2^2 \,, \tag{4.28}$$

where the inverse preconditioning matrix $\mathbf{M}$ minimizes the norm on the right and $\mathbf{e}_j$ and $\mathbf{m}_j$ are the j^{th} columns of the identity and preconditioning matrices respectively. Here, $\mathbf{M}$ is the actual inverse of the preconditioner and is formed directly by the algorithm. On the right hand side of Eq (4.28), the Frobenius norm is represented as a effective sum of N linear system residual norms. A few iterations of a subspace method or other iterative scheme can be used to effectively solve these systems to a loose tolerance and yield the columns of the approximate preconditioning matrix. At each iteration, threshold values are used to eliminate the components of each $\mathbf{m}_j$ column to maintain the desired level of sparsity. In addition, a sparsity pattern can be predetermined and enforced during this process. For the SPAINV implementation used for this work, the sparsity pattern was defined by levels such that a level of N for an input operator $\mathbf{A}$ will generate a preconditioning matrix $\mathbf{M}$ with the same sparsity pattern as $\mathbf{A}^{N+1}$. By using the norm minimization strategy instead of a factorization such as ILUT, parallel results are reproduced regardless of parallel problem size, meaning that a serial computation will converge in the same number of iterations as a computation with thousands of cores.

MCSA preconditioned with SPAINV was used to solve the fuel assembly problem as with ILUT for a 1-group SP_1 discretization. To study convergence sensitivity to SPAINV parameters, the number of levels in the sparsity pattern and threshold were parametrically varied with the number of iterations to converge a single eigenvalue iteration for the fuel assembly problem recorded along with the maximum number of

non-zero entries observed in all matrix rows for the left/right preconditioned composite linear operator. For each calculation, the number of iterations required to converge reported was for a single eigenvalue iteration with 3×10^4 histories at each MCSA iteration to compute the Monte Carlo correction using the adjoint collision estimator. All SPAINV calculations reported here were performed on a single CPU.

Figure 4.17 gives the number of MCSA iterations needed to converge the fuel assembly problem and Figure 4.18 the maximum number of non-zero entries per row obeserved in the composite linear operator as a function the SPAINV threshold. For this analysis, the number of levels in the sparsity pattern was varied from 3 to 7 with convergence of the fuel assembly problem not achieved for smaller level values. Compared to ILUT preconditioning, at higher levels we see comparable iterative performance with a relative invariance to the threshold value. The threshold did have a significant effect, however, on the time required to generate the preconditioner with a threshold of 0.001 over an order of magnitude slower than a threshold of 0.01 due to the inclusion of a significant number of extra values in the input operator. In addition to comparable iterative performance, the sparsity of the composite operator is greatly improved with nearly an order of magnitude reduction in number of non-zero entries per row for lower level values.

With the comparable iterative performance and improved sparsity, SPAINV preconditioning should then have a more favorable quality metric than ILUT preconditioning. Figure 4.19 gives the quality metric as a function of the threshold value for each of the sparsity levels computed. Not only do we see an improved quality metric over ILUT preconditioning (about an order of magnitude less), but we also note that the ideal sparsity level is not the largest nor the smallest. In fact, the largest and smallest levels (3 and 7) performed the worst in terms of the quality metric with a level of 4 performing the best. When CPU time to generate the preconditioner is considered, 4 is the clear choice for this problem as more time is required to do the approximate inversion as the sparsity pattern of the preconditioner grows in size.

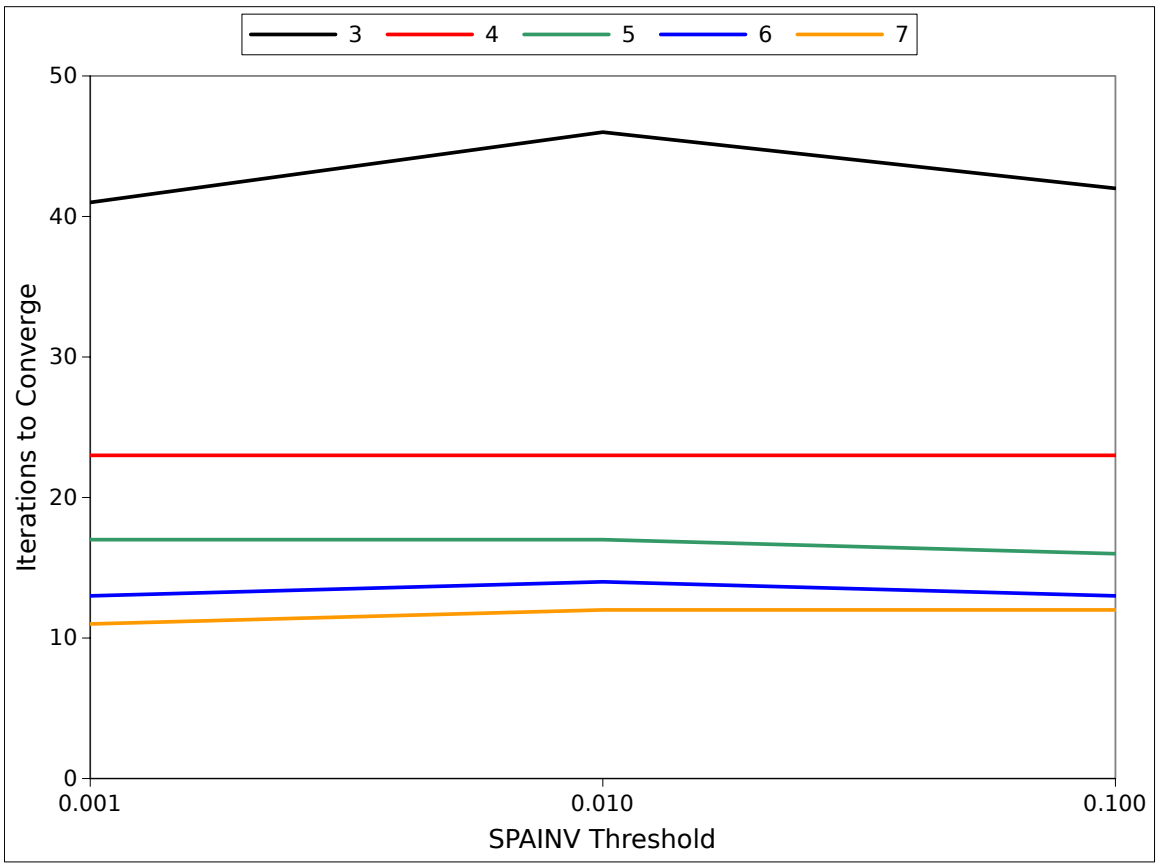

Figure 4.17: **Number of MCSA iterations required to converge a single eigenvalue iteration for the fuel assembly problem with SPAINV preconditioning as a function of SPAINV threshold.** *Each colored curve represents the iteration behavior for a different SPAINV level pattern. Levels of 3, 4, 5, 6, and 7 were used.*

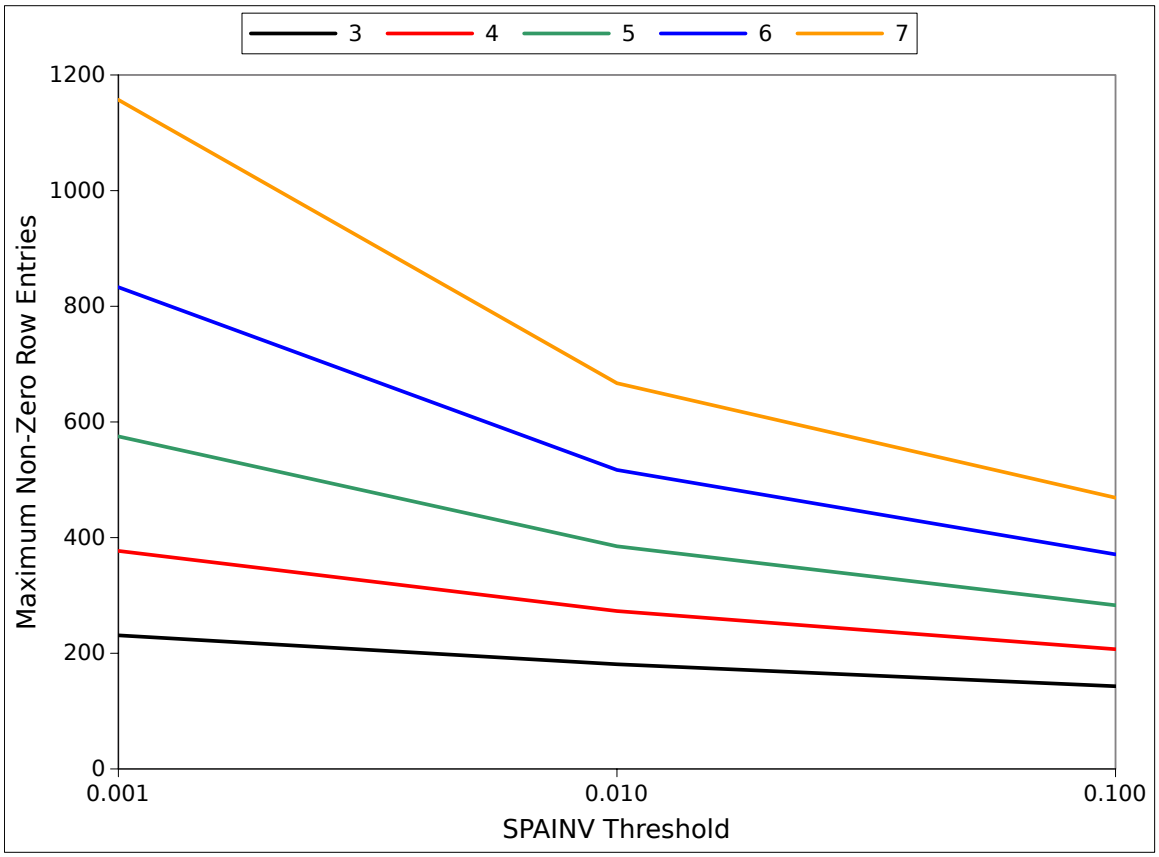

Figure 4.18: **Maximum number of non-zero entries observed for all rows in the composite linear operator for the fuel assembly problem with SPAINV preconditioning given as a function of SPAINV threshold.** *Each colored curve represents the row size for a different SPAINV level pattern. Levels of 3, 4, 5, 6, and 7 were used.*

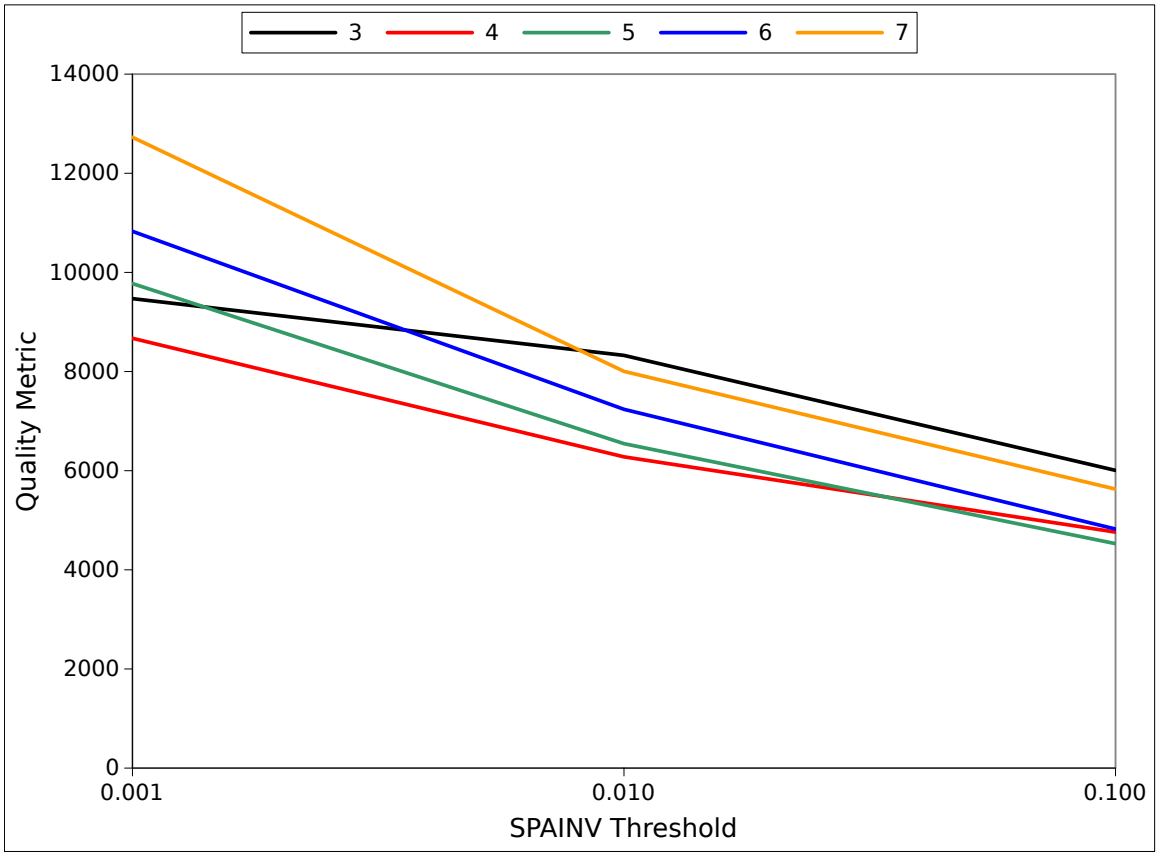

Figure 4.19: **SPAINV preconditioning quality metric for the fuel assembly problem given as a function of SPAINV threshold.** *Each colored curve represents the quality metric behavior for a different SPAINV level pattern. Levels of 3, 4, 5, 6, and 7 were used.*

SPAINV Improved ILUT Preconditioning

As a means to improve convergence over using only SPAINV preconditioning, ILUT preconditioning may be modified through an additional application of SPAINV (SPAINV-ILUT)(Saad, 2003). For the ILUT preconditioning case we have the following linear problem to solve for the substituted variable:

$$\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}\mathbf{u} = \mathbf{L}^{-1}\mathbf{b}\,. \tag{4.29}$$

We then apply SPAINV and minimize the following problem, this time on the left:

$$||\mathbf{I} - \mathbf{M}\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}||_F^2\,, \tag{4.30}$$

giving a new preconditioned system:

$$\mathbf{M}\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}\mathbf{u} = \mathbf{M}\mathbf{L}^{-1}\mathbf{b}\,. \tag{4.31}$$

Doing this permits a smaller fill level to be used with ILUT and a lower-order sparsity pattern to be used with SPAINV, giving marginally more sparsity in the resulting composite linear operator while maintaining good iterative performance.

MCSA preconditioned with SPAINV-ILUT was also used to solve the same fuel assembly problem as with the other preconditioners for the same 1-group SP_1 discretization. To study convergence sensitivity to SPAINV-ILUT parameters, the ILUT drop tolerance and ILUT fill level were parametrically varied with the number of iterations to converge a single eigenvalue iteration for the fuel assembly problem recorded along with the maximum number of non-zero entries observed in all matrix rows for the left/right preconditioned composite linear operator. SPAINV parameters for the data presented here were fixed as it was found that the preconditioning quality was largely insenstive to their variation. In addition, the fact that SPAINV is being used to preconditioned a problem already preconditioned with ILUT means that the input matrix sparsity pattern is already somewhat dense and large SPAINV levels would result in an even denser system. Therefore, a smaller SPAINV sparsity pattern level will be used for this preconditioning. For each calculation, the number of iterations required to converge reported was for a single eigenvalue iteration. Again, 3×10^4 histories are used at each MCSA iteration to compute the Monte Carlo correction using the adjoint collision estimator. All SPAINV-ILUT calculations reported here were performed on a single CPU.

Figure 4.20 gives the number of iterations required to converge as a function of ILUT drop tolerance for ILUT fill levels of 1.5, 2.0, 3.0, 4.0 and 5.0 with a fixed SPAINV threshold of 1.0 and sparsity level of 1. SPAINV threshold values higher than 1.0 resulted in a loss of convergence and lower values did not significantly affect performance. Larger sparsity levels resulted in a composite operator that was too dense and a prohibitive amount of CPU time required to compute the preconditioner. At smaller levels of fill, the benefits of using an additional step of SPAINV preconditioning are readily observed with the number of iterations required to converge cut in half for large drop tolerances. For smaller drop tolerances and higher ILUT levels of fill, the additional step of SPAINV preconditioning offers marginal improvement to iterative performance. As shown in Figure 4.21, because ILUT is used in the preconditioning sequence, sparsity is again lost with only marginally improved non-zero entry numbers due to the application of the SPAINV threshold. As a result, the quality metrics given by Figure 4.22 are only marginally better than those observed for ILUT preconditioning and much larger than the SPAINV quality metric values.

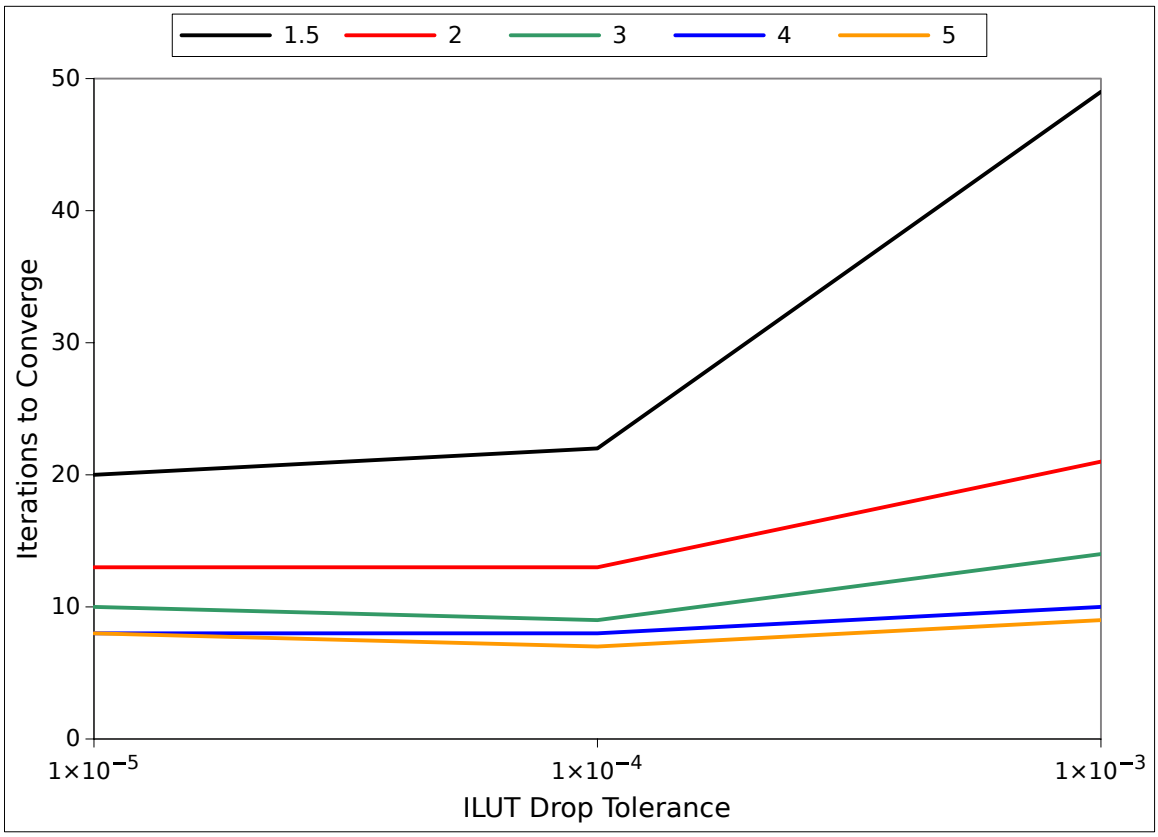

Figure 4.20: **Number of MCSA iterations required to converge a single eigenvalue iteration for the fuel assembly problem with SPAINV-ILUT preconditioning as a function of ILUT drop tolerance.** *Each colored curve represents the row size for a different ILUT fill level. Fill levels of 1.5, 2.0, 3.0, 4.0, and 5.0 were used.*

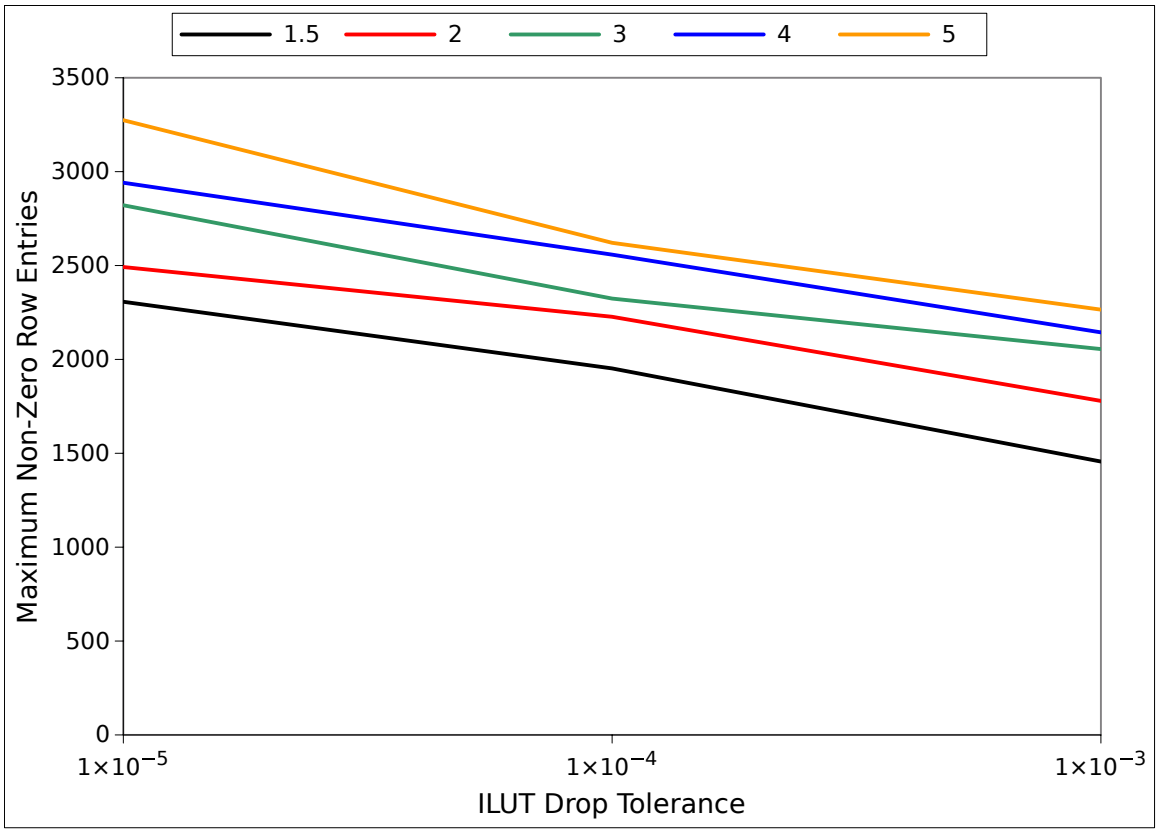

Figure 4.21: **Maximum number of non-zero entries observed for all rows in the composite linear operator for the fuel assembly problem with SPAINV-ILUT preconditioning given as a function of ILUT drop tolerance.** *Each colored curve represents the row size for a different ILUT fill level. Fill levels of 1.5, 2.0, 3.0, 4.0 and 5.0 were used.*

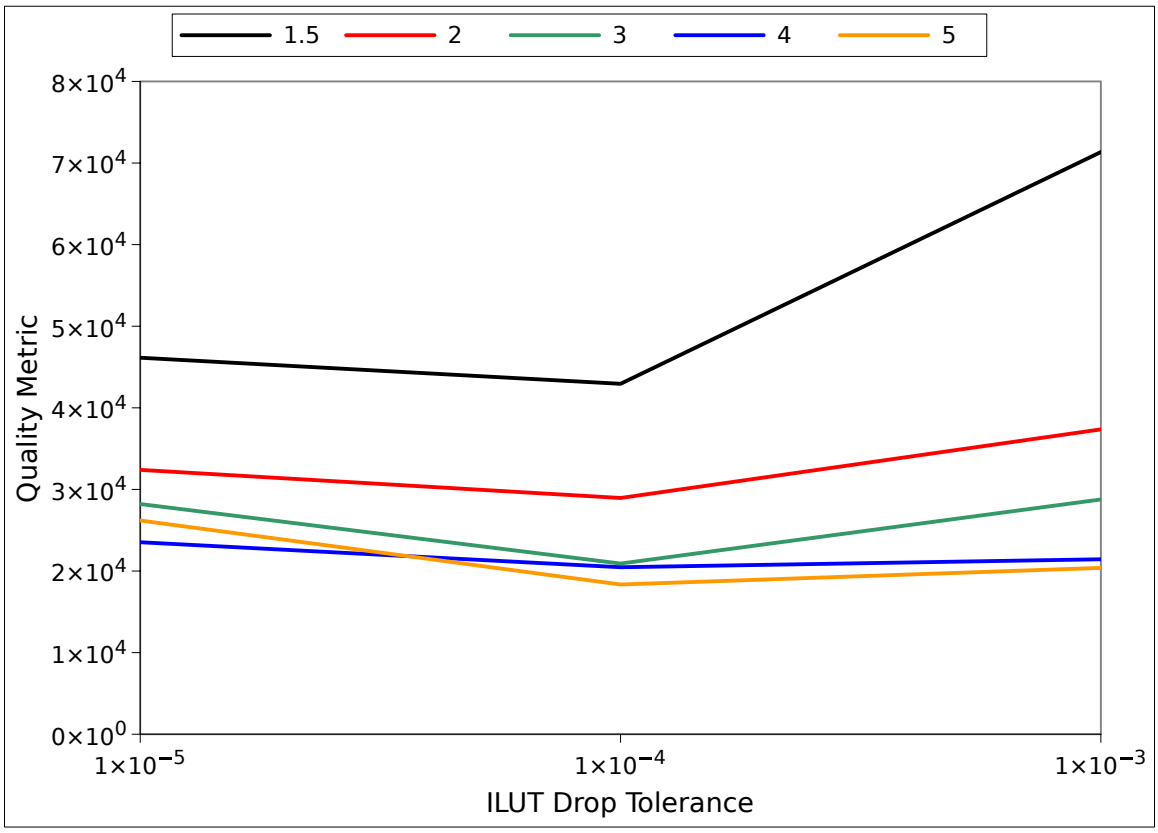

Figure 4.22: **SPAINV-ILUT preconditioning quality metric for the fuel assembly problem given as a function of ILUT drop tolerance.** *Each colored curve represents the row size for a different ILUT fill level. Fill levels of 1.5, 2.0, 3.0, 4.0 and 5.0 were used.*

Applying the Reduced Domain Approximation

For each preconditioning technique presented convergence was achieved for the single fuel assembly problem. However, a primary concern is the number of non-zero states in each row of the system generated by the explicit preconditioning strategy. In many cases, orders of magnitude more matrix elements were generated resulting in poor scalability for domain decomposed algorithms and overall performance issues for Monte Carlo. As outlined in § 3.9, the reduced domain approximation may be used as a mechanism to potentially alleviate this problem by filtering elements of the composite matrix in each row that fall below a certain threshold value or by maintaining the largest N elements in each row where N is a designated fill level.

For the ILUT, SPAINV, and SPAINV-ILUT preconditioning strategies the reduced domain approximation will be applied to reduce the density of the composite linear operator to more manageable levels. For each preconditioner, the parameters that achieved the best quality metric results from the previous analysis were used. These correspond to ILUT parameters of a fill level of 5.0 and a drop tolerance of 1×10^{-5}, SPAINV parameters of a sparsity level of 4 and a threshold of 0.1 and SPAINV-ILUT with ILUT parameters of a fill level of 4.0 and a drop tolerance of 1×10^{-5} and SPAINV parameters of a sparsity level of 1 and a threshold of 1.0. The reduced domain threshold was set to 1×10^{-10} in order to eliminate any exceedingly small values from the matrix (generally this is simply removing non-zero elements within the floating point tolerance of zero). The reduced domain fill level was then varied, starting with the largest non-zero entries per row value observed for each of the preconditioning types in order to assess its effects relative to the case where no reduced domain approximation was applied.

Figure 4.23 gives the number of iterations required to converge for each preconditioner type as a function of reduced domain fill level. Figure 4.24 gives the corresponding quality metric for each data point where the number of non-zero entries used to compute the metric is equivalent to the reduced domain fill level. We first note the SPAINV preconditioning alone is signficantly more sensitive to the reduction in domain size over the ILUT-based methods, although the preconditioner was of $O(100)$ non-zero entries per row without any approximation applied. For the ILUT-based methods, performance was significantly better with convergence achieved in less than 40 MCSA iterations with only 10 non-zero entries in each row (vs. 7 non-zero entries for the case with no preconditioning). SPAINV-ILUT iterative performance was marginally better than ILUT alone for all reduced domain fill levels. However, it

should be noted that the ILUT level of fill used for the ILUT only calculations was set to 4 for this case while the SPAINV-ILUT preconditioner used an ILUT level of fill of 5 and therefore the marginally better performance is more likely a result of this addition of fill rather than the extra SPAINV preconditioning. Looking at the quality metric data, we see a nice power-law reduction in the quality metric as a function of the reduced domain fill level, achieving 2 orders of magnitude reduction in the quality metric for the ILUT-based preconditioning methods.

Although applying the reduced domain approximation results in a successful recovery of sparsity for the Monte Carlo problem while maintaining good convergence properties, there is still an issue of forming the composite operator before applying the approximation and potentially generating the transpose of this operator in the case of the adjoint Monte Carlo method. Because of this, memory and scaling issues may still be observed when building the probability and weight matrices for the Monte Carlo problem. In addition, the expensive extraction of the inverse of the preconditioning operators for the explicit scheme creates a significant cost in overall performance. Future work in this area should be considerate of these important components of the preconditioned MCSA algorithm.

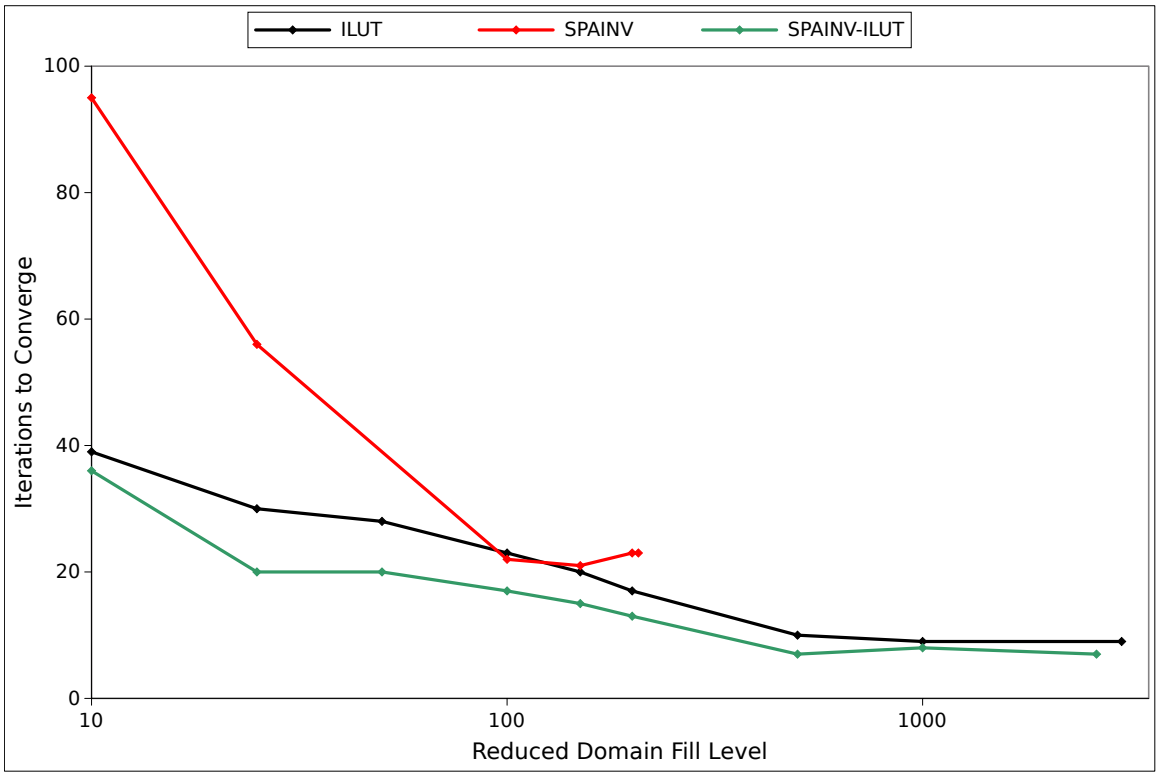

Figure 4.23: **Number of MCSA iterations required to converge a single eigenvalue iteration for the fuel assembly problem with each preconditioning as a function of reduced domain approximation fill level.** *The largest fill level for each preconditioning presented is that using the parameters that gave the best results without the approximation.*

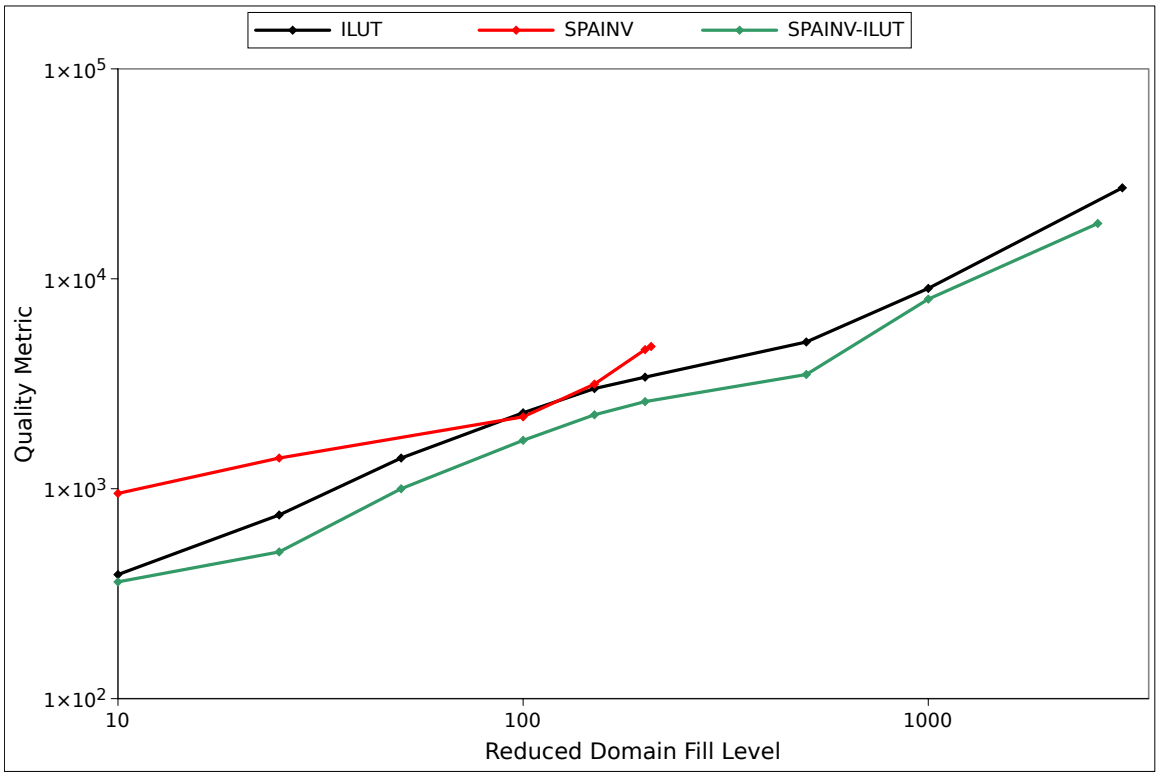

Figure 4.24: **Preconditioning quality metric for the fuel assembly problem given as a function of reduced domain approximation fill level.** *The largest fill level for each preconditioning presented is that using the parameters that gave the best results without the approximation.*

MCSA Relaxation Parameters

The Richardson iteration upon which the Neumann-Ulam method is built may be implemented with a scalar relaxation parameter that can be adjusted to improve convergence:

$$\mathbf{x} = \mathbf{x} + \omega \mathbf{r} \,, \tag{4.32}$$

where ω is the relaxation parameter. This is effectively a form of preconditioning where the system is being scaled on the left by a constant value in all rows. Analogously, these relaxation parameter techniques can be applied to Monte Carlo to improve convergence as demonstrated by Dimov (Dimov et al., 1998). In this case, building an iteration matrix from Eq (4.32) gives:

$$\mathbf{H} = \mathbf{I} - \omega \mathbf{A} \,, \tag{4.33}$$

with the probabilities and weights for the Monte Carlo game appropriately scaled. By inspection, such a scaling is a simple form of preconditioning on the left where all rows in the system are scaled by the same scalar parameter. For MCSA, we can stage this scheme with two separate relaxation parameters; one for the outer Richardson iteration and one for the inner Monte Carlo solve:

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + \omega_R \mathbf{r}^{k} \,, \tag{4.34a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \,, \tag{4.34b}$$

$$\omega_N \mathbf{A} \delta\mathbf{x}^{k+1/2} = \omega_N \mathbf{r}^{k+1/2} \,, \tag{4.34c}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2} \,, \tag{4.34d}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \,, \tag{4.34e}$$

where ω_R is the Richardson iteration relaxation parameter and ω_N is the Neumann-Ulam Monte Carlo solve relaxation parameter.

We apply these relaxation parameters to the fuel assembly problem along with the reduced domain approximation as a means of studying their effects. For each calculation presented, the number of iterations required to converge reported was for a single eigenvalue iteration. Again, 3×10^4 histories are used at each MCSA iteration to compute the Monte Carlo correction using the adjoint collision estimator. A reduced domain fill level of 100 is used with a threshold of 1×10^{-10} to filter small values.

Figure 4.25: **Number of iterations to converge a single eigenvalue iteration of the fuel assembly problem as a function of the Richardson relaxation parameter.** *The Neumann relaxation parameter was fixed at 1.0 and* 3×10^4 *stochastic histories were used to compute the MCSA correction at each iteration.*

Figure 4.26: **CPU time in seconds to converge a single eigenvalue iteration of the fuel assembly problem as a function of the Richardson relaxation parameter.** *The Neumann relaxation parameter was fixed at 1.0 and* 3×10^4 *stochastic histories were used to compute the MCSA correction at each iteration.*

Figure 4.27: **Number of iterations to converge a single eigenvalue iteration of the fuel assembly problem as a function of the Neumann relaxation parameter.** *The Richardson relaxation parameter was fixed at 1.0 and* 3×10^4 *stochastic histories were used to compute the MCSA correction at each iteration.*

Figure 4.28: **CPU time in seconds to converge a single eigenvalue iteration of the fuel assembly problem as a function of the Neumann relaxation parameter.** *The Richardson relaxation parameter was fixed at 1.0 and* 3×10^4 *stochastic histories were used to compute the MCSA correction at each iteration.*

Monte Carlo Estimator Comparison

With convergence obtained for the fuel assembly problem, the advanced preconditioners will next be studied with each Monte Carlo estimator. Although the expected value estimator has achieved the best iterative performance in the previous comparison and analysis, it is possible that the other estimators may perform better from a timing perspective due to the sparsity lost by explicit preconditioning. The deterministic component of the expected value estimator that couples the current state to other local states through the iteration matrix stencil requires application of the estimator to potentially several orders of magnitude more states at every transition event.

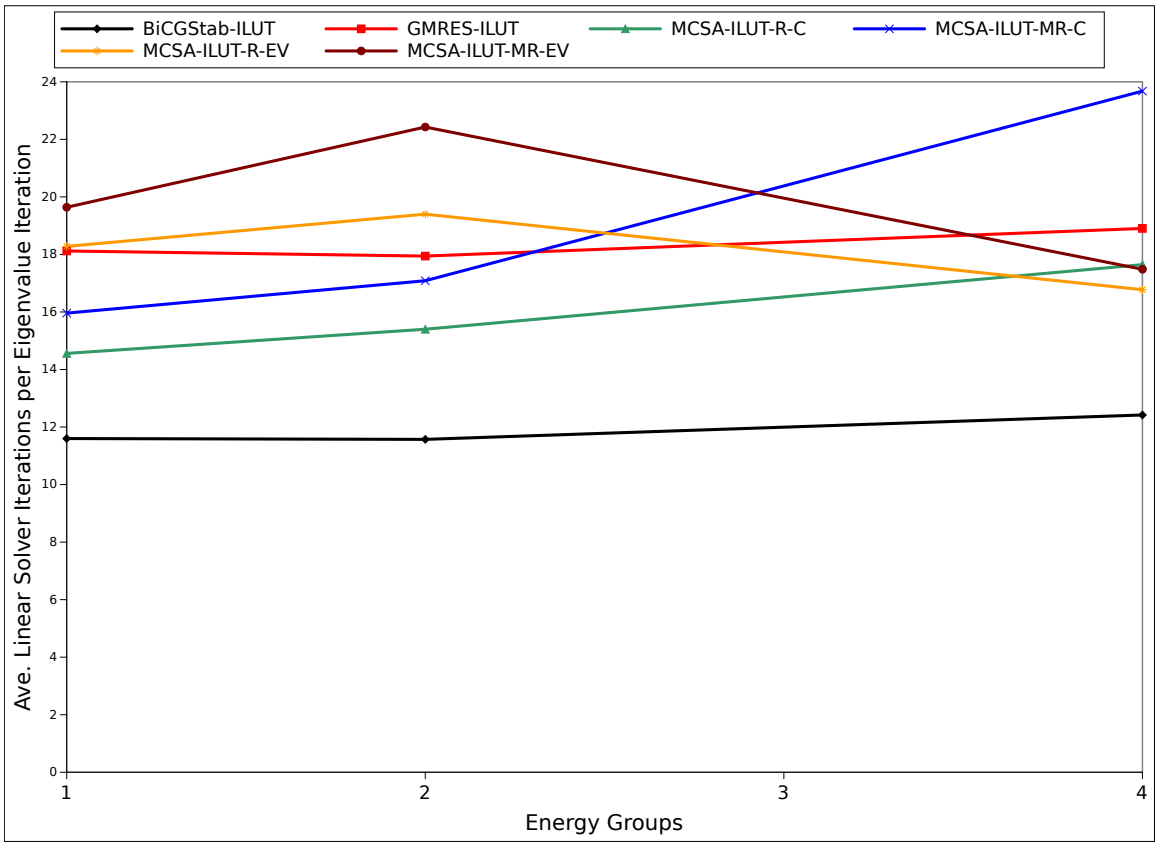

Figure 4.29: **Average number of iterations required to converge each eigenvalue iteration for the fuel assembly problem as a function of energy groups.** *All methods were preconditioned with identical ILUT parameters. For MCSA, several variations of fixed point iterations and estimators were used.*

4.6 MCSA Comparison to Conventional Methods

Results

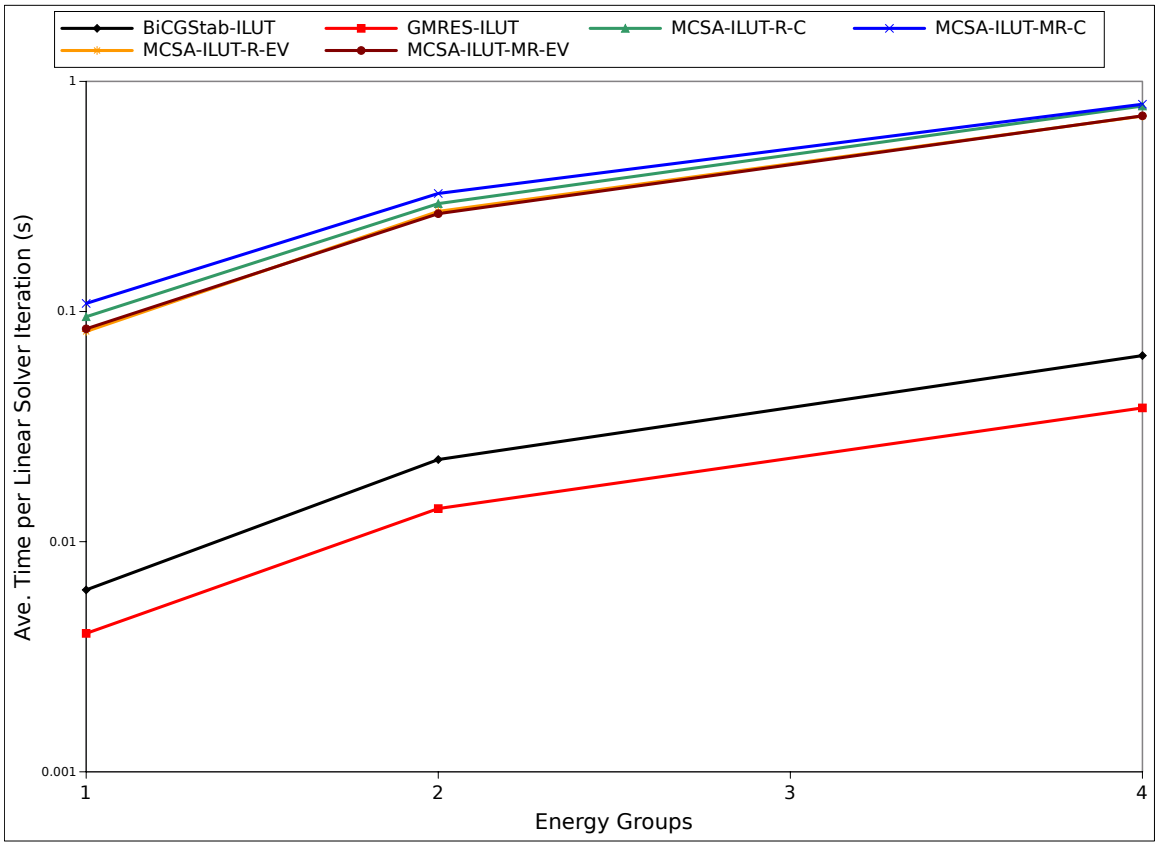

Figure 4.30: **Average CPU time per iteration in seconds for the fuel assembly problem as a function of energy groups.** *All linear solver iterations over all eigenvalue iterations were used to compute the averge. All methods were preconditioned with identical ILUT parameters. For MCSA, several variations of fixed point iterations and estimators were used.*

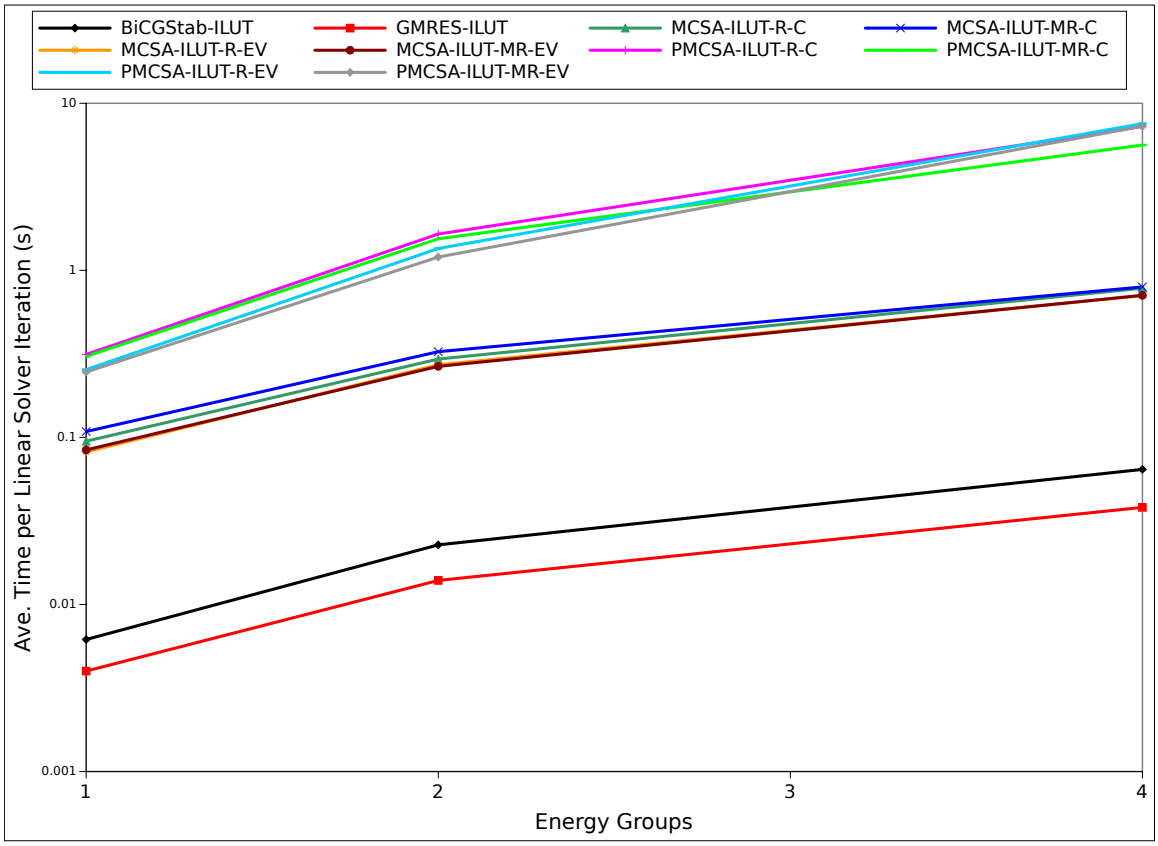

Figure 4.31: **Average CPU time per iteration in seconds for the fuel assembly problem as a function of energy groups with preconditioning time included for the MCSA methods.** *All linear solver iterations over all eigenvalue iterations were used to compute the averge. All methods were preconditioned with identical ILUT parameters. For MCSA, several variations of fixed point iterations and estimators were used.*

Chapter 5

Monte Carlo Solution Methods Applied to Nonlinear Systems

Nonlinear equation sets are a common occurrence in multiphysics problems. Systems of partial differential equations such as those that describe fluid flow or more general transport processes when discretized by conventional methods yield discrete sets of stiff equations with nonlinearities present in the variables. Traditionally, such systems have been solved by linearizing them in a form where the nonlinearities in the variables are eliminated and more traditional linear methods can be used for solutions. Often characterized as segregated methods where physics operators are split and their action on the system approximated in steps, such methods lack consistency and accuracy in resolving the nonlinear component of the solution. In the last 30 years, fully consistent nonlinear methods based on Newton's method have become more popular and many advances have been made in the computational physics field to employ these methods.

In the context of solving standalone linear systems, Monte Carlo methods do not provide significant merit over Krylov methods due to the fact that the linear operator must be explicitly formed. For many applications, such a requirement is prohibitive and perhaps not even feasible to implement. Therefore, a Monte Carlo solver is best suited for situations in which not only are Krylov methods applicable, but also in which the operator is readily, if not naturally, formed. Modern nonlinear methods meet both of these requirements with Newton methods used in conjunction with Krylov methods for a robust, fully implicit solution strategy. Furthermore, modern techniques exist that permit the automatic construction of the linear operator generated within a Newton method based on the nonlinear residual evaluations, providing all of the components necessary for a Monte Carlo solver to provide value. We therefore devise a new nonlinear method based on the MCSA algorithm and Newton's method and discuss its potential benefits.

5.1 Preliminaries

We formulate the *nonlinear problem* as follows (Knoll and Keyes, 2004):

$$\mathbf{F}(\mathbf{u}) = \mathbf{0}\,, \tag{5.1}$$

where $\mathbf{u} \in \mathbb{R}^n$ is the solution vector and $\mathbf{F} : \mathbb{R}^N \to \mathbb{R}^N$ is the function of nonlinear residuals. We write the nonlinear system in this form so that when an exact solution for $\mathbf{u}$ is achieved, all residuals evaluate to zero. *Newton's method* is a root finding algorithm and therefore we can use it to solve Eq (5.1) if we interpret the exact solution $\mathbf{u}$ to be the roots of $\mathbf{F}(\mathbf{u})$. Newton's method is also an iterative scheme, and we can generate this procedure by building the Taylor expansion of the $k+1$ iterate of the nonlinear residuals about the previous k iterate:

$$\mathbf{F}(\mathbf{u}^{k+1}) = \mathbf{F}(\mathbf{u}^k) + \mathbf{F}'(\mathbf{u}^k)(\mathbf{u}^{k+1} - \mathbf{u}^k) + \frac{\mathbf{F}''(\mathbf{u}^k)}{2}(\mathbf{u}^{k+1} - \mathbf{u}^k)^2 + \cdots\,. \tag{5.2}$$

If we ignore the nonlinear terms in the expansion and assert that at the $k+1$ iterate $\mathbf{u}^{k+1}$ is the exact solution such that $\mathbf{F}(\mathbf{u}^{k+1}) = \mathbf{0}$, then we are left with the following equality:

$$-\mathbf{F}(\mathbf{u}^k) = \mathbf{F}'(\mathbf{u}^k)(\mathbf{u}^{k+1} - \mathbf{u}^k)\,. \tag{5.3}$$

We note two things of importance in Eq (5.3). The first is that $\mathbf{F}'(\mathbf{u}^k)$ is in fact the *Jacobian*, $\mathbf{J}(\mathbf{u})$, of the set of nonlinear residuals and is defined element-wise as:

$$J_{ij} = \frac{\partial F_i(\mathbf{u})}{\partial u_j}\,. \tag{5.4}$$

Second, we note that $(\mathbf{u}^{k+1} - \mathbf{u}^k)$ is simply the solution update from the k iterate to the $k+1$ iterate. We will define this update as the *Newton correction* at the k iterate, $\delta\mathbf{u}^k$. To finish, we can then rearrange Eq (5.3) to define the Newton iteration scheme for nonlinear problems:

$$\mathbf{J}(\mathbf{u})\delta\mathbf{u}^k = -\mathbf{F}(\mathbf{u}^k) \tag{5.5a}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \delta\mathbf{u}^k\,. \tag{5.5b}$$

There are then three distinct steps to perform: evaluation of the nonlinear residuals using the solution at the k iterate, the solution of a linear system to compute the Newton correction where the Jacobian matrix of the nonlinear equation set is the

linear operator, and the application of the correction to the previous iterate's solution to arrive at the next iterate's solution. In the asymptotic limit, the iterations of Newton's method will converge the nonlinear residual quadratically (Kelley, 1995). Convergence criteria is set for stopping the iteration sequence based on the nonlinear residual. Commonly, the following criteria is used:

$$||\mathbf{F}(\mathbf{u}^k)|| < \epsilon||\mathbf{F}(\mathbf{u}^0)|| \,, \tag{5.6}$$

where ϵ is a user defined tolerance parameter. Newton's method is *consistent* in that all components of the nonlinear functions that describe the physics we are modeling are updated simultaneously in the iteration sequence with respect to one another. This is in comparison to *inconsistent* strategies, such as a pressure correction strategy for solving the Navier-Stokes equations (Pletcher et al., 1997), where the components of $\mathbf{u}$ are updated in a staggered fashion depending on the particular equations that they are associated with.

5.2 Inexact Newton Methods

Inexact Newton methods arise when the Jacobian operator is not exactly inverted, resulting in an inexact Newton correction as initially described by Dembo and others (Dembo et al., 1982). For common sparse nonlinear systems, which in turn yield a sparse Jacobian matrix, this situation occurs when conventional iterative methods are applied. In their definition, Dembo formulated inexact methods such that they are independent of the linear method used to solve for the Newton correction and therefore are amenable to use with any linear solver. Furthermore, they bind the convergence of the outer nonlinear iteration to the inner linear iteration such that:

$$||\mathbf{J}(\mathbf{u}^k)\delta\mathbf{u}^k + \mathbf{F}(\mathbf{u}^k)|| \leq \eta^k||\mathbf{F}(\mathbf{u}^k)|| \,, \tag{5.7}$$

where $\eta^k \in [0, 1)$ is defined as the *forcing term* at the k iterate. Eq (5.7) then states that the residual generated by the linear solver is bound by the nonlinear residual and how tightly it is bound is defined by the forcing term. This is useful in that we can vary how tightly coupled the convergence of the linear iterations used to generate the Newton correction is to the nonlinear iteration by relaxing or tightening the convergence properties on the linear iterative method. As a result, strategies for determining the forcing term can vary depending on the problem type and can greatly affect the convergence of the method or even prohibit convergence (Eisenstat and

Walker, 1996). In addition, *globalization methods* may be used to modify the Newton correction in a more desire able direction such that convergence properties can be improved when the initial guess for $\mathbf{u}$ is poor (Pawlowski et al., 2006).

Newton-Krylov Methods

A form of inexact Newton methods, *Newton-Krylov methods* are nonlinear iterative methods that leverage a Krylov subspace method as the linear solver for generating the Newton correction (Kelley, 1995). As we investigated in Chapter 2, Krylov methods are robust and enjoy efficient parallel implementations on modern architectures. Furthermore, their lack of explicit dependence on the operator make them easier to implement than other methods. Additionally, although many iterations can become memory intensive due to the need to store the Krylov subspace for the orthogonalization procedure, at each nonlinear iteration this cost is reset as the Jacobian matrix will change due to its dependence on the solution vector. This means that for every nonlinear iteration, a completely new linear system is formed for generating the Newton correction and we can modify the Krylov solver parameters accordingly to accommodate this. In most nonlinear problems, the Jacobian operator is generally non-symmetric and therefore either Krylov methods with long recurrence relations that can handle non-symmetric systems must be considered or the Newton correction system must be preconditioned such that the operator is symmetric and short recurrence relation methods can be potentially be used.

With many Krylov methods available, which to use with the Newton method is dependent on many factors including convergence rates and memory usage. Several studies have been performed to investigate this (McHugh and Knoll, 1993; Knoll and McHugh, 1995). In their numerical studies in 1995, Knoll and McHugh used the set of highly nonlinear and stiff convection-diffusion-reaction equations to solve a set of tokamak plasma problems with the goal of measuring solver performance with Newton's method. They note several trade offs in using Krylov methods with the Newton solver. The first is that the optimization condition that results from the constraints (e.g. the minimization of the GMRES residual over the Krylov space) can be relaxed by restricting the size of the subspace such that only a fixed number of subspace vectors may be maintained, thus reducing memory requirements. We can also relax the optimization condition by instead restarting the recurrence relation with a new set of vectors once a certain number of vectors have been generated. The optimization condition is maintained over that particular set of vectors, however,

Knoll and McHugh note that this ultimately slows the convergence rate as compared to keeping all vectors as the new set of vectors is not necessarily orthogonal to the previous set, and therefore not optimal over the entire iteration procedure. The orthogonality condition can be relaxed by using a recurrence relation that does not generate a strictly orthonormal basis for the Krylov subspace such as the Lanzcos biorthogonalization procedure, resulting in memory savings due to the shorter Lanzcos recurrence relation.

As a comparison, Knoll and McHugh chose an Arnoldi-based GMRES with a fixed vector basis approximately the size of the number of iterations required to converge as the long recurrence relation solver and conjugate gradients squared (CGS), bi-orthogonalized conjugate gradient stabilized (Bi-CGSTAB), and transpose-free quasiminimal residual (TFQMR) methods as Lanzcos-based short recurrence relation solvers. All solvers were used to compute the right-preconditioned Newton correction system. For standard implementations of Newton's method where the Jacobian operator was explicitly formed using difference equations, all methods exhibited roughly equivalent iteration count performance for both the inner linear iterations and the outer nonlinear iterations in terms of iterations required to converge. Bi-CGSTAB typically performed the best for implementations where the Jacobian was explicitly formed and GMRES performing best for matrix-free implementations. However, upon investigating the convergence of the inner iterations, it was observed that the GMRES solver was significantly more robust, always generating a monotonically decreasing residual as compared to the Lanzcos-based methods which had the tendency to oscillate. Based on these results, in all of their future work Knoll and McHugh tended to use GMRES as the Krylov solver (Knoll and Keyes, 2004).

Jacobian-Free Approximation

In most cases, the Jacobian is difficult to form from the difference equations and costly to evaluate for large equation sets. For simple nonlinear cases such as the Navier-Stokes equations, the derivatives can be computed and coded, but due to the complexity of those derivatives and the resulting difference equations this task can be tedious, error prone, and must be repeated for every equation set. Furthermore, in their 1995 work, Knoll and McHugh also noted that a dominating part of their computation time was the evaluation of the difference equations for building the Jacobian (Knoll and McHugh, 1995). By recognizing that Krylov methods only need the action of the operator on the vector instead of the operator itself, the Jacobian can instead be approximated through various numerical methods including a difference-based

Jacobian-free formulation.

Jacobian-Free methods, and in particular *Jacobian-Free Newton-Krylov* (JFNK) methods (Knoll and Keyes, 2004), rely on forming the action of the Jacobian on a vector as required by the Krylov solver through a forward difference scheme. In this case, the action of the Jacobian on some vector $\mathbf{v}$ is given as:

$$\mathbf{J}(\mathbf{u})\mathbf{v} = \frac{\mathbf{F}(\mathbf{u} + \epsilon\mathbf{v}) - \mathbf{F}(\mathbf{u})}{\epsilon}\,, \tag{5.8}$$

where ϵ is a small number typically on the order of machine precision. Kelley (Kelley, 1995) points out a potential downfall of this formulation in that if the discretization error in $\mathbf{F}(\mathbf{u})$ is on the order of the perturbation parameter ϵ, then the finite difference error from Eq (5.8) pollutes the solution. In addition, Knoll and McHugh noted that for preconditioning purposes, part of the Jacobian must still explicitly be formed periodically and that linear solver robustness issues were magnified by the matrix-free approach due to the first-order approximation. This formation frequency coupled with the numerous evaluations of the Jacobian approximation create a situation where after so many nonlinear iterations, it becomes cheaper to instead fully form the Jacobian. For simple equation sets, this may only take 5-10 Newton iterations to reach this point while over 30 may be required for larger equations sets and therefore larger Jacobians.

Automatic Differentiation for Jacobian Generation

If it is acceptable to store the actual Jacobian matrix, other methods are available to construct it without requiring hand-coding and evaluating derivatives, thus eliminating the associated issues. In addition, if any additional equations are added to the system or a higher order functional approximation is desired, it would be useful to avoid regenerating and coding these derivatives. Becoming more prominent in the 1990's, *automatic differentiation* is a mechanism by which the derivatives of a function can be generated automatically by evaluating it. Automatic differentiation is built on the concept that all functions discretely represented in a computer are ultimately represented by elementary mathematical operations. If the chain rule is applied to those elementary operations, then the derivatives of those functions can be computed to the order of accuracy of their original discretization in a completely automated way (Averick et al., 1994).

The work of Bartlett and others (Bartlett et al., 2006) extended initial Fortran-based work in the area of automatic differentiation implementations to leverage the parametric type and operator overloading features of C++ (Stroustrup, 1997).

They formulate the differentiation problem from an element viewpoint by assuming that a global Jacobian can be assembled from local element function evaluations of $e_k : \mathbb{R}^{n_k} \rightarrow \mathbb{R}^{m_k}$, similar to the finite element assembly procedure as:

$$\mathbf{J}(\mathbf{u}) = \sum_{i=1}^{N} \mathbf{Q}_i^T \mathbf{J}_k \mathbf{P}_i \,, \tag{5.9}$$

where $\mathbf{J}_{k_i} = \partial e_{k_i} / \partial P_i u$ is the k^{th} element function Jacobian, $\mathbf{Q} \in \mathbb{R}^{n_{k_i} \times N}$ is a projector onto the element domain and $\mathbf{P} \in \mathbb{R}^{m_{k_i} \times N}$ a projector onto the element range for $\mathbf{F}(\mathbf{u}) \in \mathbb{R}^{N \times N}$. The Jacobian matrix for each element will therefore have entirely local data in a dense structure, eliminating the need for parallel communication and sparse techniques during differentiation. Only when all local differentials are computed does communication of the Jacobian occur through gather/scatter operations in order to properly assembly it. Also of benefit is the fact that element-level computations generally consist of a smaller number of degrees of freedom, thus reducing memory requirements during evaluation as compared to a global formulation of the problem. Such a formulation is not limited to finite element formulations and is amenable to any scheme where the system is globally sparse with degrees of freedom coupled to local domains including finite volume representations. The templating capabilities of C++ were leveraged with the element-based evaluation and assembly scheme as in Eq (5.9) by templating element function evaluation code on the evaluation type. If these functions are instantiated with standard floating point types then the residual is returned. If they are instead instantiated with the operator-overloaded automatic differentiation types, both the residual and Jacobian are returned.

Of interest to Bartlett, Averick, and the many others that have researched automatic differentiation are measures of its performance relative to hand-coded derivatives and capturing the Jacobian matrix from matrix-free approximations. Given their element-based function evaluation scheme, Bartlett's work varied the number of degrees of freedom per element and compared both the floating point operation count and CPU time for both the templated automatic differentiation method and hand-coded derivatives for Jacobian evaluations. Although they observed a 50% increase in floating point operations in the templated method over the hand-coded method, run times were observed to be over 3 times faster for the templated method. They hypothesize that this is due to the fact that the element-based formulation of the templated method is causing better utilization of cache and therefore faster data access. Furthermore, they observed linear scaling behavior for automatic differentiation as the number of degrees of freedom per element were increased up to a few hundred. Based on these

results, this type of automatic differentiation formulation was deemed acceptable for use in large-scale, production physics codes.

5.3 The FANM Method

In production physics codes based on nonlinear equations sets, Newton-Krylov methods are the primary means of generating a fully consistent solution scheme (Evans et al., 2006, 2007; Gaston et al., 2009; Godoy and Liu, 2012). Typically, for large scale simulations these problems are memory limited due to the subspaces generated by robust Krylov methods. Often, a matrix-free approach is chosen to relax memory requirements over directly generating the Jacobian matrix and facilitate the implementation. However, as we observed in previous sections, these matrix-free methods suffer from poorly scaled problems and the first order error introduced by the Jacobian approximation. In addition, it was observed that the savings induced by the matrix-free approach is eventually amortized over a number of nonlinear iterations where it becomes more efficient computationally to instead form the Jacobian.

In Chapter 3, we focused our efforts on developing and improving Monte Carlo methods for inverting linear systems. These methods, when used to accelerate a stationary method in MCSA, enjoy a robust implementation and exponential convergence rates. Further, the only storage required is that of the full linear system including the linear operator so that we may generate transition probabilities for the random walk sequence. Although this requires more storage to represent the linear system than that of a Krylov method where the operator is not required, we do not incur any additional storage costs once the iteration sequence begins. In the context of nonlinear problems, the Jacobian matrix that we are required to generate for the Monte Carlo solvers may be generated at will from the nonlinear functions in the Newton system using automatic differentiation. Not only do we then have a simple and automated way to generate the Jacobian, but we also enjoy a Jacobian of numerical precision equivalent to that of our function evaluations. We therefore propose the *Forward-Automated Newton-MCSA* (FANM) method that utilizes all of the above components. We hypothesize that such a method will be competitive with Newton-Krylov methods not only from a convergence and timing perspective, but also relax scaling requirements of matrix-free methods and memory costs of both matrix-free and fully formed Jacobian methods to allow the application developer to solve problems of finer discretization and higher-order functional representations while maintaining a robust and efficient parallel implementation.

Jacobian Storage vs. Subspace Storage and Restarts

To gauge the memory benefits of a FANM method over a Krylov-based scheme, we must look at the storage requirements of the Jacobian matrix as compared to the Krylov subspace vectors for sparse linear problems. Saad's text provides us relations for both of these cases (Saad, 2003). Beginning with the Jacobian storage, for sparse problems production linear algebra library implementations utilize *compressed row storage* to efficiently store the non-zero components of a sparse matrix while still allowing for all standard parallel matrix computations to be performed. Per § 2.4, recall that efficient parallel matrix-vector multiplications rely on processors knowing which other processors contain their neighboring matrix and vector elements. Typically this information is represented by a graph which must be stored along with the matrix elements themselves. To store the elements, consider the following sparse matrix:

$$\mathbf{A} = \begin{bmatrix} 2 & 0 & 8 & 0 & 0 & 0 \\ 4 & 5 & 0 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 1 & 0 \\ 0 & 0 & 3 & 7 & 0 & 2 \\ 0 & 0 & 0 & 4 & 9 & 0 \\ 0 & 0 & 0 & 0 & 9 & 1 \end{bmatrix}.$$

The compressed form of $\mathbf{A}$ would then be:

$\mathbf{A}$ *values* :	2	8	4	5	1	2	1	1	3	7	2	4	9	9	1
column :	1	3	1	2	4	2	3	5	3	4	6	4	5	5	6
row start :	1	3	6	9	12	14	16								

where the first row contains the non-zero values of the matrix $\mathbf{A}$ starts at, the second row contains the column index of those values, and the third row contains the index of the A value and column index rows that each matrix row starts at ending with the start of the next row. For this simple example, we actually break even by storing 36 elements in the full matrix and 36 pieces of data in the compressed format. However, for large, sparse matrices there is a significant storage savings using the compressed format.

In his discussion on orthogonalization schemes for Krylov subspaces, Saad provides us with space and time complexities for these operations. To find a solution vector $\mathbf{x} \in \mathbb{R}^{N \times N}$ with m Krylov iterations (and therefore $m+1$ subspace vectors) the storage requirement is then $(m+1)N$ for most common orthogonalization techniques.

Using our simple example above (even though we did not see any gains in storage over the full matrix representation), if it takes 10 GMRES iterations using Arnoldi orthogonalization to converge to a solution, then a total of 66 pieces of data are required to be stored instead of 36 for the explicitly formed matrix case (we do not consider graph storage here for comparison as both methods will require the graph for parallel operations). For larger sparse matrices, this disparity in memory requirements will be even greater, especially for ill-conditioned systems that require many GMRES iterations to converge.

Parallel FANM Implementation

Like the parallelization of the MCSA method described in § **??**, a parallel FANM method relies on a basic set of parallel matrix-vector operations as well as the global residual and Jacobian assembly procedure described in § 5.2. Consider the Newton iteration scheme in Eq (5.5). We must first assemble the linear system in parallel through the element-wise function evaluations to generate both the global Jacobian operator and the global residual vector on the right hand side. Per Bartlett's work, efficient and automated parallel mechanisms are available to do this through a sequence of scatter/gather operations. With these tools available for residual and Jacobian generation, the remainder of the parallel procedure is simple, with the linear Newton correction system solved using the parallel MCSA method as previously described and the Newton correction applied to the previous iterate's solution through a parallel vector update operation.

As Newton methods are formulated independent of the inner linear solver generating the Newton corrections, the actual performance of the nonlinear iterations using MCSA is expected to be similar to that of traditional Newton-Krylov methods. Furthermore, we expect to achieve numerically identical answers with a Newton-MCSA method as other Newton methods and we should indeed verify this. The parallel performance of such a method will inherently be bound to the parallel Monte Carlo implementation of the linear solver as the parallel operations at the nonlinear iteration level are identical to those that you would perform with a Newton-Krylov method. Matrix-free formulations will have different parallel performance than these methods, and therefore we should compare FANM performance to JFNK-based schemes. More important than performance in this situation is the reduced memory pressure that a FANM implementation provides, as discussed in § 5.3, because a FANM method will not generate a subspace in the linear solver and compressed storage for sparse matrices

are utilized, we expect significant memory savings over Newton-Krylov methods. We must measure the memory utilization of both of these methods in order to quantify their differences and provide additional analysis of the FANM method's merits, or lack thereof.

5.4 Monte Carlo Solution Methods for the Navier-Stokes Equations

FANM Verification

To verify the FANM method for nonlinear problems, we choose benchmark solutions for the 2-dimensional, steady, incompressible Navier-Stokes equations on a rectilinear grid in much the same way as Shadid and Pawlowski's work on Newton-Krylov methods for the solution of these equations (Shadid et al., 1997; Pawlowski et al., 2006). We define these equations as follows:

$$\rho \mathbf{u} \cdot \nabla \mathbf{u} - \nabla \cdot \mathbf{T} - \rho \mathbf{g} = \mathbf{0} \tag{5.10a}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{5.10b}$$

$$\rho C_p \mathbf{u} \cdot \nabla T + \nabla \cdot \mathbf{q} = 0 \,, \tag{5.10c}$$

where ρ is the fluid density, $\mathbf{u}$ is the fluid velocity, $\mathbf{g}$ gravity, C_p the specific heat capacity at constant pressure of the fluid and T the temperature of the fluid. Eq (5.10a) provides momentum transport, Eq (5.10b) provides the mass balance, and Eq (5.10c) provides energy transport with viscous dissipation effects neglected. In addition, we close the system with the following equations:

$$\mathbf{T} = -P\mathbf{I} + \mu[\nabla \mathbf{u} + \nabla \mathbf{u}^T] \tag{5.11a}$$

$$\mathbf{q} = -k \nabla T \,, \tag{5.11b}$$

where $\mathbf{T}$ is the stress tensor, P is the hydrodynamic pressure, μ is the dynamic viscosity of the fluid, $\mathbf{q}$ is the heat flux in the fluid, and k is the thermal conductivity of the fluid. This set of strongly coupled equations possesses both the nonlinearities and asymmetries that we are seeking for qualification of the FANM method. Further, physical parameters within these equations can be tuned to enhance the nonlinearities. We will then apply these equations to the following three standard benchmark problems.

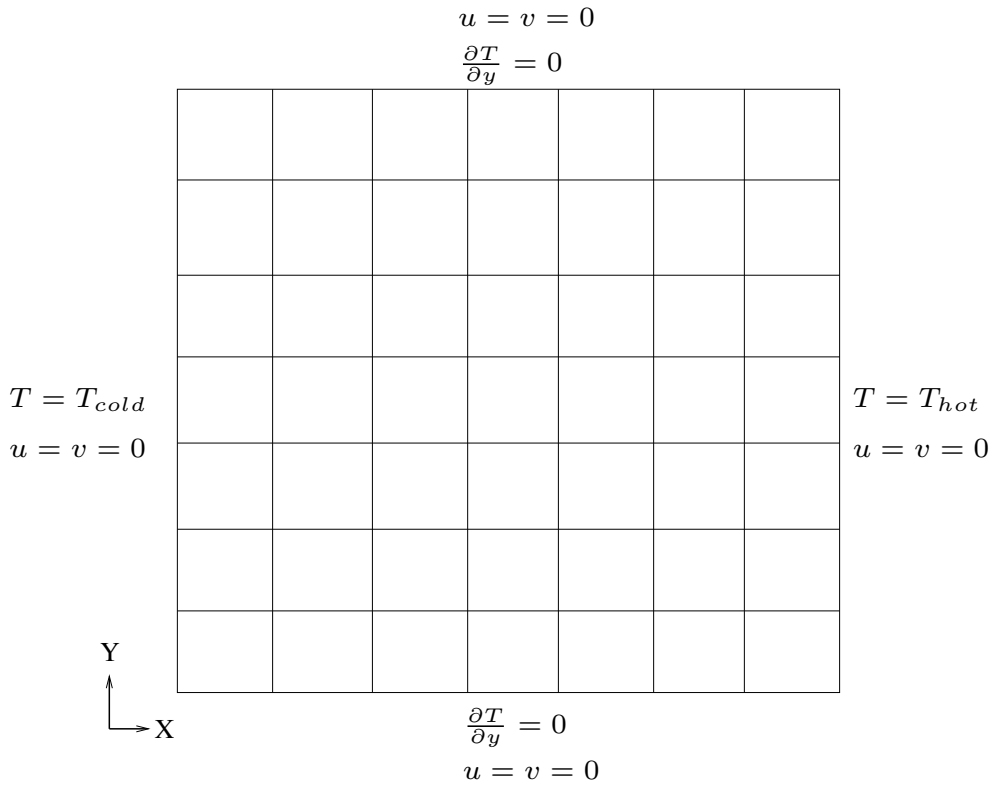

Figure 5.1: **Problem setup for the natural convection cavity benchmark.** *Dirichlet conditions are set for the temperature on the left and right while Neumann conditions are set on the top and bottom of the Cartesian grid. The temperature gradients will cause buoyancy-driven flow. Zero velocity Dirichlet conditions are set on each boundary. No thermal source was present.*

Thermal Convection Cavity Problem

In 1983 a benchmark solution for the natural convection of air in a square cavity was published (De Vahl Davis, 1983) as shown in Figure 5.1 for the solution of the energy, mass, and momentum equations. In this problem, a rectilinear grid is applied to the unit square. No heat flow is allowed out of the top and bottom of the square with a zero Neumann condition specified. Buoyancy driven flow is generated by the temperature gradient from the cold and hot Dirichlet conditions on the left and right boundaries of the box. By adjusting the Rayleigh number of the fluid (and therefore adjusting the ratio of convective to conductive heat transfer), we can adjust the influence of the nonlinear convection term in Eq (5.10a). In Shadid's work, Rayleigh numbers of up to 1×10^6 were used for this benchmark on a 100×100 square mesh grid.

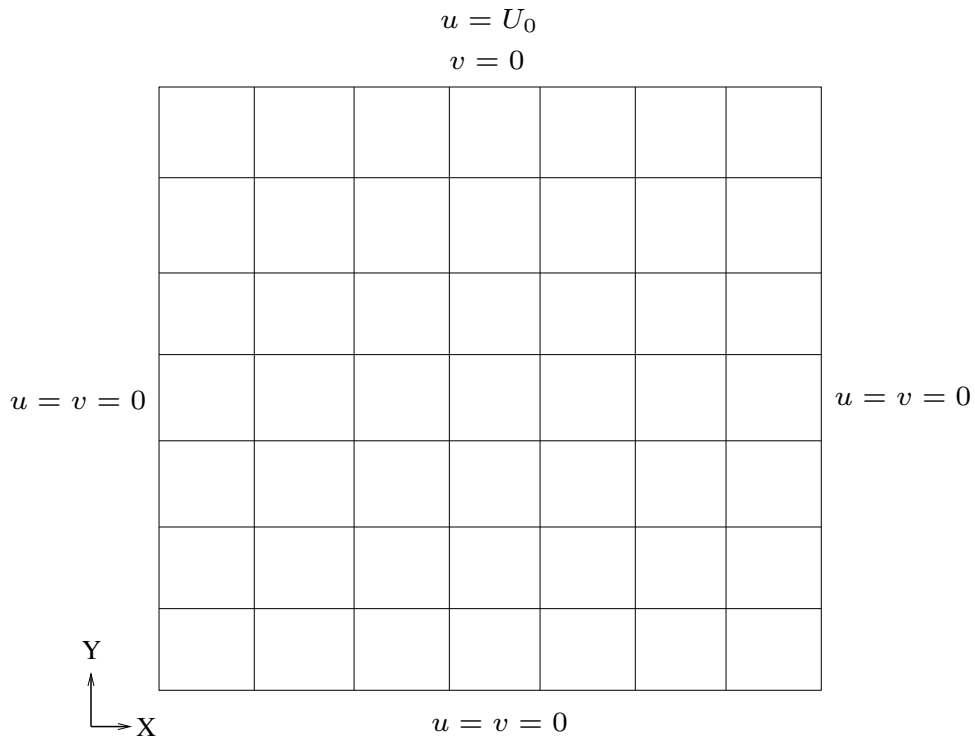

Figure 5.2: **Problem setup for the lid driven cavity benchmark.** *Dirichlet conditions of zero are set for the velocity on the left and right and bottom while the Dirichlet condition set on the top provides a driving force on the fluid.*

Lid Driven Cavity Problem

As an extension of the convection problem, the second benchmark problem given by Ghia (Ghia et al., 1982) adds a driver for the flow to introduce higher Reynolds numbers into the system, providing more inertial force to overcome the viscous forces in the fluid. The setup for this problem is equally simple, containing only the Dirichlet conditions as given in Figure 5.2 and is only applied to the mass and momentum equations on the unit square. The top boundary condition will provide a driver for the flow and its variation will in turn vary the Reynolds number of the fluid. An increased velocity will generate more inertial forces in the fluid, which will overcome the viscous forces and again increase the influence of the nonlinear terms in Eq (5.10a). Shadid also used a 100×100 grid for this benchmark problem with Reynolds numbers up to 1×10^4.

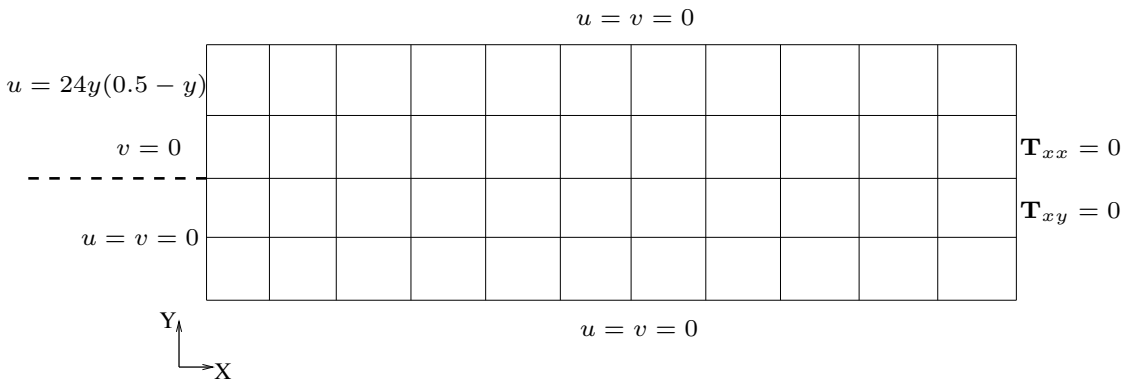

Figure 5.3: **Problem setup for the backward facing step benchmark.** *Zero velocity boundary conditions are applied at the top and bottom of the domain while the outflow boundary condition on the right boundary is represented by zero stress tensor components in the direction of the flow. For the inlet conditions, the left boundary is split such that the top half has a fully formed parabolic flow profile and the bottom half has a zero velocity condition, simulating flow over a step.*

Backward-Facing Step Problem

The third benchmark was generated by Gartling in 1990 and consists of both flow over a backward step and an outflow boundary condition (Gartling, 1990). Using the mass and momentum equations while neglecting the energy equation, this problem utilizes a longer domain with a 1/30 aspect ratio with the boundary conditions as shown in Figure 5.3. In this problem, the inflow condition is specified by a fully-formed parabolic flow profile over a zero velocity boundary representing a step. The flow over this step will generate a recirculating backflow under the inlet flow towards the step. As in the lid driven cavity problem, the nonlinear behavior of this benchmark and the difficulty in obtaining a solution is dictated by the Reynolds number of the fluid. In Shadid's work, a 20×400 non-square rectilinear grid was used to discretize the domain with Reynolds number up to 5×10^2.

Chapter 6

Parallel Monte Carlo Solution Methods for Linear Systems

For Monte Carlo methods, and in particular MCSA, to be viable at the production scale, scalable parallel implementations are required. In a general linear algebra context for discrete systems, this has not yet been achieved. Therefore, we will develop and implement parallel algorithms for these methods leveraging both the knowledge gained from the general parallel implementations of Krylov methods and modern parallel strategies for Monte Carlo as developed by the reactor physics community. In order to formulate a parallel MCSA algorithm, we recognize that the algorithm occurs in two stages, an outer iteration performing Richardson's iteration and applying the correction, and an inner Monte Carlo solver that is generating the correction via the adjoint method. The parallel aspects of both these components must be considered.

In this chapter we briefly review particle transport methods for domain decomposed Monte Carlo and then provide an analytic framework from which to estimate its performance. We then devise an algorithm based on these methods for efficient parallel solutions using the adjoint Neumann-Ulam method and MCSA and perform parallel scaling studings to test its performance and compare this performance to conventional methods.

6.1 Domain Decomposition for Monte Carlo

As observed in the discussion on parallel Krylov methods, large-scale problems will surely have their data partitioned such that each parallel process owns a subset of the equations in the linear system. Given this convention, the adjoint Monte Carlo algorithm must perform random walks over a domain that is decomposed and must remain decomposed due to memory limitations. This naturally leads us to seek parallel Monte Carlo algorithms that handle domain decomposition.

In the context of radiation transport, Brunner and colleagues provided a survey of algorithms for achieving this as implemented in production implicit Monte Carlo codes (Brunner et al., 2006). In their work they identify two data sets that are required to be communicated: the sharing of particles that are transported from one domain

to another and therefore from one processor to another and a global communication that signals if particle transport has been completed on all processors. The algorithms presented are a fully-locking synchronous scheme, an asynchronous-send/synchronous-receive pattern, a traditional master/slave scheme, and a modified master/slave scheme the implements a binary tree pattern for the global reduction type operations needed to communicate between the master and slave processes. They observed that the modified master/slave scheme performed best in that global communications were implemented more efficiently than those required by the asynchronous scheme. Furthermore, none of these schemes handled load-imbalanced cases efficiently. Such cases will be common if the source sampled in the Monte Carlo random walk is not isotropic and not evenly distributed throughout the global domain. It was noted that efficiencies were improved by increasing the frequency by which particle data was communicated between domain-adjacent processors. However, this ultimately increases communication costs. In 2009, Brunner extended his work by using a more load-balanced approach with a fully asynchronous communication pattern (Brunner and Brantley, 2009). Although the extended implementation was more robust and allowed for scaling to larger numbers of processors, performance issues were still noted with parallel efficiency improvements needed in both the weak and strong scaling cases for unbalanced problems. These results led Brunner to conclude that a combination of domain decomposition and domain replication could be used to solve some of these issues.

Multiple-Set Overlapping-Domain Decomposition

In 2010, Wagner and colleagues developed the *multiple-set overlapping-domain* (MSOD) decomposition for parallel Monte Carlo applications for full-core light water reactor analysis (Wagner et al., 2010). In their work, an extension of Brunner's, their scheme employed similar parallel algorithms for particle transport but a certain amount of overlap between adjacent domains was used to decrease the number of particles leaving the local domain. In addition, Wagner utilized a level of replication of the domain such that the domain was only decomposed on $O(100)$ processors and if replicated $O(1,000)$ times achieves simulation on $O(100,000)$ processors, thus providing spatial and particle parallelism. Each collection of processors that constitutes a representation of the entire domain is referred to as a set, and within a set overlap occurs among its sub-domains. The original motivation was to decompose the domain in a way that it remained in a physical cabinet in a large distributed machine, thus reducing

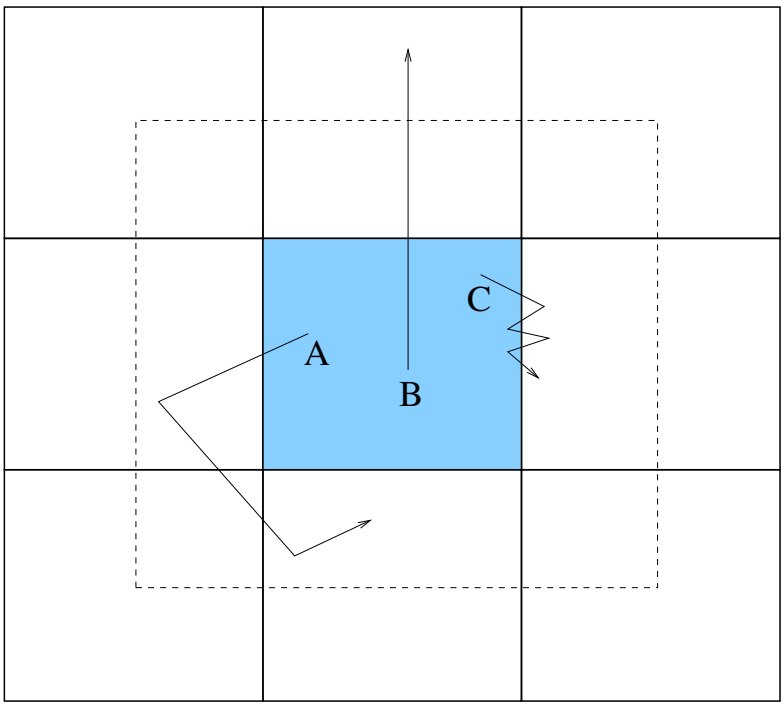

Figure 6.1: **Overlapping domain example illustrating how domain overlap can reduce communication costs.** *All particles start in the blue region of interest. The dashed line represents 0.5 domain overlap between domains.*

latency costs during communication. A multiple set scheme is also motivated by the fact that communication during particle transport only occurs within a set, limiting communications during the transport procedure to a group of $O(100)$ processors, a number that was shown to have excellent parallel efficiencies in Brunner's work and therefore will scale well in this algorithm. The overlapping domains within each set also demonstrated reduced communication costs. On each processor, the source is sampled in the local domain that would exist if no overlap was used while tallies can be made over the entire overlapping domain.

To demonstrate this, consider the example adapted from Mervin's work with Wagner and others in the same area (Mervin et al., 2012) and presented in Figure 6.1. In this example, 3 particle histories are presented emanating from the blue region of interest. Starting with particle A, if no domain overlap is used then the only the blue domain exists on the starting processor. Particle A is then transported through 3 other domains before the history ends, therefore requiring three communications to occur in Brunner's algorithm. If a 0.5 domain overlap is permitted as shown by the dashed line, then the starting process owns enough of the domain such that no communications

must occur in order to complete the particle A transport process. Using 0.5 domain overlap also easily eliminates cases such as the represented by the path of particle C. In this case, particle C is scattering between two adjacent domains, incurring a large latency cost for a single particle. Finally, with particle B we observe that 0.5 domain overlap will still not eliminate all communications. However, if 1 domain overlap were used, the entire geometry shown in Figure 6.1 would be contained on the source processor and therefore transport of all 3 particles without communication would occur.

Wagner and colleagues used this methodology for a 2-dimensional calculation of a pressurized water reactor core and varied the domain overlap from 0 to 3 domain overlap (a 7×7 box in the context of our example) where a domain constituted a fuel assembly. For the fully domain decomposed case, they observed that 76.12% of all source particles leave the domain. At 1.5 domain overlap, the percentage of source particles born in the center assembly leaving the processor domain dropped to 1.05% and even further for 0.02% for the 3 domain overlap. Based on these results, this overlap approach, coupled with the multiple sets paradigm that will scale for existing parallel transport algorithms, provides a scalable Monte Carlo algorithm for today's modern machines.

6.2 An Analytic Performance Framework for Domain-Decomposed Monte Carlo

With Wagner's work (Wagner et al., 2010), we observed that domain overlap implemented with a domain decomposed Monte Carlo setting reduces the amount of nearest neighbor communication that must occur during transport. To date, parallel Neumann-Ulam methods have been limited to full domain replication with parallelism exploited through individual histories (Alexandrov, 1998) and in the previous chapter we exploited Wagner's work to alleviate this. To accomplish this, we recognized from the literature that stochastic histories must be transported from domain to domain as the simulation progresses and they transition to states that are not in the local domain. Because we have chosen a domain decomposition strategy in a parallel environment, this means that communication of these histories must occur between compute nodes owning neighboring pieces of the global domain as outlined in our algorithms. We wish to characterize this communication not only because communication is in general expensive, but also because these nearest-neighbor communication sequences, specifi-

cally, have poor algorithmic strong scaling (Gropp et al., 2001) in much the same way as a matrix-vector multiply operation. Therefore, we desire a framework to provide a simple, analytic theory based on the properties of the linear system that will allow for estimates of the domain decomposed behavior of the adjoint Neumann-Ulam method in terms of communication costs.

Leakage Fractions for Symmetric Systems

When solving problems where the linear operator is symmetric, a host of analytic theories exist based on the eigenvalue spectrum of the operator that characterize their behavior in the context of deterministic linear solvers. Using past work, these theories are adapted to the domain decomposed adjoint Neumann-Ulam method using the one-speed, two-dimensional neutron diffusion equation and spatial discretization presented in § 4.4. Using the linear system generated by the discretization of the model problem, we use a spectral analysis to generate analytic relations for the eigenvalues of the operator based on system parameters. Using the eigenvalue spectra, we then build relationships to characterize the transport of stochastic histories in a decomposed domain and the fraction of histories that leak from a domain and will therefore have to be communicated. Finally, we compare these analytic results to numerical experiments conducted with the model problem.

Spectral Analysis

The convergence of the Neumann series in Eq (3.32) approximated by the Monte Carlo solver is dependent on the eigenvalues of the iteration matrix. We will compute these eigenvalues by assuming eigenfunctions of the form (LeVeque, 2007):

$$\Phi_{p,q}(x,y) = e^{2\pi\imath px} e^{2\pi\imath qy} \,, \tag{6.1}$$

where different combinations of p and q represent the different eigenmodes of the solution. As these are valid forms of the solution, then the action of the linear operator on these eigenfunctions should give the eigenvalues of the matrix as they exist on the unit circle in the complex plane.

Iteration Matrix Spectrum

For the model problem, we first compute the eigenvalues for the diffusion operator $\mathbf{D}$ by applying the operator to the eigenfunctions and noting that $x = ih$ and $y = jh$:

$$\begin{aligned}\mathbf{D}\Phi_{p,q}(x,y) = \lambda_{p,q}(\mathbf{D}) = \\ -\frac{D}{6h^2}\Big[&4e^{-2\pi\imath ph} + 4e^{2\pi\imath ph} + 4e^{-2\pi\imath qh} + 4e^{2\pi\imath qh} + e^{-2\pi\imath ph}e^{-2\pi\imath qh} \\ &+ e^{-2\pi\imath ph}e^{2\pi\imath qh} + e^{2\pi\imath ph}e^{-2\pi\imath qh} + e^{2\pi\imath ph}e^{2\pi\imath qh} - 20\Big] + \Sigma_a\,. \quad (6.2)\end{aligned}$$

Using Euler's formula, we can collapse the exponentials to trigonometric functions:

$$\lambda_{p,q}(\mathbf{D}) = -\frac{D}{6h^2}[8\cos(\pi ph) + 8\cos(\pi qh) + 4\cos(\pi ph)\cos(\pi qh) - 20] + \Sigma_a\,. \quad (6.3)$$

As Eq (4.18) is diagonally dominant, point Jacobi preconditioning as outlined in §**??** is sufficient to reduce the spectral radius of the iteration matrix below unity and therefore ensure convergence of the Neumann series. Applying this preconditioner, we are then solving the following diffusion system:

$$\mathbf{M}^{-1}\mathbf{D}\boldsymbol{\phi} = \mathbf{M}^{-1}\mathbf{s}\,. \quad (6.4)$$

The operator $\mathbf{M}^{-1}\mathbf{D}$ is merely the original diffusion operator with each row scaled by the diagonal component. As we have defined a homogeneous domain, the scaling factor, α, is the same for all rows in the operator and defined as the $\phi_{i,j}$ coefficient from Eq (4.22):

$$\alpha = \left[\frac{10D}{3h^2} + \Sigma_a\right]^{-1}\,. \quad (6.5)$$

Using this coefficient, we then have the following spectrum of preconditioned eigenvalues:

$$\lambda_{p,q}(\mathbf{M}^{-1}\mathbf{D}) = \alpha\lambda_{p,q}(\mathbf{D})\,. \quad (6.6)$$

The spectral radius of the iteration matrix is obtained by seeking its largest eigenvalue. As with the diffusion operator, we can use the same analysis techniques to find the eigenvalues for the iteration matrix. We use a few simplifications by noting that if the Jacobi preconditioned iteration matrix is $\mathbf{H} = \mathbf{I} - \mathbf{M}^{-1}\mathbf{D}$, the we except all terms on the diagonal of the iteration matrix to be zero such that we have the

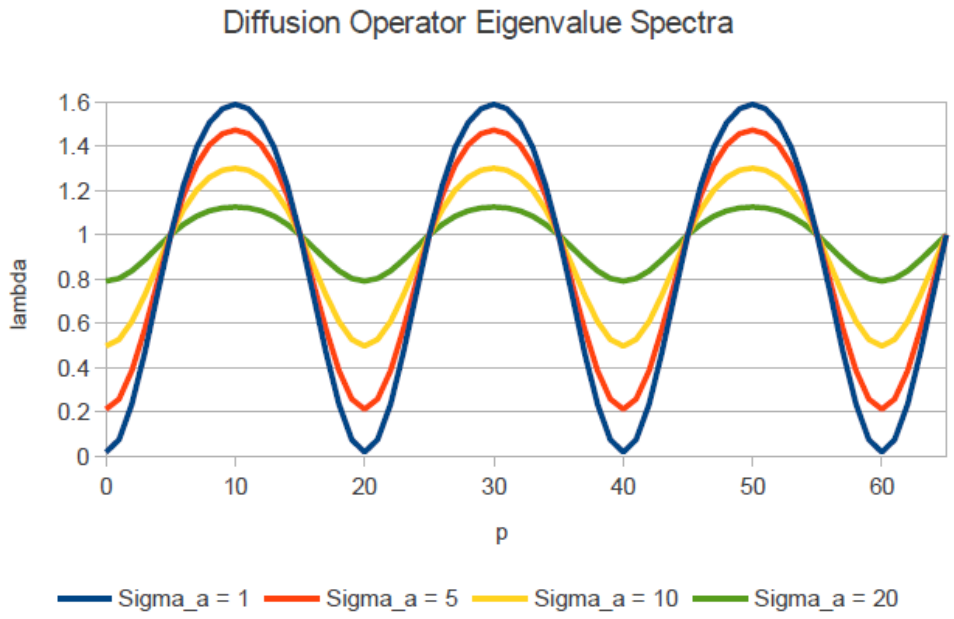

Figure 6.2: **Eigenvalue spectra for the diffusion equation.**

following stencil:

$$\mathbf{H}\boldsymbol{\phi} = \frac{\alpha D}{6h^2}\Big[4\phi_{i-1,j} + 4\phi_{i+1,j} + 4\phi_{i,j-1} + 4\phi_{i,j+1} + \phi_{i-1,j-1} + \phi_{i-1,j+1} + \phi_{i+1,j-1} + \phi_{i+1,j+1}\Big]. \quad (6.7)$$

Inserting the eigenfunctions defined by Eq (6.1) we get:

$$\lambda_{p,q}(\mathbf{H}) = \frac{\alpha D}{6h^2}\Big[4e^{-2\pi\imath ph} + 4e^{2\pi\imath ph} + 4e^{-2\pi\imath qh} + 4e^{2\pi\imath qh} + e^{-2\pi\imath ph}e^{-2\pi\imath qh} + e^{-2\pi\imath ph}e^{2\pi\imath qq} + e^{2\pi\imath ph}e^{-2\pi\imath qh} + e^{2\pi\imath ph}e^{2\pi\imath qh}\Big], \quad (6.8)$$

which simplifies to:

$$\lambda_{p,q}(\mathbf{H}) = \frac{\alpha D}{6h^2}[8\cos(\pi ph) + 8\cos(\pi qh) + 4\cos(\pi ph)\cos(\pi qh)], \quad (6.9)$$

giving the eigenvalue spectrum for the Jacobi preconditioned iteration matrix. To find these maxium eigenvalue, Eq (6.6) is plotted as a function of p with $p = q$ in Figure 6.2 for various values of Σ_a. We find that the maximum eigenvalue exists when $p = q = 0$, giving the following for the spectral radius of the Jacobi preconditioned

iteration matrix:

$$\rho(\mathbf{H}) = \frac{10\alpha D}{3h^2} \,. \tag{6.10}$$

Neumann Series Convergence

As outlined in §3.1, the adjoint Monte Carlo method is effectively an approximation to a stationary method. In the adjoint Neumann-Ulam method, k iterations, equivalent to k applications of the iteration matrix, are approximated by a random walk of average length k to yield the summation in Eq (3.33) (Dimov et al., 1998; Danilov et al., 2000). This random walk length, or the number of transitions before the termination of a history (either by the weight cutoff, absorption, or exiting the global domain) is therefore approximately the number of stationary iterations required to converge to the specified tolerance. In the case of the adjoint Neumann-Ulam method, no such tolerance exists, however, we have specified a weight cutoff, W_c, that determines when low-weight histories will be prematurely terminated as their contributions are deemed minute. After k iterations, a stationary method is terminated as the error has reached some fraction, ϵ, of the initial error:

$$||\mathbf{e}^k||_2 = \epsilon||\mathbf{e}^0||_2 \,. \tag{6.11}$$

Per Eq (2.22), we see that this fraction is equivalent to $\epsilon = \rho(\mathbf{H})^k$. For the adjoint Neumann-Ulam method, if we take this fraction to be the weight cutoff, a measure of how accurately the contributions of a particular history to the solution are tallied, we then have the following relationship for k:

$$k = \frac{\log(W_c)}{\log(\rho(\mathbf{H}))} \,. \tag{6.12}$$

This then gives us a means to estimate the length of the random walks that will be generated from a particular linear operator based on the eigenvalues of its iteration matrix (independent of the linear operator splitting chosen) and based on the weight cutoff parameter used in the Neumann-Ulam method.

Domain Leakage Approximations

In a domain decomposed situation, not all histories will remain within the domain they started in and must instead be communicated. This communication, expected to be expensive, was analyzed by Siegel and colleagues for idealized, load balanced situations for full nuclear reactor core Monte Carlo neutral particle simulations (Siegel

et al., 2012a). To quantify the number of particles that leak out of the local domain they define a leakage fraction, Λ, as:

$$\Lambda = \frac{average\ \#\ of\ particles\ leaving\ local\ domain}{total\ of\ \#\ of\ particles\ starting\ in\ local\ domain}\ . \tag{6.13}$$

For their studies, Siegel and colleagues assumed that the value of Λ was dependent on the total cross subsection of the system via the Wigner rational approximation. Outlined more thoroughly by Hwang's chapter in (Azmy and Sartori, 2010), we will use both the Wigner rational approximation and the mean chord approximation as a means to estimate the leakage fraction.

In the case of domain decomposed linear operator equations, we can use diffusion theory to estimate the optical thickness of a domain in the decomposition and the corresponding leakage fraction in terms of properties of the linear operator and the discretization. To begin we must first calculate the mean distance a Monte Carlo history will move in the grid by computing the mean squared distance of its movement along the chord of length l defined across the domain. After a single transition a history will have moved a mean squared distance of:

$$\bar{\langle r_1^2 \rangle} = (n_s h)^2\ , \tag{6.14}$$

where h is the size of the discrete grid elements along the chord and n_s is the number of grid elements a history will move on average every transition. For our diffusion model problem, n_s would equate to the expected number of states in the i (or j as the problem is symmetric) direction that a history will move in a single transition and is dependent on the stencil used for the discretization. After k transitions in the random walk, the history will have moved a mean squared distance of:

$$\bar{\langle r_k^2 \rangle} = k(n_s h)^2\ . \tag{6.15}$$

If our chord is of length l and there are n_i grid elements (or states to which a history may transition) along that chord, then $h = l/n_i$ giving:

$$\bar{\langle r_k^2 \rangle} = k\left(\frac{n_s l}{n_i}\right)^2\ . \tag{6.16}$$

From diffusion theory, we expect the average number of interactions along the chord

to be:

$$\tau = \frac{l}{2d\sqrt{\langle \bar{r}_k^2 \rangle}} \,, \tag{6.17}$$

where d is the dimensionality of the problem and $\sqrt{\langle \bar{r}_k^2 \rangle}$ is effectively the mean free path of the Monte Carlo history in the domain. We can readily interpret τ to be the *effective optical thickness* of a domain of length l. Inserting Eq (6.16) we arrive at:

$$\tau = \frac{n_i}{2dn_s\sqrt{k}} \,, \tag{6.18}$$

which if expanded with Eq (6.12) gives us the final relation for the effective optical thickness:

$$\tau = \frac{n_i}{2dn_s}\sqrt{\frac{\log(\rho(\mathbf{H}))}{\log(W_c)}} \,. \tag{6.19}$$

For optically thin domains, we expect that most histories will be communicated, while optically thick domains will leak the fraction of histories that did not interact within. Using the optical thickness defined in Eq (6.19), we can then complete the leakage approximations by defining the bounds of $\tau \to 0, \Lambda \to 1$ and $\tau \to \infty, \Lambda \to \tau^{-1}$. With these bounds we can then define the leakage fraction out of a domain for the adjoint Neumann-Ulam method using the Wigner rational approximation:

$$\Lambda = \frac{1}{1+\tau} \,, \tag{6.20}$$

and using the mean-chord approximation:

$$\Lambda = \frac{1-e^{-\tau}}{\tau} \,. \tag{6.21}$$

Here, the leakage fraction is explicitly bound to the eigenvalues of the iteration matrix, the size of the domain, the content of the discretization stencil, and the weight cutoff selected to terminate low weight histories.

Numerical Experiments

To test the relationships developed by the spectral analysis, we form two simple numerical experiments using the diffusion model problem: one to measure the length of the random walks as a function of the iteration matrix eigenvalues, and one to measure the domain leakage fraction as a function of the iteration matrix eigenvalues and the discretization properties. Before doing this, we verify our computation of the

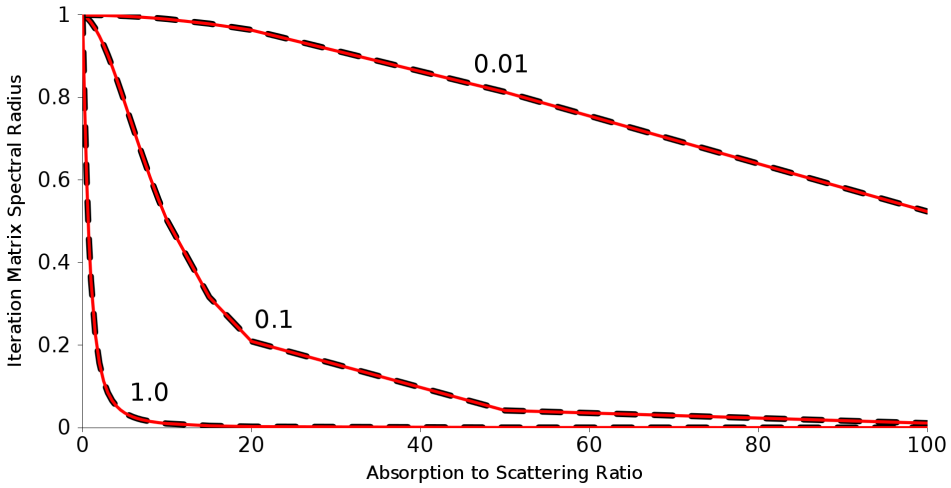

Figure 6.3: **Measured and analytic preconditioned diffusion operator spectral radius as a function of the absorption cross subsection to scattering cross subsection ratio.** *Values of* $h = 0.01$*,* $h = 0.1$*, and* $h = 1.0$ *were used. The red data was computed numerically by an eigensolver while the black dashed data was generated by Eq (6.10).*

spectral radius of the iteration matrix by numerically computing the largest eigenvalue of the diffusion operator using an iterative eigenvalue solver. For this verification, a 100×100 square grid with $h = 0.01$, $h = 0.1$, and $h = 1.0$ and the absorption cross varied from 0 to 100 while the scattering cross subsection was fixed at unity. Figure 6.3 gives the measured spectral radius of the iteration matrix and the computed spectral radius for the preconditioned diffusion operator using Eq (6.10) as function of the absorption to scattering ratio (Σ_a/Σ_s). Excellent agreement was observed between the analytic and numerical results with all data points computed within the tolerance of the iterative eigenvalue solver.

Random Walk Length

With the eigenvalue derivations verified, we can go about setting up an experiment to measure the length of the random walks generated by the adjoint Neumann-Ulam solver. To do this, we again use a 100×100 square grid with $h = 0.1$ and the absorption cross varied from 0 to 100 while the scattering cross subsection was fixed at unity. Three weight cutoff values of 1×10^{-2}, 1×10^{-4}, and 1×10^{-8} were used with 10,000 histories generated by a point source of strength 1 in the center of the domain. For each of the histories, the number of transitions made was tallied to provide an effective value of k for each history. This value was then averaged over

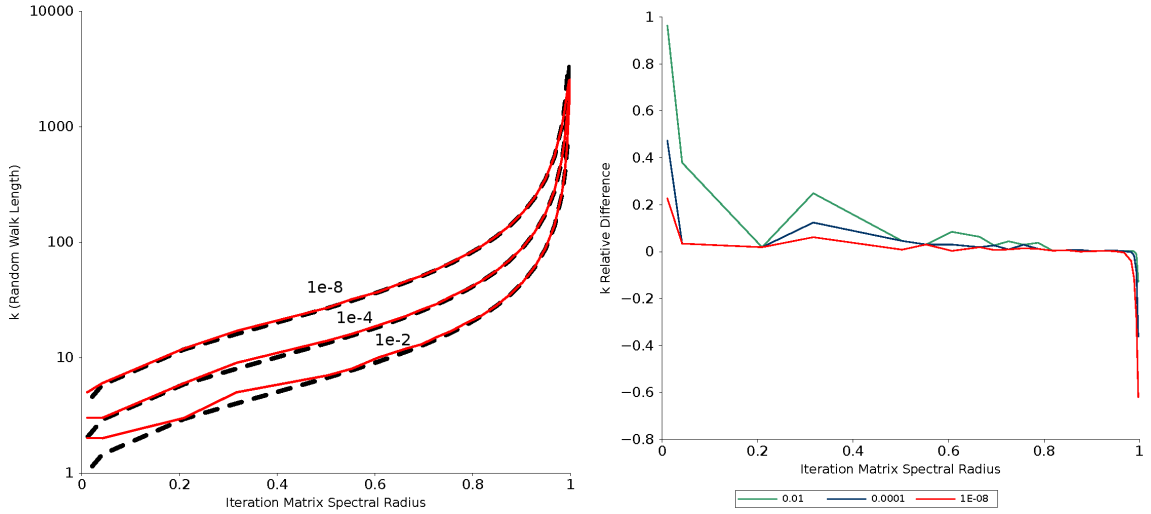

Figure 6.4: **Measured and analytic random walk length as a function of the iteration matrix spectral radius.** *The weight cutoff was varied with* 1×10^{-2}, 1×10^{-4}, *and* 1×10^{-8}. *In the left plot, the red data was computed numerically by an adjoint Neumann-Ulam implementation while the black dashed data was generated by Eq (6.12). In the right plot, the relative difference between the predicted and measured results is presented for each weight cutoff.*

all histories to get a measured value of k for the particular operator. On the left, Figure 6.4 presents these measurements as well as the analytic result computed by Eq (6.12) as a function of the iteration matrix spectral radius, $\rho(\mathbf{H})$. On the right, Figure 6.4 gives the relative difference between the predicted and observed results. We note good qualitative agreement between the measured and analytic results. However, we observe a larger relative difference for both long and short random walks.

Domain Leakage

Finally, we seek to measure the leakage from a domain in a domain decomposed Monte Carlo calculation and assess the quality of our analytic relation for the optical thickness of a domain and the associated leakage approximations. For this experiment, a square grid with $h = 0.1$ was decomposed into 9 square domains, 3 in each cardinal direction with measurements occurring in the central domain without boundary grid points. For the cross subsections, the absorption cross subsection was varied from 1 to 100 while the scattering cross subsection was set to zero to create a purely absorbing environment with weight cutoff of 1×10^{-4}. The optical thickness of these domains will vary as a function of the absorption cross subsection if the other parameters are

fixed. To compute the optical thickness, along with the spectral radius as given by Eq (6.10), we also need the parameters n_i and n_s which respectively describe the typical domain length and the average number of states moved along that typical length per history transition. For our grid above, the domains are varied in size with 50×50, 100×100, and 200×200 cells giving $n_i = 50$, $n_i = 100$, and $n_i = 200$ grid points or states along the typical length of the domain respectively. Looking at the Laplacian stencil in Eq (4.21), we see that all history transitions will only move a single state in either the i or j directions due to the symmetry of the problem. Furthermore, if we choose the i direction, not all states we will transition to will move the history in that direction. Therefore, we look to the definition of the iteration matrix in Eq (6.7) and the definition of the adjoint probability matrix in Eq (3.38) to estimate the n_s parameter. For a particular transition starting at state (i, j), 6 of the 8 possible new states in the stencil move the history in i direction with relative coefficients of 4 for moving in the $(\pm i, 0)$ direction and of 1 for moving in the $(\pm i, \pm j)$. These coefficients dictate the frequency those states are visited relative to the others. For those 6 states we can visit along the typical length, their sum is 12 out of the total 20 for the coefficients for all possible states with their ratio giving $n_s = \frac{3}{5}$.

To compute the leakage fraction numerically, 3×10^5 histories were sampled from a uniform source of strength unity over the global domain. At the start of a stage of histories, the number of histories starting in the center domain was computed and as the stage progressed, the number of histories that exited that domain was tallied with the ratio of the two numbers providing a numerical measure for the leakage fraction. Figure 6.5 gives the domain leakage measurements for the domain in the center of the global grid as well as the analytic result computed by Eqs (6.20) and (6.21) as a function of the iteration matrix spectral radius. Again, we note good qualitative agreement between the measured and analytic quantities but we begin to see the limits of the leakage approximations. To compare the quality of the two approximations, the absolute difference between the computed leakage fraction and that generated by the Wigner rational and mean chord approximations is plotted in Figure 6.6 for all domain sizes tested. From these difference results, the mean chord approximation is shown to have a lower difference for ill-conditioned systems as compared to the Wigner approximation while the Wigner approximation produces less difference for more well-conditioned systems. We also note that for the optically thick domains, the difference is likely corresponded to that observed in Figure 6.4 for the k parameter while the large relative difference in k for optically thin domains does not affect the approximation significantly. In general, the mean chord approximation

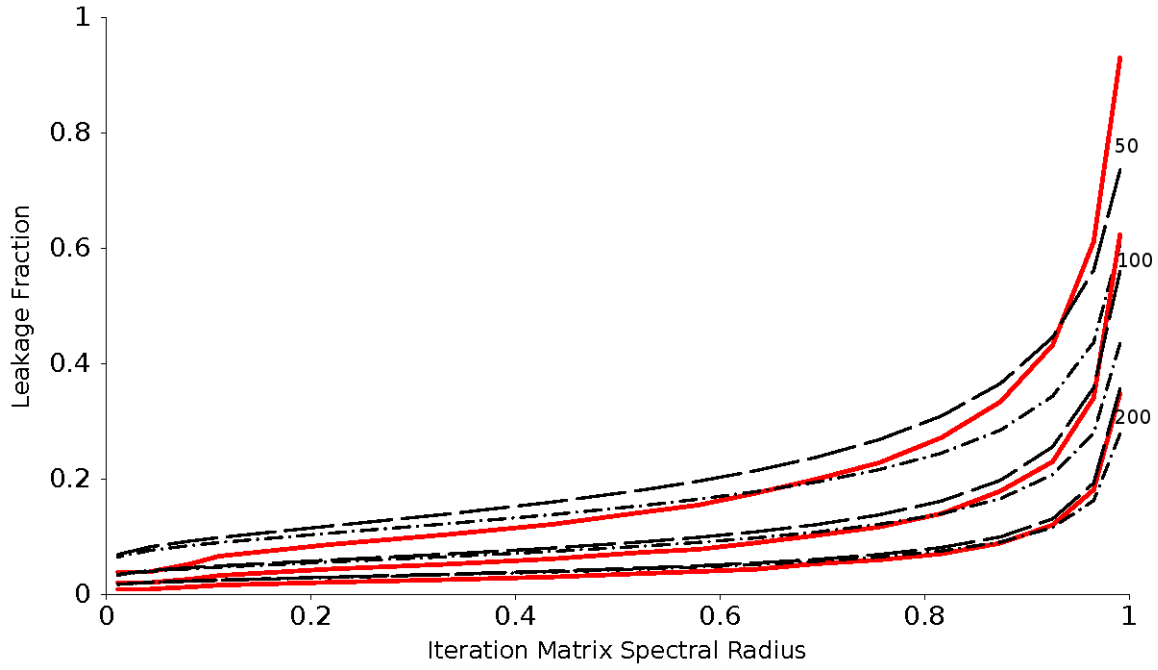

Figure 6.5: **Measured and analytic domain leakage as a function of the iteration matrix spectral radius.** *To test the behavior with respect to domain size,* $n_i = 50$*,* $n_i = 100$*,and* $n_i = 200$ *were used. The red data was computed numerically by a domain-decomposed adjoint Neumann-Ulam implementation, the black dashed data was generated by Eq (6.21) using the mean-chord approximation, and the dashed-dotted black data was generated by Eq (6.20) using the Wigner rational approximation.*

is a better choice to estimate the leakage fraction in a domain from the adjoint Neumann-Ulam method and except for a single data point with $n_i = 50$, the mean chord approximation yielded leakage fractions within 0.05 of the measured results. As the domain becomes more optically thick (with both increasing n_i and decreasing $\rho(\mathbf{H})$), the approximations are more accurate.

Reduction in Domain Leakage from Overlap

Communication Costs for Symmetric Systems

Use Seigel's work to get communication costs.

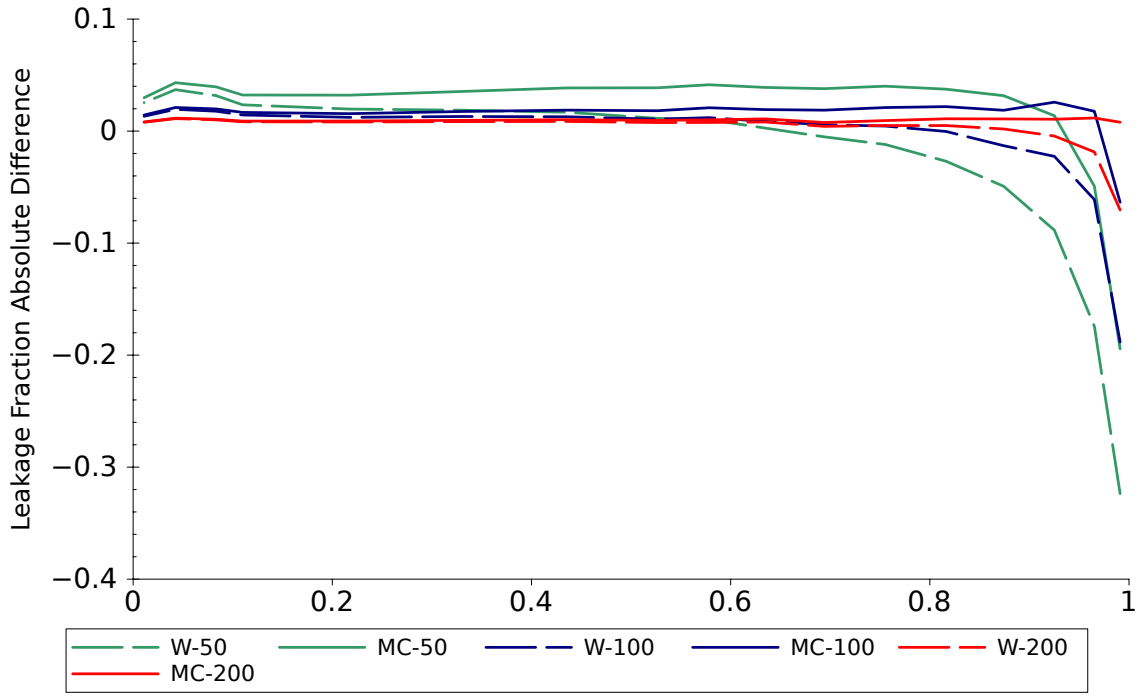

Figure 6.6: **Measured and analytic domain leakage absolute difference as a function of the iteration matrix spectral radius.** *To test the behavior with respect to domain size, $n_i = 50$ (green), $n_i = 100$ (blue), and $n_i = 200$ (red) were used. The dashed lines represent the difference using the Wigner rational approximation while the solid lines represent the difference using the mean-chord approximation.*

6.3 Fully Asynchronous MSOD Neumann-Ulam Algorithm

We can take much of what was learned from Brunner and Wagner's parallel Monte Carlo methods for radiation transport and directly apply it to a parallel formulation of our stochastic linear solvers. Direct analogs can be derived from these works by noting that the primary difference between solving a linear system with Monte Carlo methods and fixed source Monte Carlo transport problems is the content of the Markov chains that are generated. The transitions represented by these chains are bound by probabilities and weights and are initiated by the sampling of a source. In the context of transport problems, those transitions represent events such as particle scattering and absorption with probabilities that are determined by physical data in the form of cross sections. For stochastic matrix inversion, those transitions represent moving between the equations of the linear system (and therefore the physical domain which

they represent) and their probabilities are defined by the coefficients of those equations. Ultimately, we tally the contributions to generate expectation values in the desired states as we progress through the chains. Therefore, parallel methods for Monte Carlo radiation transport can be abstracted and we can use those concepts that apply to matrix inversion methods as an initial means of developing a parallel Neumann-Ulam-type solver. In this section the fully asynchronous MSOD Neumann-Ulam algorithm developed for this work is presented step-by-step in detail.

Parallel Random Number Generation

Correct parallel Monte Carlo requires parallel random number generation. To ensure distinct Markov chains for each history in the problem, regardless of its state in phase space or parallel space, a unique stream of random numbers is necessary to achieve uncorrelated tally results for the solution.

Generating the Transport Domain with Overlap and Preconditioning

Generating the Transport Source with Preconditioning

Transition Processing Kernel

Processing the transition of a single stochastic history from one state to another in the system is the fundamental kernel of the transport procedure. In this kernel, an incoming history has a state consisting of the current discrete state in the system in which it resides and its current weight and is processed as given by Algorithm **??**. The core component here is the sampling of the CDF for a particular state generated

Algorithm 6.1 Stochastic History Transition Processing Kernel

```
processTransition( history ):
i = history.state
j = sampleRowCDF( C[i] )
history.state = j
history.weight *= w[i,j]
```

by the probablities from the Neumann-Ulam decomposition. Upon exiting, the kernel has modified the state of the history with a new state and multiplied the weight with the corresponding transition weight to form the product in Eq (3.10).

Local Domain Transport Kernel

Domain-to-Domain Communication Kernel

Asynchronous Transport Kernel

Multiple Set Algorithm

Load Balancing Concerns

Although domain decomposition was shown to be efficient in a perfectly load balanced situation in Siegel's work (Siegel et al., 2012a), careful consideration must be made for situations where this is not the case. Given the stochastic nature of the problem and lack of a globally homogeneous domain, parallel Monte Carlo simulations are inherently load imbalanced. Procassini and others worked to alleviate some of the load imbalances that are generated by both particle and spatial parallelism and are therefore applicable to the MSOD algorithm (Procassini et al., 2005). They chose a dynamic balancing scheme in which the number of times a particular domain was replicated was dependent on the amount of work in that domain (i.e. domains with a high particle flux and therefore more particle histories to compute require more work). In this variation, domains that require more work will be replicated more frequently at reduced particle counts in each replication. Furthermore, Procassini and colleagues noted that as the simulation progressed and particles were transported throughout the domain, the amount of replication for each domain would vary as particle histories began to diffuse, causing some regions to have higher work loads and some to have smaller work loads than the initial conditions.

Consider the example in Figure 6.7 adapted from Procassini's work. In this example, a geometry is decomposed into 4 domains on 8 processors with a point source in the bottom left domain. To begin, because the point source is concentrated in one domain, that domain is replicated 5 times such the amount of work it has to do per processor is roughly balanced with the others. As the particles begin to diffuse away from the point source, the amount of replication is adjusted to maintain load balance. Near the end of the simulation, the diffusion of particles is enough that all domains have equal replication. By doing this, load balance is improved as each domain has approximately equal work although each domain may represent a different spatial location and have a differing number of histories to compute. Compared to Wagner's work where the fission source was distributed relatively evenly throughout the domain, fixed source problems (and especially those that have a point-like source)

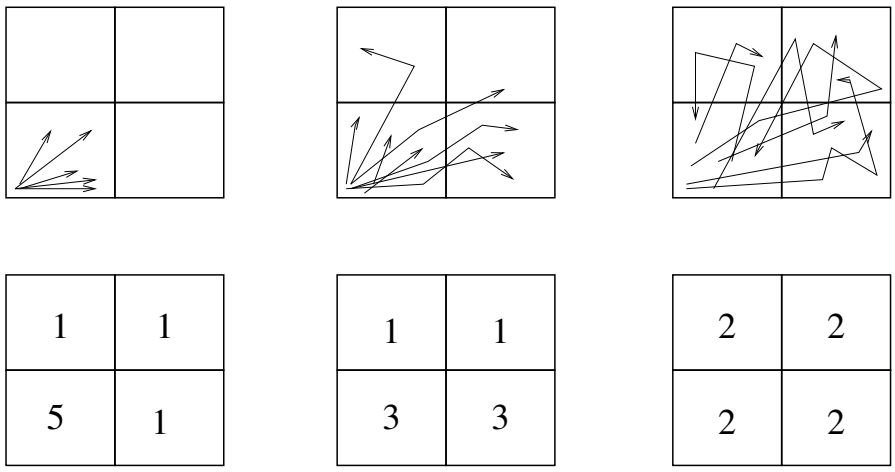

Figure 6.7: **Example illustrating how domain decomposition can create load balance issues in Monte Carlo.** *A domain is decomposed into 4 zones on 8 processors with a point source in the lower left zone. As the particles diffuse from the source in the random walk sequence as shown in the top row, their tracks populate the entire domain. As given in the bottom row, as the global percentage of particles increases in a zone, that zone's replication count is increased.*

like those presented in Procassini's work will be more prone to changing load balance requirements.

Reproducible Domain Decomposed Results

The 2006 work of Brunner is notable in that the Monte Carlo codes used to implement and test the algorithms adhered to a strict policy of generating identical results independent of domain decomposition or domain replication as derived from the work of Gentile and colleagues (Gentile et al., 2005). In Gentile's work, a procedure is given for obtaining results reproducible to machine precision for an arbitrary number of processors and domains. Differences can arise from using a different random number sequence in each domain and performing a sequence of floating point operations on identical data in a different order, leading to variations in round-off error and ultimately a non-identical answer. They use a simple example, recreated below in Figure 6.8, that illustrates these issues. In this example, the domain is decomposed on two processors with each processor owning one of the two zones. Starting with particle A, it is born in zone 1 and is transported to zone 2 where a scattering event occurs. Concerning the first reproducibility issue, if the same sequence of random numbers is not used to compute the trajectory from the new scattering event, we

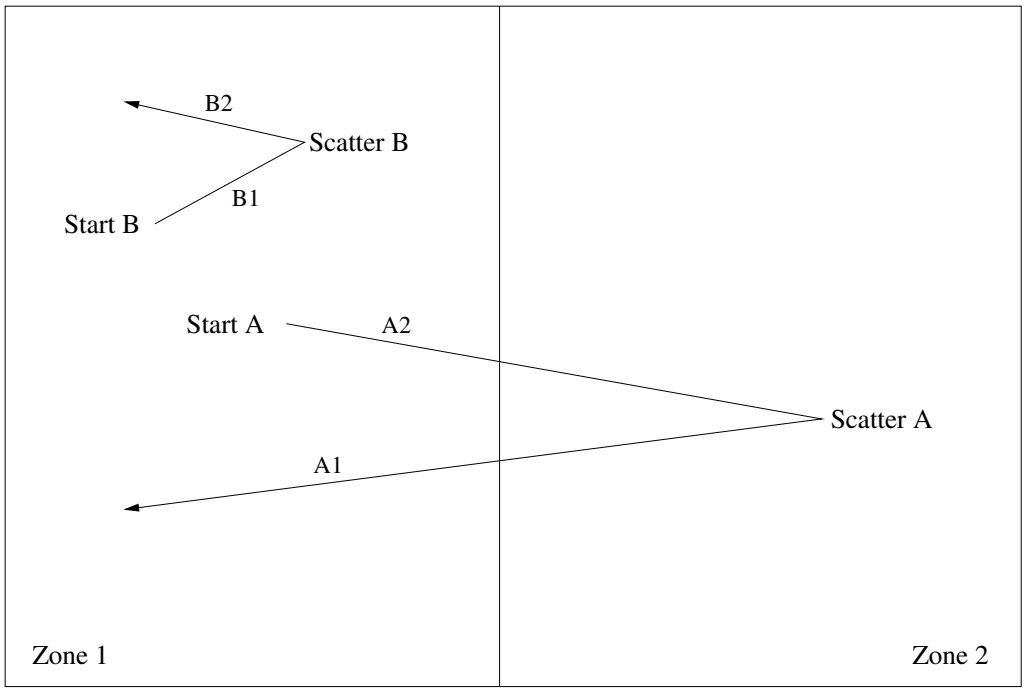

Figure 6.8: **Gentile's example illustrating how domain decomposition can create reproducibility issues in Monte Carlo.** *Both particles A and B start in zone 1 on processor 1. Particle A moves to zone 2 on processor 2 and scatters back to zone 1 while B scatters in zone 1 and remains there. A1 and A2 denote the track of particle A that is in zone 1 while B1 and B2 denote the track of particle B that is in zone 1.*

cannot expect to achieve the same result if the domain were not decomposed. If the numbers are different, the scattering event in zone 2 may keep the particle there or even eject it from the domain if the sequence were different. The second issue is demonstrated by adding another particle B that remains in the domain. In this case, an efficient algorithm will transport particle A on processor 1 until it leaves zone 1 and then transport particle B. Particle A will not renter the domain until it has been communicated to processor 2, processor 2 performs the transport, and it is communicated back to processor 1. If we are doing a track-length tally in zone 1, then we sum the tracks lengths observed in that zone. In the single processor, single zone case particle A would be transported in its entirety and then particle B transported. This would result in a tally sum with the following order of operations: $(((A1 + A2) + B1) + B2)$. If we were instead to use 2 processors, we would instead have the following order: $(((A1 + B1) + B2) + A2)$. In the context of floating point operations, we cannot expect these to have an identical result to machine precision as

round-off approximations will differ resulting in non-commutative addition.

Procassini's solutions to these problems are elegant in that they require a minimal amount of modification to be applied to the Monte Carlo algorithm. To solve the first issue, in order to ensure each particle maintains an invariant random number sequence that determines its behavior regardless of domain decomposition, each particle is assigned a random number seed that describes its current state upon entering the domain of a new processor. These seeds are derived from the actual geometric location of the particle such that it is decomposition invariant. Non-commutative floating point operations are overcome by instead mapping floating point values to 64-bit integer values for which additions will always be commutative. Once the operations are complete, these integers are mapped back to floating point values.

6.4 Parallel MCSA

With the parallel adjoint Neumann-Ulam solver implementation described aboive, the parallel implementation of the MCSA method is trivial. Recall the MCSA iteration procedure outlined in Eq (3.59). In §2.4 we discussed parallel matrix and vector operations as utilized in conventional Krylov methods. We utilize these here for the parallel MCSA implementation. In the first step, a parallel matrix-vector multiply is used to apply the split operator to the previous iterate's solution. A parallel vector update is then performed with the source vector to arrive at the initial iteration guess. In the next step, the residual is computed by the same operations where now the operator is applied to the solution guess with a parallel matrix-vector multiply and then a parallel vector update with the source vector is performed. Once the correction is computed with a parallel adjoint Neumann-Ulam solve, this correction is applied to the guess with a parallel vector update to get the new iteration solution. Additionally, as given by Eq (3.60), 2 parallel vector reductions will be required to check the stopping criteria: one initially to compute the infinity norm of the source vector, and another at every iteration to compute the infinity norm of the residual vector. For this implementation, all of the issues that will be potentially generated by the parallel adjoint solver implementation will manifest themselves here as the quality of the correction will be of intense study.

6.5 Parallel Scaling Studies

For all cases, we explore scaling for just the Monte Carlo transport kernel and MCSA as a whole iterative scheme. This will be necessary to determine if the initial relaxation step required at each iteration is a limiting factor in convergence. For each case, it will be constructive to study the scaling performance as a function of spectral radius - dictated by the cross sections and grid size to diffusion length ratio. In addition, all of these studies should hopefully have data from both the cluster and Titan and should all be compared to Aztec GMRES and CG implementations with appropriate preconditioning.

Overlapping Domain Decomposition

Weak Scaling

Strong Scaling

Multiple Set Domain Decomposition

Weak Scaling

Strong Scaling

MSOD Decomposition

Weak Scaling

Strong Scaling

Effects of Load Imbalance

Weak Scaling

Strong Scaling

6.6 Multiple Fuel Assembly SP_N Challenge Problem

Chapter 7

Conclusions and Analysis

Put conclusions here.

7.1 Monte Carlo Solutions for the SP_N Equations

7.2 Monte Carlo Solutions for Nonlinear Problems

7.3 Parallel Monte Carlo Solutions

7.4 Future Work

7.5 Closing Remarks

Appendix A

Derivation of the P_N Equations

Here we derive the P_N equations, a simplified form of the general transport equation where Legendre polynomials are used to expand the angular flux and scattering cross section variables as a means of capturing the angular structure of the solution. Before deriving this form of the transport equation, we briefly discuss a few properties of Legendre polynomials that we will find useful in the derivation.

A.1 Legendre Polynomials

The Legendre polynomials are an orthogonal set of functions that are solutions to Legendre's differential equation. They have the following form (Lewis, 1993):

$$P_l(\mu) = \frac{1}{2^l l!}\frac{d^l}{d\mu^l}(\mu^2 - 1)^l \,. \tag{A.1}$$

These functions have several useful properties including orthogonality:

$$\int_{-1}^{1} P_l(\mu)P_{l'}(\mu)d\mu = \frac{1}{2l+1}\delta_{ll'} \,, \tag{A.2}$$

a recurrence relation:

$$\mu P_l(\mu) = \frac{1}{2l+1}[(l+1)P_{l+1}(\mu) + lP_{l-1}(\mu)] \,, \tag{A.3}$$

and an addition theorem:

$$P_l(\hat{\Omega}\cdot\hat{\Omega}') = \frac{1}{2n+1}\sum_{m=-l}^{l} Y_{lm}(\hat{\Omega})Y_{lm}^*(\hat{\Omega}') \,, \tag{A.4}$$

where the functions $Y_{lm}(\hat{\Omega})$ are the spherical harmonics. We can form the addition theorem in this way because the spherical harmonics are in fact just harmonic multiples of the Legendre polynomials:

$$Y_{lm}(\hat{\Omega}) = \sqrt{\frac{(2l+1)(l-m)!}{(l+m)!}}P_l^m(\mu)e^{i\omega m} \,, \tag{A.5}$$

where ω is the azimuthal component of the streaming direction. We can reduce Eq (A.4) for the planar geometry we are studying by ignoring the azimuthal components of the addition theorem. As shown in Eq (A.5), the azimuthal dependence is given by the harmonic component, $e^{i\omega m}$, and therefore we choose to ignore all terms in Eq. (A.4) where $m \neq 0$. This gives:

$$P_l(\hat{\Omega} \cdot \hat{\Omega}') = \frac{1}{2n+1} Y_{l0}(\hat{\Omega}) Y_{l0}^*(\hat{\Omega}') \,. \tag{A.6}$$

Per Eq (A.5) we have:

$$Y_{l0}(\hat{\Omega}) = \sqrt{2l+1} P_l^0(\mu) \,, \tag{A.7}$$

where $P_l^0(\mu) = P_l(\mu)$ is the 0^{th} associated Legendre function. Finally, we can reduce the addition theorem from Eq (A.6) with Eq (A.7) to a simple product for planar geometry:

$$P_l(\hat{\Omega} \cdot \hat{\Omega}') = P_l(\mu) P_l(\mu') \,. \tag{A.8}$$

A.2 Planar P_N Equations

With the Legendre polynomial properties defined above, we can proceed by deriving the P_N equations for planar geometry. For these equations, we will start by assuming a monoenergtic field of neutrons such that we are solving the following reduced form of the transport equation:

$$\mu \frac{\partial}{\partial x} \psi(x,\mu) + \sigma(x)\psi(x,\mu) = \int \sigma_s(x, \hat{\Omega}' \rightarrow \hat{\Omega}) \psi(\vec{r}, \hat{\Omega}') d\Omega' + \frac{q(x)}{4\pi} \,. \tag{A.9}$$

The P_N equations introduce the approximation that the angular dependence of the scattering cross section, σ_s, and the angular flux ψ, can be discretized by expanding them in Legendre polynomials as follows:

$$\psi(x,\mu) = \sum_{n=0}^{\infty} (2n+1) P_n(\mu) \phi_n(x) \,, \tag{A.10}$$

$$\sigma_{sm}(x) = \sum_{m=0}^{\infty} (2m+1) P_m(\mu) \sigma_s(x) \,, \tag{A.11}$$

where we have supressed the 2π generated by integrating away the azimuthal angular component and $\phi_n(x)$ in Eq (A.10) is referred to as the n^{th} Legendre moment of the

neutron flux and is given by:

$$\phi_n(x) = \int_{-1}^{1} P_n(\mu)\psi(x,\mu)d\mu \,. \tag{A.12}$$

We first insert the expansions given by Eq (A.10) and (A.11) into the planar transport equation given by Eq (A.9):

$$\frac{\partial}{\partial x}\Big[\sum_{n=0}^{\infty}(2n+1)\phi_n\mu P_n(\mu)\Big] + \sigma\sum_{n=0}^{\infty}(2n+1)\phi_n P_n(\mu) = \int_{-1}^{1}\sum_{m=0}^{\infty}(2m+1)\sigma_{sm}P_m(\mu_0)\sum_{n=0}^{\infty}(2n+1)\phi_n P_n(\mu')d\mu' + q \,, \tag{A.13}$$

where the dependence on the spatial variable, x, has been supressed. To arrive at the P_N equations we multiply Eq (A.13) by $P_m(\mu)$ and integrate over the angular domain $\int_{-1}^{1} d\mu$. We will look at each term in Eq (A.13) individually.

Streaming Term We first apply the multiplication and integration as prescribed above:

$$\frac{\partial}{\partial x}\left[\sum_{n=0}^{\infty}(2n+1)\mu\phi_n P_n(\mu)\right] \rightarrow \int_{-1}^{1}\frac{\partial}{\partial x}\left[\sum_{n=0}^{\infty}(2n+1)\phi_n\mu P_n(\mu)P_m(\mu)\right]d\mu \,. \tag{A.14}$$

The $\mu P_n(\mu)$ term can be eliminated via the recurrence relation given by Eq (A.3):

$$\int_{-1}^{1}\frac{\partial}{\partial x}\left[\sum_{n=0}^{\infty}(2n+1)\frac{\phi_n}{2n+1}[(n+1)P_{n+1}(\mu)+nP_{n-1}(\mu)]P_m(\mu)\right]d\mu \,, \tag{A.15}$$

which expands to:

$$\sum_{n=0}^{\infty}\frac{\partial}{\partial x}\phi_n\left[(n+1)\int_{-1}^{1}P_{n+1}(\mu)P_m(\mu)d\mu + n\int_{-1}^{1}P_{n-1}(\mu)P_m(\mu)d\mu\right] . \tag{A.16}$$

This reveals the orthogonality relation given by Eq (A.2) that when inserted into Eq (A.15):

$$\sum_{n=0}^{\infty}\frac{\partial}{\partial x}\phi_n\left[(n+1)\frac{1}{2n+1}\delta_{n,n+1} + n\frac{1}{2n+1}\delta_{n,n-1}\right] . \tag{A.17}$$

We can then distribute the Legendre moment to arrive at the final form of the streaming term:

$$\sum_{n=0}^{\infty}\frac{\partial}{\partial x}\frac{1}{2n+1}\Big[(n+1)\phi_{n+1} + n\phi_{n-1}\Big] . \tag{A.18}$$

Collision Term To reduce the collision term, the orthogonality relation is again applied after the integration:

$$\sigma \sum_{n=0}^{\infty}(2n+1)\phi_n P_n(\mu) \rightarrow \int_{-1}^{1} \sigma \sum_{n=0}^{\infty}(2n+1)\phi_n P_n(\mu)P_m(\mu)d\mu \,, \tag{A.19}$$

$$\sigma \sum_{n=0}^{\infty}(2n+1)\phi_n \frac{1}{2n+1} \,, \tag{A.20}$$

giving for the final collision term:

$$\sum_{n=0}^{\infty}\sigma\phi_n \,. \tag{A.21}$$

Scattering Term For the scattering term:

$$\int_{-1}^{1}\sum_{m=0}^{\infty}(2m+1)\sigma_{sm}P_m(\mu_0)\sum_{n=0}^{\infty}(2n+1)\phi_n P_n(\mu')d\mu' \rightarrow \int_{-1}^{1}\int_{-1}^{1}\sum_{m=0}^{\infty}(2m+1)\sigma_{sm}P_m(\mu)P_m(\mu_0)\sum_{n=0}^{\infty}(2n+1)\phi_n P_n(\mu')d\mu' d\mu \,. \tag{A.22}$$

The addition thereom from Eq (A.8) is applied to give:

$$\int_{-1}^{1}\int_{-1}^{1}\sum_{m=0}^{\infty}(2m+1)\sigma_{sm}P_m(\mu)P_m(\mu)P_m(\mu')\sum_{n=0}^{\infty}(2n+1)\phi_n P_n(\mu')d\mu' d\mu \,, \tag{A.23}$$

which can be rearranged as:

$$\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}(2m+1)\sigma_{sm}(2n+1)\phi_n \int_{-1}^{1} P_m(\mu')P_n(\mu')d\mu' \int_{-1}^{1} P_m(\mu)P_m(\mu)d\mu \,. \tag{A.24}$$

Again we can apply orthogonality to eliminate the Legendre polynomials:

$$\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}(2m+1)\sigma_{sm}(2n+1)\phi_n \frac{1}{2n+1}\delta_{nm}\frac{1}{2m+1}\delta_{mm} \,, \tag{A.25}$$

which is reduced to:

$$\sum_{n=0}^{\infty}\sigma_{sn}\phi_n \,, \tag{A.26}$$

with the dependence on the index m eliminated.

Source Term The last term we are concerned with in Eq (A.13) is the source term:

$$q \to \int_{-1}^{1} qP_n(\mu)d\mu \,. \tag{A.27}$$

We can leverage orthogonality by multiplying by $P_0(\mu) = 1$:

$$\int_{-1}^{1} qP_0P_n(\mu)d\mu = \frac{q}{2*0+1}\delta_{n0} \,, \tag{A.28}$$

giving a final source term of:

$$q\delta_{n0} \,. \tag{A.29}$$

Now that we have expanded all angular dependent terms in Eq (A.13) and reduced them appropriately, we can combine them to generate the P_N equations:

$$\sum_{n=0}^{\infty} \frac{\partial}{\partial x}\frac{1}{2n+1}\Big[(n+1)\phi_{n+1} + n\phi_{n-1}\Big] + \sum_{n=0}^{\infty}\sigma\phi_n = \sum_{n=0}^{\infty}\sigma_{sn}\phi_n + q\delta_{n0} \,. \tag{A.30}$$

More formally, the P_N equations are written as:

$$\frac{1}{2n+1}\frac{\partial}{\partial x}\Big[(n+1)\phi_{n+1} + n\phi_{n-1}\Big] + \Sigma_n\phi_n = q\delta_{n0} \,, \tag{A.31}$$

where $\Sigma_n = \sigma - \sigma_{sn}$ and the summations are truncated at some level of approximation N such that $n = 0, 1, \ldots, N$. This yields a set of $N+1$ equations for $N+2$ flux moments. We therefore require an additional equation to close the system. In accordance with the series truncation as an approximation we choose the last moment in the expansion to be zero:

$$\phi_{N+1} = 0 \,. \tag{A.32}$$

As an example, we will construct the P5 equations from Eq (A.31) and the closure

given by Eq (A.32):

$$\frac{\partial}{\partial x}\phi_1 + \Sigma_0\phi_0 = q\,, \tag{A.33a}$$

$$\frac{1}{3}\frac{\partial}{\partial x}\Big[2\phi_2 + \phi_0\Big] + \Sigma_1\phi_1 = 0\,, \tag{A.33b}$$

$$\frac{1}{5}\frac{\partial}{\partial x}\Big[3\phi_3 + 2\phi_1\Big] + \Sigma_2\phi_2 = 0\,, \tag{A.33c}$$

$$\frac{1}{7}\frac{\partial}{\partial x}\Big[4\phi_4 + 3\phi_2\Big] + \Sigma_3\phi_3 = 0\,, \tag{A.33d}$$

$$\frac{1}{9}\frac{\partial}{\partial x}\Big[5\phi_5 + 4\phi_3\Big] + \Sigma_4\phi_4 = 0\,, \tag{A.33e}$$

$$\frac{1}{11}\frac{\partial}{\partial x}5\phi_4 + \Sigma_5\phi_5 = 0\,. \tag{A.33f}$$

This gives us a set of 6 coupled equations for the six Legendre moments requested defined over the entire spatial domain for a single energy group. In practice, only odd-numbered P_N orders are generally used (Lewis, 1993). This is due to the fact that using odd N yields an even number of $N+1$ equations which can be split evenly on the left and right boundaries of the problem to facilitate the description of boundary conditions. We will choose this convention when deriving the boundary conditions.

A.3 Boundary Conditions for the P_N Equations

Per the analysis in (Lewis, 1993), two types of boundary conditions will be discussed: reflecting and Marshak. Marshak boundary conditions can be used to specify both vacuum conditions and isotropic source conditions on the boundary.

Reflecting Boundary Conditions In this case, the incoming flux should be equivalent to the outgoing flux at the boundary point x_b:

$$\psi(x_b, \mu) = \psi(x_b, -\mu)\,. \tag{A.34}$$

Given the legendre expansion for the flux defined in Eq (A.10) and the Legendre polynomial property that $P_n(\mu) = (-1)^n P_n(-\mu)$, the condition specified by Eq (A.34) can be satisfied if

$$\phi_n = 0,\ \forall \text{ odd } n\,, \tag{A.35}$$

as all even n yield $P_n(\mu) = P_n(-\mu)$ and therefore an equivalent reflecting condition for the flux moments.

Marshak Boundary Conditions The Marshak conditions come directly from the Legendre moments of the flux:

$$\int_{\mu_b} P_i(\mu)\psi(\mu)d\mu = \int_{\mu_b} P_i(\mu)\psi_b(\mu)d\mu \qquad \text{for } i = 1, 3, ..., N\,, \tag{A.36}$$

where $\psi_b(\mu)$ is the prescribed angular flux on the boundary of interest and μ_b the angular domain defined by the boundary. To discretize this condition, we again insert the angular flux expansions from Eq (A.10) for the fluxes defined in the domain:

$$\int_{\mu_b} P_i(\mu) \sum_{n=0}^{N} (2n+1)\phi_n P_n(\mu)d\mu = \int_{\mu_b} P_i(\mu)\psi_b(\mu)d\mu\,. \tag{A.37}$$

The boundary flux, $\phi_b(\mu)$ is assumed to known and therefore Eq (A.37) defines a set of $(N+1)/2$ equations to be solved on each boundary in the planar case, closing the system in the spatial domain.

As an example of applying the Marshak conditions, consider the P_3 case with an isotropic boundary source ϕ_b on the left side of the domain. In this case, the angular domain over which the boundary flux is defined will be $\mu_b \in [0, 1]$, giving the bounds of integration. We first expand the summation for $i = 1$:

$$\int_0^1 \mu \left[\phi_0 + 3\phi_1\mu + \frac{5}{2}\phi_2(3\mu^2 - 1) + \frac{7}{2}\phi_3(5\mu^3 - 3\mu)\right] d\mu = \int_0^1 \mu\phi_b d\mu\,, \tag{A.38}$$

and then $i = 3$:

$$\int_0^1 \frac{1}{2}(5\mu^3 - 3\mu)\left[\phi_0 + 3\phi_1\mu + \frac{5}{2}\phi_2(3\mu^2 - 1) + \frac{7}{2}\phi_3(5\mu^3 - 3\mu)\right] d\mu = \int_0^1 \frac{1}{2}(5\mu^3 - 3\mu)\phi_b d\mu\,. \tag{A.39}$$

Expanding the polynomials in μ and carrying out the simple integration then gives 2 equations for the left hand boundary:

$$\phi_0 + 2\phi_1 + \frac{5}{4}\phi_2 = \phi_b\,, \tag{A.40}$$

$$\phi_0 - 5\phi_2 - 8\phi_3 = \phi_b\,. \tag{A.41}$$

The right hand side boundary condition will yield 2 complementary equations if Marshak conditions are used or 2 non-zero moments to be solved for if reflected conditions are used. The formulation above also holds for vacuum conditions where $\phi_b = 0$.

A.4 Eigenvalue Form of the P_N Equations

In addition to fixed source problems, the P_N equations may be modified to solve eigenvalue problems by applying the same discretization techniques to the fission source in Eq (4.3). Considering the multigroup case, we first replace the integration over energy with a summation over energy groups:

$$\frac{1}{k}\chi(E)\iint \nu\sigma_f(x,E')\psi(x,\hat{\Omega}',E')d\Omega' dE' = \frac{1}{k}\sum_0^G \chi^g \int \nu\sigma_f^{g'}(x)\psi^{g'}(x,\hat{\Omega}')d\Omega' \,, \quad \text{(A.42)}$$

where $G = N_g - 1$ and N_g is the number of energy groups. Next, we expand the angular flux in the fission term by applying the expansion in Eq (A.10) and remove the dependence on the azimuthal angle:

$$\frac{1}{k}\sum_0^G \chi^g \int \nu\sigma_f^{g'}(x)\psi^{g'}(x,\hat{\Omega}')d\Omega' = \frac{1}{k}\sum_0^G \chi^g \int_{-1}^1 \nu\sigma_f^{g'}(x)\sum_{n=0}^{\infty}(2n+1)P_n(\mu)\phi_n^{g'}(x)d\mu' \,. \quad \text{(A.43)}$$

We can again use the the orthogonality property of the Legendre polynomials by multplying by $P_0(\mu) = 1$:

$$\frac{1}{k}\sum_0^G \chi^g \int_{-1}^1 \nu\sigma_f^{g'}(x)\sum_{n=0}^{\infty}(2n+1)P_n(\mu)P_0(\mu)\phi_n^{g'}(x)d\mu' \,, \quad \text{(A.44)}$$

giving:

$$\frac{1}{k}\sum_0^G \chi^g \nu\sigma_f^{g'}(x)\phi_n^{g'}(x)\delta_{n0} \,, \quad \text{(A.45)}$$

where χ^g and $\sigma_f^{g'}\phi^{g'}$ means that an effective inner product amongst group terms will be computed with the discrete χ spectrum and the fission reaction rate. In a multigoup form, if we choose to represent the energy dependent moments as vectors, $\mathbf{\Phi_n}$, composed of all energy groups, then this inner product takes the form of a spatially

dependent fission matrix $\mathbf{F}$:

$$\mathbf{F} = \nu \begin{bmatrix} (\chi^0\sigma_f^0) & (\chi^0\sigma_f^1) & \cdots & (\chi^0\sigma_f^G) \\ (\chi^1\sigma_f^0) & (\chi^1\sigma_f^1) & \cdots & (\chi^1\sigma_f^G) \\ \vdots & \vdots & \ddots & \vdots \\ (\chi^G\sigma_f^0) & (\chi^G\sigma_f^1) & \cdots & (\chi^G\sigma_f^G) \end{bmatrix}. \tag{A.46}$$

This gives a final multigroup P_N fission source of:

$$\frac{1}{k}\mathbf{F}\mathbf{\Phi}_\mathbf{n}\delta_{n0} \tag{A.47}$$

where the spatial dependence of the flux moments and fission matrix has been supressed.

Appendix B

Derivation of the Multigroup SP_N Equations

In the text, we formulated the SP_N equations for a single neutron energy. To expand these equations for multiple energies, we start by stating the multigroup neutron transport equation for a single dimension in planar geometry:

$$\mu\frac{\partial}{\partial x}\psi^g(x,\mu)+\sigma^g(x)\psi^g(x,\mu)= \sum_{g'=0}^{G}\int\sigma_s^{gg'}(x,\hat{\Omega}'\rightarrow\hat{\Omega})\psi^{g'}(x,\hat{\Omega}')d\Omega'+\frac{q^g(x)}{4\pi}\,, \quad \text{(B.1)}$$

where g denotes the group index of 0 to G groups, $G=N_g-1$, and the integration of the scattering emission term over energy has been replaced by a discrete summation. For scattering, $\sigma_s^{gg'}$ provides the probability of scattering at a particular angle from group g to g'. The result is an equation nearly identical in form to Eq (4.2) where now instead of forming the SP_N equations for a single energy group, we form them for each of the energy groups with group coupling occurring through the scattering term. The multigroup P_N equations are then:

$$\frac{1}{2n+1}\frac{\partial}{\partial x}\Big[(n+1)\phi_{n+1}^g+n\phi_{n-1}^g\Big]+\sum_{g'}(\sigma^g\delta_{gg'}-\sigma_{sn}^{gg'})\phi_n^g=q\delta_{n0}\,, \quad \text{(B.2)}$$

for $n=0,1,\ldots,N$ where the flux and scattering moments are defined in a group. We observe that a $N_g\times N_g$ scattering matrix is formed:

$$\mathbf{\Sigma}_n=\sum_{g'}(\sigma^g\delta_{gg'}-\sigma_{sn}^{gg'})\,, \quad \text{(B.3)}$$

and when expanded gives:

$$\mathbf{\Sigma}_n = \begin{bmatrix} (\sigma^0 - \sigma_{sn}^{00}) & -\sigma_{sn}^{01} & \dots & -\sigma_{sn}^{0G} \\ -\sigma_{sn}^{10} & (\sigma^1 - \sigma_{sn}^{11}) & \dots & -\sigma_{sn}^{1G} \\ \vdots & \vdots & \ddots & \vdots \\ -\sigma_{sn}^{G0} & -\sigma_{sn}^{G1} & \dots & (\sigma^G - \sigma_{sn}^{GG}) \end{bmatrix} . \tag{B.4}$$

It is also useful to combine the group flux moments and sources into a single vector for more compact notation:

$$\mathbf{\Phi_n} = (\phi_n^0 \ \phi_n^1 \ \cdots \phi_n^G)^T , \tag{B.5}$$

$$\mathbf{q} = (q^0 \ q^1 \ \cdots q^G)^T . \tag{B.6}$$

Next, we apply the SP_N approximation to Eq (B.2) in identical fashion to the monoenergetic case. This gives:

$$\begin{aligned} -\nabla \cdot \Bigg[& \frac{n}{2n+1} \mathbf{\Sigma_{n-1}}^{-1} \nabla \Big(\frac{n-1}{2n-1} \mathbf{\Phi_{n-2}} + \frac{n}{2n-1} \mathbf{\Phi_n} \Big) \\ & + \frac{n+1}{2n+1} \mathbf{\Sigma_{n+1}}^{-1} \nabla \Big(\frac{n+1}{2n+3} \mathbf{\Phi_n} + \frac{n+2}{2n+3} \mathbf{\Phi_{n+2}} \Big) \Bigg] \\ & + \mathbf{\Sigma_n \Phi_n} = \mathbf{q} \delta_{n0} \qquad n = 0, 2, 4, \cdots, N . \end{aligned} \tag{B.7}$$

This adds more complexity than the monoenergetic formulation in that all unknowns in this group of equations are now vector quantities and scattering relationships are contained in matrices rather than a scalar quantity. Because of this, the same sequence of variable changes and algebra can be used to build a set of matrix equations, this time in a block formulation:

$$-\nabla \cdot \mathbb{D}_n \nabla \mathbb{U}_n + \sum_{m=1}^{4} \mathbb{A}_{nm} \mathbb{U}_m = \mathbb{Q}_n , \tag{B.8}$$

where the definition of all quantities are the same with internal scalar values replaced by the group-vector values. In addition, $\mathbb{A}$ is now a block matrix of $N_g \times N_g$ sub-matrices

generated from the moment scattering matrices:

$$\mathbf{A} = \begin{bmatrix} (\boldsymbol{\Sigma}_0) & (-\frac{2}{3}\boldsymbol{\Sigma}_0) & (\frac{8}{15}\boldsymbol{\Sigma}_0) & (-\frac{16}{35}\boldsymbol{\Sigma}_0) \\ (-\frac{2}{3}\boldsymbol{\Sigma}_0) & (\frac{4}{9}\boldsymbol{\Sigma}_0 + \frac{5}{9}\boldsymbol{\Sigma}_2) & (-\frac{16}{45}\boldsymbol{\Sigma}_0 - \frac{4}{9}\boldsymbol{\Sigma}_2) & (\frac{32}{105}\boldsymbol{\Sigma}_0 + \frac{8}{21}\boldsymbol{\Sigma}_2) \\ (\frac{8}{15}\boldsymbol{\Sigma}_0) & (-\frac{16}{45}\boldsymbol{\Sigma}_0 - \frac{4}{9}\boldsymbol{\Sigma}_2) & (\frac{64}{225}\boldsymbol{\Sigma}_0 + \frac{16}{45}\boldsymbol{\Sigma}_2 + \frac{9}{25}\boldsymbol{\Sigma}_4) & (-\frac{128}{525}\boldsymbol{\Sigma}_0 - \frac{32}{105}\boldsymbol{\Sigma}_2 - \frac{54}{175}\boldsymbol{\Sigma}_4) \\ (-\frac{16}{35}\boldsymbol{\Sigma}_0) & (\frac{32}{105}\boldsymbol{\Sigma}_0 + \frac{8}{21}\boldsymbol{\Sigma}_2) & (-\frac{128}{525}\boldsymbol{\Sigma}_0 - \frac{32}{105}\boldsymbol{\Sigma}_2 - \frac{54}{175}\boldsymbol{\Sigma}_4) & (\frac{256}{1225}\boldsymbol{\Sigma}_0 + \frac{64}{245}\boldsymbol{\Sigma}_2 + \frac{324}{1225}\boldsymbol{\Sigma}_4 + \frac{13}{49}\boldsymbol{\Sigma}_6) \end{bmatrix}. \tag{B.9}$$

Boundary conditions for these equations are provided in Appendix C.

Appendix C

Boundary Conditions for the SP_N Equations

We approach the formulation of the boundary conditions in much the same way as for the P_N equations with both reflecting and Marshak conditions possible. To begin, we perform the expansion in Eq (A.37) for the left side of the planar system, this time for $N = 7$ to correspond to our SP_7 system. For $i = 1$:

$$\int_0^1 \mu \Bigg[\phi_0 + 3\phi_1\mu + \frac{5}{2}\phi_2(3\mu^2 - 1) + \frac{7}{2}\phi_3(5\mu^3 - 3\mu) + \frac{1}{8}(63\mu^5 - 70\mu^3 + 15\mu) \Bigg] d\mu = \int_0^1 \mu\phi_b d\mu \,, \quad \text{(C.1)}$$

for $i = 3$,

$$\int_0^1 \frac{1}{2}(5\mu^3 - 3\mu) \Bigg[\phi_0 + 3\phi_1\mu + \frac{5}{2}\phi_2(3\mu^2 - 1) + \frac{7}{2}\phi_3(5\mu^3 - 3\mu) + \frac{1}{8}(63\mu^5 - 70\mu^3 + 15\mu) \Bigg] d\mu = \int_0^1 \frac{1}{2}(5\mu^3 - 3\mu)\phi_b d\mu \,, \quad \text{(C.2)}$$

for $i = 5$:

$$\int_0^1 \frac{1}{8}(63\mu^5 - 70\mu^3 + 15\mu) \Bigg[\phi_0 + 3\phi_1\mu + \frac{5}{2}\phi_2(3\mu^2 - 1) + \frac{7}{2}\phi_3(5\mu^3 - 3\mu) + \frac{1}{8}(63\mu^5 - 70\mu^3 + 15\mu) \Bigg] d\mu = \int_0^1 \frac{1}{8}(63\mu^5 - 70\mu^3 + 15\mu)\phi_b d\mu \,, \quad \text{(C.3)}$$

and for $i = 7$:

$$\int_0^1 \frac{1}{16}(429\mu^7 - 693\mu^5 + 315\mu^3 + 35\mu)\Bigg[\phi_0 + 3\phi_1\mu + \frac{5}{2}\phi_2(3\mu^2 - 1) + \frac{7}{2}\phi_3(5\mu^3 - 3\mu) + \frac{1}{8}(63\mu^5 - 70\mu^3 + 15\mu) + \frac{1}{16}(429\mu^7 - 693\mu^5 + 315\mu^3 + 35\mu)\Bigg] d\mu = \int_0^1 \frac{1}{16}(429\mu^7 - 693\mu^5 + 315\mu^3 + 35\mu)\phi_b d\mu \,. \quad \text{(C.4)}$$

Carrying out the simple integrations yields the following system of equations:

$$\frac{1}{2}\phi_0 + \phi_1 + \frac{5}{8}\phi_2 - \frac{3}{16}\phi_4 + \frac{13}{128}\phi_6 = \frac{1}{2}\phi_b \quad \text{(C.5a)}$$

$$-\frac{1}{8}\phi_0 + \frac{5}{8}\phi_2 + \phi_3 + \frac{81}{128}\phi_4 - \frac{13}{64}\phi_6 = -\frac{1}{8}\phi_b \quad \text{(C.5b)}$$

$$\frac{1}{16}\phi_0 - \frac{25}{128}\phi_2 + \frac{81}{128}\phi_4 + \phi_5 + \frac{325}{512}\phi_6 = \frac{1}{16}\phi_b \quad \text{(C.5c)}$$

$$-\frac{5}{128}\phi_0 + \frac{7}{64}\phi_2 - \frac{105}{512}\phi_4 + \frac{325}{512}\phi_6 + \phi_7 = -\frac{5}{128}\phi_b \,, \quad \text{(C.5d)}$$

where ϕ_b is again an isotropic source prescribed on the planar boundary. For the odd moment in each boundary equation, we insert Eq (4.5) to remove them, leaving only the even moments and a set of first-order differential equations:

$$\frac{1}{2}\phi_0 + \frac{1}{3\Sigma_1}\frac{\partial}{\partial x}(\phi_0 + 2\phi_2) + \frac{5}{8}\phi_2 - \frac{3}{16}\phi_4 + \frac{13}{128}\phi_6 = \frac{1}{2}\phi_b \quad \text{(C.6a)}$$

$$-\frac{1}{8}\phi_0 + \frac{5}{8}\phi_2 + \frac{1}{7\Sigma_3}\frac{\partial}{\partial x}(3\phi_2 + 4\phi_4) + \frac{81}{128}\phi_4 - \frac{13}{64}\phi_6 = -\frac{1}{8}\phi_b \quad \text{(C.6b)}$$

$$\frac{1}{16}\phi_0 - \frac{25}{128}\phi_2 + \frac{81}{128}\phi_4 + \frac{1}{11\Sigma_5}\frac{\partial}{\partial x}(5\phi_4 + 6\phi_6) + \frac{325}{512}\phi_6 = \frac{1}{16}\phi_b \quad \text{(C.6c)}$$

$$-\frac{5}{128}\phi_0 + \frac{7}{64}\phi_2 - \frac{105}{512}\phi_4 + \frac{325}{512}\phi_6 + \frac{1}{15\Sigma_7}\frac{\partial}{\partial x}(7\phi_6) = -\frac{5}{128}\phi_b \,. \quad \text{(C.6d)}$$

We can again make the substitution of variables given by Eqs (4.9) and (4.10) for consistency with the equations defined on the domain. In addition, we apply the SP_N approximation to the derivatives by assuming they are instead multidimensional

gradients on the boundary such that $\frac{\partial}{\partial x} \rightarrow \hat{\mathbf{n}} \cdot \nabla$. Doing this gives:

$$\frac{1}{2}\Big(u_1 - \frac{2}{3}u_2 + \frac{8}{15}u_3 - \frac{16}{35}u_4\Big) + \frac{1}{3\Sigma_1}\hat{\mathbf{n}} \cdot \nabla\Big(\Big(u_1 - \frac{2}{3}u_2 + \frac{8}{15}u_3 - \frac{16}{35}u_4\Big) + 2\Big(\frac{1}{3}u_2 - \frac{4}{15}u_3 + \frac{8}{35}u_4\Big)\Big) + \frac{5}{8}\Big(\frac{1}{3}u_2 - \frac{4}{15}u_3 + \frac{8}{35}u_4\Big) - \frac{3}{16}\Big(\frac{1}{5}u_3 - \frac{6}{35}u_4\Big) + \frac{13}{128}\Big(\frac{1}{7}u_4\Big) = \frac{1}{2}\phi_b\,, \quad \text{(C.7)}$$

$$-\frac{1}{8}\Big(u_1 - \frac{2}{3}u_2 + \frac{8}{15}u_3 - \frac{16}{35}u_4\Big) + \frac{5}{8}\Big(\frac{1}{3}u_2 - \frac{4}{15}u_3 + \frac{8}{35}u_4\Big) + \frac{1}{7\Sigma_3}\hat{\mathbf{n}} \cdot \nabla\Big(3\Big(\frac{1}{3}u_2 - \frac{4}{15}u_3 + \frac{8}{35}u_4\Big) + 4\Big(\frac{1}{5}u_3 - \frac{6}{35}u_4\Big)\Big) + \frac{81}{128}\Big(\frac{1}{5}u_3 - \frac{6}{35}u_4\Big) - \frac{13}{64}\Big(\frac{1}{7}u_4\Big) = -\frac{1}{8}\phi_b\,, \quad \text{(C.8)}$$

$$\frac{1}{16}\Big(u_1 - \frac{2}{3}u_2 + \frac{8}{15}u_3 - \frac{16}{35}u_4\Big) - \frac{25}{128}\Big(\frac{1}{3}u_2 - \frac{4}{15}u_3 + \frac{8}{35}u_4\Big) + \frac{81}{128}\Big(\frac{1}{5}u_3 - \frac{6}{35}u_4\Big) + \frac{1}{11\Sigma_5}\hat{\mathbf{n}} \cdot \nabla\Big(5\Big(\frac{1}{5}u_3 - \frac{6}{35}u_4\Big) + 6\Big(\frac{1}{7}u_4\Big)\Big) + \frac{325}{512}\Big(\frac{1}{7}u_4\Big) = \frac{1}{16}\phi_b\,, \quad \text{(C.9)}$$

$$-\frac{5}{128}\Big(u_1 - \frac{2}{3}u_2 + \frac{8}{15}u_3 - \frac{16}{35}u_4\Big) + \frac{7}{64}\Big(\frac{1}{3}u_2 - \frac{4}{15}u_3 + \frac{8}{35}u_4\Big) - \frac{105}{512}\Big(\frac{1}{5}u_3 - \frac{6}{35}u_4\Big) + \frac{325}{512}\Big(\frac{1}{7}u_4\Big) + \frac{1}{15\Sigma_7}\hat{\mathbf{n}} \cdot \nabla\Big(7\Big(\frac{1}{7}u_4\Big)\Big) = -\frac{5}{128}\phi_b\,. \quad \text{(C.10)}$$

By rearranging the system such that we have a single gradient operator in each equation, we again have a matrix system:

$$\mathbf{n} \cdot D_n \nabla u_n + \sum_{m=1}^{4} B_{nm} u_m = s_n \qquad n = 1, 2, 3, 4\,, \quad \text{(C.11)}$$

where $\mathbf{D}$ and $\mathbf{u}$ are defined as before and:

$$\mathbf{s} = \Big(\frac{1}{2}\phi_b \;\; -\frac{1}{8}\phi_b \;\; \frac{1}{16}\phi_b \;\; -\frac{5}{128}\phi_b\Big)^T\,, \quad \text{(C.12)}$$

is the source vector on the boundary and

$$\mathbf{B} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{8} & \frac{1}{16} & -\frac{5}{128} \\ -\frac{1}{8} & \frac{7}{24} & -\frac{41}{384} & \frac{1}{16} \\ \frac{1}{16} & -\frac{41}{384} & \frac{407}{1920} & -\frac{233}{2560} \\ -\frac{5}{128} & \frac{1}{16} & -\frac{233}{2560} & \frac{3023}{17920} \end{bmatrix}, \tag{C.13}$$

is a dense matrix of coefficients. Again, as pointed out by Evans, the SP_1 approximation reduces this boundary condition to the standard Marshak diffusion boundary condition. For reflecting boundary conditions, we use the same procedure as the P_N equations where the odd-moments are zero such that Eq (A.34) is true. From setting Eq (4.5) to zero for odd ϕ_n and again substituting Eq (4.9) we immediately find that:

$$\nabla \mathbf{u} = 0 \tag{C.14}$$

for reflecting SP_N boundaries, providing enough equations to close the system.

Analogously, for Marshak conditions on the boundaries of the multigroup problem we have:

$$\hat{\mathbf{n}} \cdot \mathbb{D}_n \nabla \mathbb{U}_n + \sum_{m=1}^{4} \mathbb{B}_{nm} \mathbb{U}_m = \mathbb{S}_n \,, \tag{C.15}$$

with $\mathbb{S}_n$ the vector of group-wise boundary source term on each boundary for each psuedo-moment and $\mathbb{B}_{nm}$ is an $N_g \times N_g$ diagonal matrix with the value B_{nm} on the diagonal. For reflecting conditions, again we have $\nabla \mathbb{U}_n = 0$ for all pseudo-moments.

Appendix D

Finite Volume Discretization for the SP_N Equations

The moment-discretized form of the SP_N equations given by Eq (4.12) for the monoenergetic case has yet to have the spatial components of the solution and the derivative operators discretized. To achieve this, we choose a conservation law approach, using Eq (4.12) as the balance equation, to yield a finite volume calculation (LeVeque, 2002). To do this, we first define a control volume about point (i, j, k) in the domain as a cell in a Cartesian mesh as shown in Figure D.1).

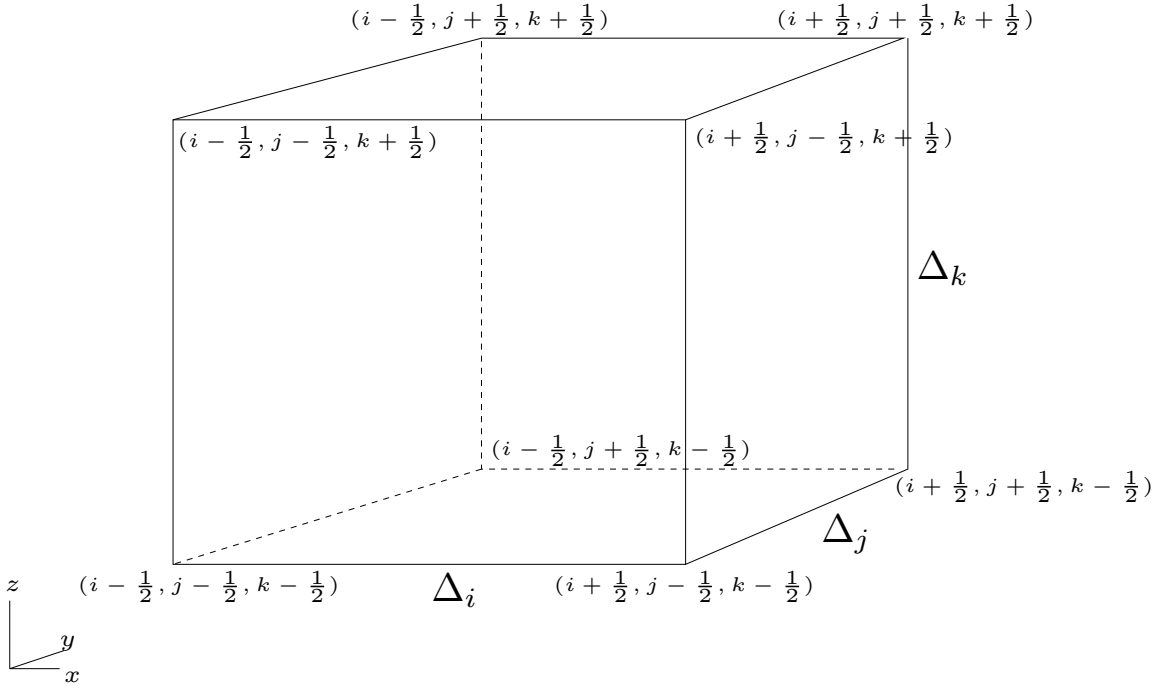

Figure D.1: **Cartesian mesh cell used for the spatial discretization of the SP_N equations.** *The cell is centered about (i, j, k) and has a volume $V_{ijk} = \Delta_i \Delta_j \Delta_k$.*

To begin, we first define the neutron current in the n^th pseudo-moment, u_n using Fick's law as:

$$J_n = -D_n \nabla u_n \,, \tag{D.1}$$

where J_n is the moment current and D_n is the moment diffusion coefficient. Next, we

substitue Eq (D.1) into Eq (4.12) to get the balance equation:

$$\nabla \cdot J_n + \sum_{m=1}^{4} A_{nm} u_m = q_n \qquad n = 1, 2, 3, 4 \,, \tag{D.2}$$

with the current term accounting for losses, q_n accounting for neutron generation, and $\sum_{m=1}^{4} A_{nm} u_m$ accounting for moment-to-moment transport. To enforce conservation, we integrate moments over the control volume of cell ijk in Figure D.1):

$$\int_{ijk} \left[\nabla \cdot J_n + \sum_{m=1}^{4} A_{nm} u_m - q_n \right] dV = 0 \qquad n = 1, 2, 3, 4 \,. \tag{D.3}$$

If for a given quantity X, we define the cell-averaged quantity X_{ijk} as:

$$X_{ijk} = \frac{\int_{ijk} X dV}{V_{ijk}} \,, \tag{D.4}$$

then Eq (D.3) can be written as:

$$\int_{ijk} \nabla \cdot J_n dV + \sum_{m=1}^{4} A_{nm,ijk} u_{m,ijk} = q_{n,ijk} \,, \tag{D.5}$$

with n now implied to be 1,2,3 or 4 and the moment matrix elements, moments, and sources now defined as volume averaged terms in the mesh cell. Using the divergence theorem, we can rewrite the remaining integral in terms of a discrete sum of area-weighted averages over the faces of the cell:

$$\int_{ijk} \nabla \cdot J_n dV = \int_{\partial} J_n \cdot \hat{n} dA = \sum_{f=1}^{6} J_{n,f} \cdot \hat{n}_f A_f \,, \tag{D.6}$$

where f is the face index, $\hat{n}_f$ is the unit normal vector for that face and A_f is the area of the face. Inserting this into Eq (D.5) and expanding the face current summation provides a discrete form of the SP_N equations:

$$\begin{aligned}(J_{n,i+1/2} - J_{n,i-1/2})\Delta_j \Delta_k + (J_{n,j+1/2} - J_{n,j-1/2})\Delta_i \Delta_k + \\ (J_{n,k+1/2} - J_{n,k-1/2})\Delta_i \Delta_j + \sum_{m=1}^{4} A_{nm,ijk} u_{m,ijk} V_{ijk} = q_{n,ijk} V_{ijk} \,,\end{aligned} \tag{D.7}$$

For the discretization we desire only cell-centered quanities and therefore we aim to elminate all $J_{n,l\pm1/2}$ terms from the discretization where $l = \{i, j, k\}$. To do this, we

enforce continuity of the moment flux and the moment currents across cell boundaries such that $J^+_{n,l\pm1/2} = J^-_{n,l\pm1/2}$. Considering $J_{n,l+1/2}$, using Eq (D.1) we then have:

$$J_{n,l+1/2} = -2D_{n,l+1/2}\frac{u_{n,l+1} - u_{n,l}}{\Delta_{l+1} + \Delta_l}\,, \tag{D.8}$$

$$J^+_{n,l+1/2} = -2D_{n,l+1}\frac{u_{n,l+1} - u_{n,l+1/2}}{\Delta_{l+1}}\,, \tag{D.9}$$

$$J^-_{n,l+1/2} = -2D_{n,l}\frac{u_{n,l+1/2} - u_{n,l}}{\Delta_l}\,. \tag{D.10}$$

To eliminate $D_{n,l+1/2}$ in Eq (D.8), we equate Eqs (D.9) and (D.10) to enforce continuity of the neutron current:

$$-2D_{n,l+1}\frac{u_{n,l+1} - u_{n,l+1/2}}{\Delta_{l+1}} = -2D_{n,l}\frac{u_{n,l+1/2} - u_{n,l}}{\Delta_l}\,. \tag{D.11}$$

Solving for $u_{n,l+1/2}$ gives:

$$u_{n,l+1/2} = \frac{\Delta_l D_{n,l+1}u_{n,l+1} + \Delta_{l+1}D_{n,l}u_{n,l}}{\Delta_l D_{n,l+1} + \Delta_{l+1}D_{n,l}}\,. \tag{D.12}$$

Next, we equate Eq (D.8) to Eq (D.9):

$$-2D_{n,l+1/2}\frac{u_{n,l+1} - u_{n,l}}{\Delta_{l+1} + \Delta_l} = -2D_{n,l+1}\frac{u_{n,l+1} - u_{n,l+1/2}}{\Delta_{l+1}}\,, \tag{D.13}$$

and rearrange:

$$\Delta_{l+1}D_{n,l+1/2}(u_{n,l+1} - u_{n,l}) = (\Delta_{l+1} + \Delta_l)D_{n,l+1}u_{n,l+1} - (\Delta_{l+1} + \Delta_l)D_{n,l+1}u_{n,l+1/2}\,. \tag{D.14}$$

Inserting Eq (D.12) into Eq (D.14) gives:

$$\begin{aligned}\Delta_{l+1}D_{n,l+1/2}(\Delta_l D_{n,l+1} + \Delta_{l+1}D_{n,l})(u_{n,l+1} - u_{n,l}) = \\ (\Delta_{l+1} + \Delta_l)D_{n,l+1}(\Delta_l D_{n,l+1} + \Delta_{l+1}D_{n,l})u_{n,l+1} - \\ (\Delta_{l+1} + \Delta_l)D_{n,l+1}(\Delta_l D_{n,l+1}u_{n,l+1} + \Delta_{l+1}D_{n,l}u_{n,l})\,.\end{aligned} \tag{D.15}$$

Expanding gives 5 terms:

$$\Delta_{l+1}D_{n,l+1/2}(\Delta_l D_{n,l+1}+\Delta_{l+1}D_{n,l})(u_{n,l+1}-u_{n,l}) = \\ (\Delta_{l+1}+\Delta_l)D_{n,l+1}\Delta_l D_{n,l+1}u_{n,l+1}+(\Delta_{l+1}+\Delta_l)D_{n,l+1}\Delta_{l+1}D_{n,l}u_{n,l+1}- \\ (\Delta_{l+1}+\Delta_l)D_{n,l+1}\Delta_l D_{n,l+1}u_{n,l+1}-(\Delta_{l+1}+\Delta_l)D_{n,l+1}\Delta_{l+1}D_{n,l}u_{n,l}\,, \quad \text{(D.16)}$$

two of which cancel, giving:

$$\Delta_{l+1}D_{n,l+1/2}(\Delta_l D_{n,l+1}+\Delta_{l+1}D_{n,l})(u_{n,l+1}-u_{n,l}) = \\ (\Delta_{l+1}+\Delta_l)\Delta_{l+1}D_{n,l+1}D_{n,l}(u_{n,l+1}-u_{n,l})\,. \quad \text{(D.17)}$$

Solving Eq (D.17) for $D_{n,l+1/2}$ then gives the face diffusion coefficient in terms of cell-centered coefficients:

$$D_{n,l+1/2}=\frac{\Delta_{l+1}D_{n,l+1}D_{n,l}+\Delta_l D_{n,l+1}D_{n,l}}{\Delta_l D_{n,l+1}+\Delta_{l+1}D_{n,l}}\,. \quad \text{(D.18)}$$

With this diffusion coefficient, we can then finish the derivation of the cell-face currents by inserting Eq (D.18) into Eq (D.8) giving:

$$J_{n,l+1/2}=-2\frac{D_{n,l+1}D_{n,l}}{\Delta_l D_{n,l+1}+\Delta_{l+1}D_{n,l}}(u_{n,l+1}-u_{n,l})\,. \quad \text{(D.19)}$$

Analagously, for cell face in the opposite direction, we have:

$$J_{n,l-1/2}=-2\frac{D_{n,l}D_{n,l-1}}{\Delta_{l-1}D_{n,l}+\Delta_l D_{n,l-1}}(u_{n,l}-u_{n,l-1})\,. \quad \text{(D.20)}$$

Before inserting the derived face fluxes into Eq (D.7), we divide by the cell volume to get:

$$(J_{n,i+1/2}-J_{n,i-1/2})\frac{1}{\Delta_i}+(J_{n,j+1/2}-J_{n,j-1/2})\frac{1}{\Delta_j}+ \\ (J_{n,k+1/2}-J_{n,k-1/2})\frac{1}{\Delta_k}+\sum_{m=1}^{4}A_{nm,ijk}u_{m,ijk}=q_{n,ijk}\,, \quad \text{(D.21)}$$

Based on the face currents computed, we identify the following coefficients that will

appear in the moment equations:

$$C_{n,l+1} = \frac{2}{\Delta_l}\left(\frac{D_{n,l+1}D_{n,l}}{\Delta_l D_{n,l+1} + \Delta_{l+1} D_{n,l}}\right), \tag{D.22}$$

$$C_{n,l-1} = \frac{2}{\Delta_l}\left(\frac{D_{n,l}D_{n,l-1}}{\Delta_{l-1} D_{n,l} + \Delta_l D_{n,l-1}}\right). \tag{D.23}$$

Inserting these coefficients into Eqs (D.19) and (D.20) gives:

$$J_{n,l+1/2} = -\Delta_l C_{n_l+1}(u_{n,l+1} - u_{n,l}) . \tag{D.24}$$

$$J_{n,l-1/2} = -\Delta_l C_{n_l-1}(u_{n,l} - u_{n,l-1}) . \tag{D.25}$$

Applying Eqs (D.24) and (D.25) Eq (D.21) gives the final spatially discretized SP_N equations:

$$-C_{n,i+1}u_{n_i+1} - C_{n,i-1}u_{n_i-1} - C_{n,j+1}u_{n_j+1} - C_{n,j-1}u_{n_j-1} - C_{n,k+1}u_{n_k+1} - C_{n,k-1}u_{n_k-1} + \sum_{m=1}^{4}\Big[A_{nm,ijk} + (C_{n,i+1} + C_{n,j+1} + C_{n,k+1} + C_{n,i-1} + C_{n,j-1} + C_{n,k-1})\delta_{nm}\Big]u_{m,ijk} = q_{n,ijk} . \tag{D.26}$$

On the boundaries of the domain, special treatment of the face currents must be made as their are no adjacent cells with which to close the system. Starting with Eq (C.11), we again insert the neutron current given by Eq (D.1):

$$\mathbf{n} \cdot J_n + \sum_{m=1}^{4} B_{nm}u_m = s_n . \tag{D.27}$$

On the low boundary we then have:

$$J_{n,1/2} = s_{n,1} - \sum_{m=1}^{4} B_{nm,1/2}u_{m,1/2} , \tag{D.28}$$

or in terms of moments:

$$J_{n,1/2} = -2D_{n,1}\frac{u_{n,1} - u_{n,1/2}}{\Delta_1} . \tag{D.29}$$

Equating Eqs (D.28) and (D.29) gives an extra equation for the boundary moments:

$$s_{n,1} = \sum_{m=1}^{4} B_{nm,1/2} u_{m,1/2} - 2D_{n,1} \frac{u_{n,1} - u_{n,1/2}}{\Delta_1} \,, \tag{D.30}$$

or

$$s_{n,1} = \sum_{m=1}^{4} \left[B_{nm,1/2} + \frac{2D_{n,1}}{\Delta_1} \delta_{mn} \right] u_{m,1/2} - 2\frac{D_{n,1}}{\Delta_1} u_{n,1} \,. \tag{D.31}$$

To use these boundary fluxes to close the system, we insert Eq (D.29) into Eq (D.21) and then add Eq (D.31) to the system to eliminate the boundary currents introduced by Eq (D.29).

For the multigroup SP_N equations, the spatial discretization is identical, this time with the multigroup moment vectors where now:

$$-\mathbb{C}_{n,i+1}\mathbb{U}_{n_i+1} - \mathbb{C}_{n,i-1}\mathbb{U}_{n_i-1} - \mathbb{C}_{n,j+1}\mathbb{U}_{n_j+1} - \mathbb{C}_{n,j-1}\mathbb{U}_{n_j-1} - \mathbb{C}_{n,k+1}\mathbb{U}_{n_k+1} - \mathbb{C}_{n,k-1}\mathbb{U}_{n_k-1} +$$
$$\sum_{m=1}^{4} \left[\mathbb{A}_{nm,ijk} + (\mathbb{C}_{n,i+1} + \mathbb{C}_{n,j+1} + \mathbb{C}_{n,k+1} + \mathbb{C}_{n,i-1} + \mathbb{C}_{n,j-1} + \mathbb{C}_{n,k-1})\delta_{nm} \right] \mathbb{U}_{m,ijk} = \mathbb{Q}_{n,ijk}, \tag{D.32}$$

and the coefficients are:

$$\mathbb{C}_{n,l+1} = \frac{2}{\Delta_l} \left(\frac{\mathbb{D}_{n,l+1}\mathbb{D}_{n,l}}{\Delta_l \mathbb{D}_{n,l+1} + \Delta_{l+1}\mathbb{D}_{n,l}} \right) , \tag{D.33}$$

$$\mathbb{C}_{n,l-1} = \frac{2}{\Delta_l} \left(\frac{\mathbb{D}_{n,l}\mathbb{D}_{n,l-1}}{\Delta_{l-1} \mathbb{D}_{n,l} + \Delta_l \mathbb{D}_{n,l-1}} \right) . \tag{D.34}$$

The boundary conditions are analagously formulated with the multigroup matrices/vectors.

references

Alexandrov, V., E. Atanassov, and I. Dimov. 2004. Parallel quasi-monte carlo methods for linear algebra problems. *Monte Carlo Methods & Applications* 10(3/4): 213–219.

Alexandrov, V.N. 1998. Efficient parallel monte carlo methods for matrix computations. *Mathematics and Computers in Simulation* 47(2âŁ“5):113–122.

Averick, Brett M., Jorge J. More, Christian H. Bischof, Alan Carle, and Andreas Griewank. 1994. Computing large sparse jacobian matrices using automatic differentiation. *SIAM Journal on Scientific Computing* 15(2):285–294.

Azmy, Yousry, and Enrico Sartori. 2010. *Nuclear computational science: A century in review*. 1st ed. Springer.

Baker, C. G., U. L. Hetmaniuk, R. B. Lehoucq, and H. K. Thornquist. 2009. Anasazi software for the numerical solution of large-scale eigenvalue problems. *ACM Trans. Math. Softw.* 36(3):1–23.

Bartlett, Roscoe A., David M. Gay, and Eric T. Phipps. 2006. Automatic differentiation of c++ codes for large-scale scientific computing. In *Computational science - ICCS 2006*, vol. 3994, 525–532. Berlin, Heidelberg: Springer Berlin Heidelberg.

Branford, S., C. Sahin, A. Thandavan, C. Weihrauch, V.N. Alexandrov, and I.T. Dimov. 2008. Monte carlo methods for matrix computations on the grid. *Future Generation Computer Systems* 24(6):605–612.

Brantley, Patrick S., and Edward Larsen. 2000. The simplified p3 approximation. *Nuclear Science and Engineering* 134(1):1–21.

Brunner, Thomas A., and Patrick S. Brantley. 2009. An efficient, robust, domain-decomposition algorithm for particle monte carlo. *Journal of Computational Physics* 228(10):3882–3890.

Brunner, Thomas A., Todd J. Urbatsch, Thomas M. Evans, and Nicholas A. Gentile. 2006. Comparison of four parallel algorithms for domain decomposed implicit monte carlo. *Journal of Computational Physics* 212(2):527–539.

Cai, Xiao-Chuan, and David E. Keyes. 2002. Nonlinearly preconditioned inexact newton algorithms. *SIAM Journal on Scientific Computing* 24(1):183–200.

Danilov, D.L., S.M. Ermakov, and J.H. Halton. 2000. Asymptotic complexity of monte carlo methods for solving linear systems. *Journal of Statistical Planning and Inference* 85(1-2):5–18.

De Vahl Davis, G. 1983. Natural convection of air in a square cavity: A bench mark numerical solution. *International Journal for Numerical Methods in Fluids* 3(3): 249–264.

Dembo, Ron S., Stanley C. Eisenstat, and Trond Steihaug. 1982. Inexact newton methods. *SIAM Journal on Numerical Analysis* 19(2):400–408.

Devine, K., E. Boman, R. Heaphy, B. Hendrickson, and C. Vaughan. 2002. Zoltan data management services for parallel dynamic applications. *Computing in Science Engineering* 4(2):90 –96.

Dimov, I., V. Alexandrov, and A. Karaivanova. 2001. Parallel resolvent monte carlo algorithms for linear algebra problems. *Mathematics and Computers in Simulation* 55(1âŁ“3):25–35.

Dimov, I.T., and V.N. Alexandrov. 1998. A new highly convergent monte carlo method for matrix computations. *Mathematics and Computers in Simulation* 47(2âŁ“5): 165–181.

Dimov, I.T., T.T. Dimov, and T.V. Gurov. 1998. A new iterative monte carlo approach for inverse matrix problem. *Journal of Computational and Applied Mathematics* 92(1):15–35.

Dimov, I.T., B. Philippe, A. Karaivanova, and C. Weihrauch. 2008. Robustness and applicability of markov chain monte carlo algorithms for eigenvalue problems. *Applied Mathematical Modelling* 32(8):1511–1529.

Dimov, Ivan, Vassil Alexandrov, Rumyana Papancheva, and Christian Weihrauch. 2007. Monte carlo numerical treatment of large linear algebra problems. In *Computational science âŁ“ ICCS 2007*, ed. Yong Shi, Geert Dick van Albada, Jack Dongarra, and Peter M. A. Sloot, 747–754. Lecture Notes in Computer Science 4487, Springer Berlin Heidelberg.

Duderstadt, James J., and Louis J. Hamilton. 1976. *Nuclear reactor analysis*. 1st ed. Wiley.

Eisenstat, Stanley C., and Homer F. Walker. 1996. Choosing the forcing terms in an inexact newton method. *SIAM Journal on Scientific Computing* 17(1):16–32.

Evans, Katherine J., D.A. Knoll, and Michael Pernice. 2006. Development of a 2-d algorithm to simulate convection and phase transition efficiently. *Journal of Computational Physics* 219(1):404–417.

———. 2007. Enhanced algorithm efficiency for phase change convection using a multigrid preconditioner with a SIMPLE smoother. *Journal of Computational Physics* 223(1):121–126.

Evans, T. M. 2013. Simplified PN methods in denovo. Technical Report RNSD-00-000, Oak Ridge National Laboratory.

Evans, Thomas, and Scott Mosher. 2009. A monte carlo synthetic acceleration method for the non-linear, time-dependent diffusion equation. *American Nuclear Society - International Conference on Mathematics, Computational Methods and Reactor Physics 2009*.

Evans, Thomas, Scott Mosher, and Stuart Slattery. 2012. A monte carlo synthetic-acceleration method for solving the thermal radiation diffusion equation. *Journal of Computational Physics* Submitted.

Evans, Thomas, Alissa Stafford, Rachel Slaybaugh, and Kevin Clarno. 2010. Denovo: A new three-dimensional parallel discrete ordinates code in SCALE. *Nuclear Technology* 171(2):171–200.

Evans, Thomas, Todd Urbatsch, H Lichtenstein, and Morel. 2003. A residual monte carlo method for discrete thermal radiative diffusion. *Journal of Computational Physics* 189(2):539–556.

Forsythe, George E., and Richard A. Leibler. 1950. Matrix inversion by a monte carlo method. *Mathematical Tables and Other Aids to Computation* 4(31):127–129. ArticleType: research-article / Full publication date: Jul., 1950 / Copyright 1950 American Mathematical Society.

Gartling, David K. 1990. A test problem for outflow boundary conditions - flow over a backward-facing step. *International Journal for Numerical Methods in Fluids* 11(7):953 – 967.

Gaston, D, G Hansen, S Kadioglu, D A Knoll, C Newman, H Park, C Permann, and W Taitano. 2009. Parallel multiphysics algorithms and software for computational nuclear engineering. *Journal of Physics: Conference Series* 180:012012.

Gentile, N.A., Malvin Kalos, and Thomas A. Brunner. 2005. Obtaining identical results on varying numbers of processors in domain decomposed particle monte carlo simulations. In *Computational methods in transport*, vol. 48, 423–433. Berlin/Heidelberg: Springer-Verlag.

Ghia, U, K.N Ghia, and C.T Shin. 1982. High-re solutions for incompressible flow using the navier-stokes equations and a multigrid method. *Journal of Computational Physics* 48(3):387–411.

Godoy, William F., and Xu Liu. 2012. Parallel jacobian-free newton krylov solution of the discrete ordinates method with flux limiters for 3D radiative transfer. *Journal of Computational Physics* 231(11):4257–4278.

Gropp, W., and B. Smith. 1993. Scalable, extensible, and portable numerical libraries. In *Scalable parallel libraries conference, 1993., proceedings of the*, 87 –93.

Gropp, William D, Dinesh K Kaushik, David E Keyes, and Barry F Smith. 2001. High-performance parallel implicit CFD. *Parallel computing in aerospace* 27(4): 337–362.

Halton, J. H. 1962. Sequential monte carlo. *Mathematical Proceedings of the Cambridge Philosophical Society* 58(01):57–78.

———. 1994. Sequential monte carlo techniques for the the solution of linear systems. *Journal of Scientific Computing* 9:213–257.

Halton, John H. 1970. A retrospective and prospective survey of the monte carlo method. *SIAM Review* 12(1):1–63.

———. 2006. Sequential monte carlo techniques for solving non-linear systems. *Monte Carlo Methods & Applications* 12(2):113–141.

Hammersley, John Michael, and David Christopher Handscomb. 1964. *Monte carlo methods*. Methuen.

Heroux, Michael A., Roscoe A. Bartlett, Vicki E. Howle, Robert J. Hoekstra, Jonathan J. Hu, Tamara G. Kolda, Richard B. Lehoucq, Kevin R. Long, Roger P.

Pawlowski, Eric T. Phipps, Andrew G. Salinger, Heidi K. Thornquist, Ray S. Tuminaro, James M. Willenbring, Alan Williams, and Kendall S. Stanley. 2005. An overview of the trilinos project. *ACM Trans. Math. Softw.* 31(3):397–423.

Ji, Hao, and Yaohang Li. 2012. Reusing random walks in monte carlo methods for linear systems. *Procedia Computer Science* 9(0):383–392.

Kelley, C. T. 1995. *Iterative methods for linear and nonlinear equations*. 1st ed. Society for Industrial and Applied Mathematics.

Keyes, D. E. 1999. *How scalable is domain decomposition in practice?*

Keyes, David E., Dinesh K. Kaushik, and Barry F. Smith. 1997. Prospects for CFD on petaflops systems. Tech. Rep.

Knoll, D.A., and D.E. Keyes. 2004. Jacobian-free newton-krylov methods: a survey of approaches and applications. *Journal of Computational Physics* 193(2):357–397.

Knoll, D.A., and P.R. McHugh. 1995. Newton-krylov methods applied to a system of convection-diffusion-reaction equations. *Computer Physics Communications* 88(2-3): 141–160.

Kogge, Peter M., and Timothy J. Dysart. 2011. Using the TOP500 to trace and project technology and architecture trends. In *Proceedings of 2011 international conference for high performance computing, networking, storage and analysis*, 28:1–28:11. SC '11, New York, NY, USA: ACM.

LeVeque, Randall. 2007. *Finite difference methods for ordinary and partial differential equations: Steady-state and time-dependent problems*. SIAM, Society for Industrial and Applied Mathematics.

LeVeque, Randall J. 2002. *Finite volume methods for hyperbolic problems*. 1st ed. Cambridge University Press.

Lewis, E. E. 1993. *Computational methods of neutron transport*. Wiley-Interscience.

Li, Yaohang, and Michael Mascagni. 2003. Analysis of large-scale grid-based monte carlo applications. *The International Journal of High Performance Computing Applications* 17(4):369–382.

McHugh, P. R., and D. A. Knoll. 1992. *Fully implicit solutions of the benchmark backward facing step problem using finite element discretization and inexact newton's method*.

McHugh, Paul R., and Dana A. Knoll. 1993. Inexact newton's method solutions to the incompressible navier-stokes and energy equations using standard and matrix-free implementations. In *AIAA 11th computational fluid dynamics conference*, vol. -1, 385–393.

———. 1994. Fully coupled finite volume solutions of the incompressible NavierâŁ“Stokes and energy equations using an inexact newton method. *International Journal for Numerical Methods in Fluids* 19(5):439–455.

Mervin, Brenden, S.W. Mosher, Thomas Evans, John Wagner, and GI Maldonado. 2012. Variance estimation in domain decomposed monte carlo eigenvalue calculations. *PHYSOR 2012 - Advances in Reactor Physics*.

Musser, David R., and Alexander A. Stepanov. 1994. Algorithm-oriented generic libraries. *Software: Practice and Experience* 24(7):623–642.

Nachtigal, Noel M., Satish C. Reddy, and Lloyd N. Trefethen. 1992. How fast are nonsymmetric matrix iterations? *SIAM Journal on Matrix Analysis and Applications* 13(3):778–795.

Notz, P.K., and Pawlowski. 2010. Graph-based software design for managing complexity and enabling concurrency in multiphysics PDE software. *ACM Trans. Math. Softw.* 40:1–24.

Pawlowski, John N. Shadid, Thomas Smith, Eric Cyr, and Paula Weber. 2012. Drekar CFD - a turbulent fluid-flow and conjugate heat transfer code: Theory manual (version 1.0). Technical Report, Sandia National Laboratories.

Pawlowski, Roger P., John N. Shadid, Joseph P. Simonis, and Homer F. Walker. 2006. Globalization techniques for newton-krylov methods and applications to the fully coupled solution of the navier-stokes equations. *SIAM Review* 48(4):700–721.

Pernice, Michael, and Homer F. Walker. 1998. NITSOL: a newton iterative solver for nonlinear systems. *SIAM Journal on Scientific Computing* 19(1):302–318.

Pletcher, Richard H., John C. Tannehill, and Dale Anderson. 1997. *Computational fluid mechanics and heat transfer, second edition*. 2nd ed. Taylor & Francis.

Procassini, Richard, M.H. O'Brien, and J Taylor. 2005. Dynamic load balancing of parallel monte carlo transport calculations. *Monte Carlo 2005*.

Rief, H. 1999. Touching on a zero-variance scheme in solving linear equations by random walk processes. *Monte Carlo Methods & Applications* 5(2):135.

Romano, P., B. Forget, and F. Brown. 2010. Towards scalable parallelism in monte carlo particle transport codes using remote memory access. In *Joint international conference on supercomputing in nuclear applications and monte carlo 2010 (SNA+ MC2010)*, 17–21.

Saad, Youcef. 1993. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing* 14(2):9.

Saad, Youcef, and Martin H. Schultz. 1986. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 7(3):856–869.

Saad, Yousef. 2003. *Iterative methods for sparse linear systems*. SIAM.

Sabelfeld, K., and N. Mozartova. 2009. Sparsified randomization algorithms for large systems of linear equations and a new version of the random walk on boundary method. *Monte Carlo Methods & Applications* 15(3):257–284.

Shadid, John N., and Ray S. Tuminaro. 1994. A comparison of preconditioned nonsymmetric krylov methods on a large-scale MIMD machine. *SIAM Journal on Scientific Computing* 15(2):440–459.

Shadid, John N., Ray S. Tuminaro, and Homer F. Walker. 1997. An inexact newton method for fully coupled solution of the navier-stokes equations with heat and mass transport. *Journal of Computational Physics* 137(1):155–185.

Siegel, A., K. Smith, P. Fischer, and V. Mahadevan. 2012a. Analysis of communication costs for domain decomposed monte carlo methods in nuclear reactor analysis. *Journal of Computational Physics* 231(8):3119–3125.

Siegel, A.R., K. Smith, P.K. Romano, B. Forget, and K. Felker. 2012b. The effect of load imbalances on the performance of monte carlo algorithms in LWR analysis. *Journal of Computational Physics* (0).

Slaybaugh, Rachel N. 2011. Acceleration methods for massively parallel deterministic transport. PhD nuclear engineering and engineering physics, University of Wisconsin-Madison, Madison, WI.

Sosonkina, M., D.C.S. Allison, and L.T. Watson. 1998. Scalable parallel implementations of the GMRES algorithm via householder reflections. In *1998 international conference on parallel processing, 1998. proceedings*, 396 –404.

Spanier, Jerome, and Ely M. Gelbard. 1969. *Monte carlo principles and neutron transport problems*. New York: Dover Publications.

Srinivasan, A. 2010. Monte carlo linear solvers with non-diagonal splitting. *Mathematics and Computers in Simulation* 80(6):1133–1143.

Stroustrup, Bjarne. 1997. *The c++ programming language*. 3rd ed. Addison-Wesley Professional.

Trefethen, Lloyd N., and David Bau III. 1997. *Numerical linear algebra*. SIAM: Society for Industrial and Applied Mathematics.

Tuminaro, Ray S., John N. Shadid, and Scott A. Hutchinson. 1998. Parallel sparse matrix vector multiply software for matrices with data locality. *Concurrency: Practice and Experience* 10(3):229–247.

U.S. Department of Energy. 2011. CASL - a project summary. CASL-U-2011-0025-000, Consortium for Advanced Simulation of LWRs.

———. 2012. Resilient extreme-scale solvers. Tech. Rep. LAB 12-742, ASCR.

Vajargah, Behrouz Fathi. 2007. New advantages to obtain accurate matrix inversion. *Applied Mathematics and Computation* 189(2):1798–1804.

———. 2008. Stochastic inverse matrix computation with minimum variance of errors. *Applied Mathematics and Computation* 199(1):181–185.

Vajargah, Behrouz Fathi, and Kianoush Fathi Vajargah. 2006. Parallel monte carlo computations for solving SLAE with minimum communications. *Applied Mathematics and Computation* 183(1):1–9.

Wagner, JC, Scott Mosher, Thomas Evans, Doug Peplow, and John Turner. 2010. Hybrid and parallel domain-decomposition methods development to enable monte carlo for reactor analyses. *Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo*.

Wang, Qiqi, David Gleich, Amin Saberi, Nasrollah Etemadi, and Parviz Moin. 2008. A monte carlo method for solving unsteady adjoint equations. *Journal of Computational Physics* 227(12):6184–6205.

Wasow, W.R. 1952. A note on the inversion of matrices by random walks. *Mathematical Tables and Other Aids to Computation* 6(38):78–81.

Zienkiewicz, O. C., R. L. Taylor, and J. Z. Zhu. 2005. *The finite element method: Its basis and fundamentals, sixth edition*. 6th ed. Butterworth-Heinemann.

Ã–kten, Giray. 2005. Solving linear equations by monte carlo simulation. *SIAM Journal on Scientific Computing* 27(2):511–531.