PARALLEL MONTE CARLO SYNTHETIC ACCELERATION METHODS FOR DISCRETE TRANSPORT PROBLEMS

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

15 June 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

PARALLEL MONTE CARLO SYNTHETIC ACCELERATION METHODS FOR DISCRETE TRANSPORT PROBLEMS

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 nearly all high-fidelity nuclear reactor simulations, linear and nonlinear transport 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 linear and 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 transport 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 physics-based motivation for advancing Monte Carlo solvers and studying their application to transport systems by providing 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. The organization of the document is then reviewed followed by an explanation of the key outcomes of this work and where in the document results and data supporting those outcomes can be located.

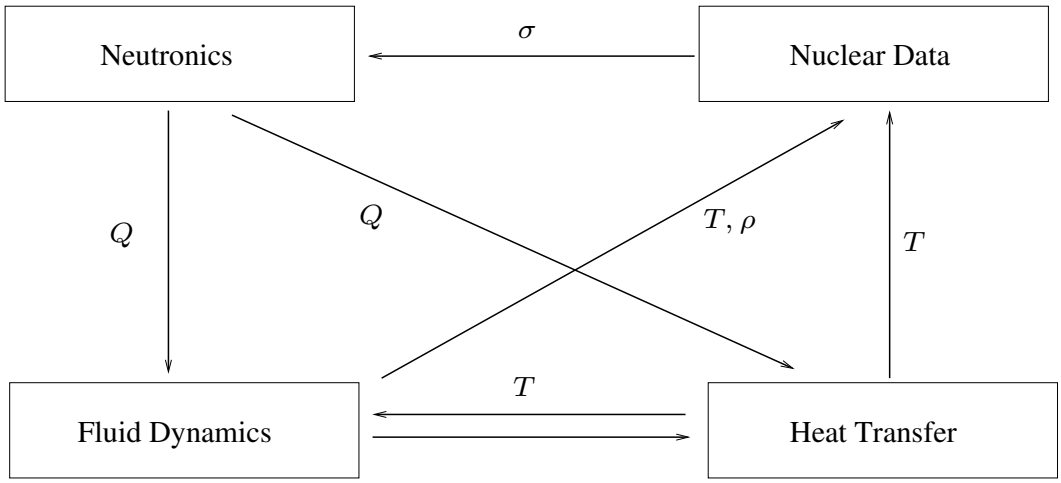

Figure 1.1: **Multiphysics dependency analysis of departure from nucleate boiling.** *A neutronics solution is required to compute power generation in the fuel pins, fluid dynamics is required to characterize boiling and fluid temperature and density, heat transfer is required to compute the fuel and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

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

1.1.1 Solutions for the SP_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 fission reactor 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 neutron transport for fission reactor analysis. The reduction in numerical complexity of current 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.

In order to leverage MCSA as a solution technique for physics problems, its current formulation requires the linear operator and all preconditioners to be explicitly formed. Recent developments in the Exnihilo neutronics package at Oak Ridge National Laboratory have permitted generation of the SP_N system of equations for detailed neutronics models of fission reactor-based systems(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 algebraic preconditioning methods. 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. 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.

1.1.2 Solutions for the Navier-Stokes Equations

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 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 are 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 work 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.

To present these new developments this document is arranged in the following manner. First, in Chapter 2, the fundamentals of the Monte Carlo method for linear systems are presented. Using this background on Monte Carlo, synthetic acceleration methods are then presented and analyzed using a simple model transport problem. Next, in Chapter 3, the Monte Carlo methods developed in Chapter 2 are applied to the neutron transport problem. Specifically, the SP_N form of the Boltzmann neutron transport equation is developed and solved using the Monte Carlo methods. Using a difficult fission reactor fuel assembly criticality calculation to drive research and development for these Monte Carlo methods, several important issues regarding Monte Carlo solver applicability and performance will be discussed along with solutions to these problems. Using these solutions, the Monte Carlo methods are then verified against modern solution techniques for the SP_N equations in order verify correctness for the fuel assembly criticality calculation. In Chapter 4, the Monte Carlo methods are applied to the Navier-Stokes equations as a model nonlinear energy and momentum transport system. Additional difficulties that arise when using Monte Carlo methods in these types of systems will be presented along with the subsequently developed solutions. In Chapter 5, the Monte Carlo methods are parallelized using modern reactor physics techniques for particle transport with a full parallel scaling analysis provided using a leadership-class computing facility. Finally, the work is summarized in Chapter 6 and topics of future work derived from the results of this research presented.

Chapter 2

Monte Carlo Synthetic Acceleration Methods

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 Leibler 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.

Following this work, Halton presented a residual Monte Carlo scheme that improved dramatically on the performance of these methods (Halton, 1962). Recent work in this area has utilized these residual Monte Carlo methods in particle transport (Evans et al., 2003), giving accelerated convergence over traditional methods. As an expansion of Halton's work, Evans and others have leveraged these methods in a synthetic acceleration scheme that has been shown to be competitive with production Krylov methods for radiation transport problems and improve performance significantly over Halton's sequential method (Evans et al., 2012).

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 method of Halton and Monte Carlo Synthetic Acceleration methods will then be presented as a means of leveraging the basic Monte Carlo methods devised by Neumann and Ulam 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 along with other parameters of the algorithms on the Monte Carlo solutions explored using a simple model transport problem.

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:

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

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:

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

2.2 Neumann-Ulam Method

We begin our discussion of Monte Carlo methods 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 (A.3) to define the following operator[1]:

$$\mathbf{H} = \mathbf{I} - \mathbf{A} \,, \tag{2.7}$$

such that we are solving the system:

$$\mathbf{x} = \mathbf{H}\mathbf{x} + \mathbf{b} \,. \tag{2.8}$$

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

$$\mathbf{A}^{-1} = (\mathbf{I} - \mathbf{H})^{-1} = \sum_{k=0}^{\infty} \mathbf{H}^k \,, \tag{2.9}$$

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:

$$\mathbf{A}^{-1}\mathbf{b} = \sum_{k=0}^{\infty} \mathbf{H}^k\mathbf{b} = \mathbf{x} \,. \tag{2.10}$$

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:

$$x_i = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \dots \sum_{i_k}^{N} h_{i,i_1} h_{i_1,i_2} \dots h_{i_{k-1},i_k} b_{i_k} \,, \tag{2.11}$$

which can interpreted as a series of transitions between states,

$$\nu = i \to i_1 \to \cdots \to i_{k-1} \to i_k \,, \tag{2.12}$$

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

[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.

matrix, we then form the *Neumann-Ulam decomposition* of $\mathbf{H}$:

$$\mathbf{H} = \mathbf{P} \circ \mathbf{W} \,, \tag{2.13}$$

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 formulation of $\mathbf{P}$ and $\mathbf{W}$ will be dependent on whether we choose a direct or adjoint Monte Carlo sequence to estimate the state transitions in Eq (2.11). In the direct method, we will use the provided linear operator $\mathbf{A}$ 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.

2.3 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 forward Neumann-Ulam decomposition per Dimov's approach by first choosing a probability matrix that is a row scaling of $\mathbf{H}$ such that its components are:

$$p_{ij} = \frac{|h_{ij}|}{\sum_j |h_{ij}|} \,. \tag{2.14}$$

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 $\mathbf{A}$ 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 (2.13) as:

$$w_{ij} = \frac{h_{ij}}{p_{ij}} \,. \tag{2.15}$$

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

It should be noted here that if $\mathbf{A}$ is sparse, then $\mathbf{H}$, $\mathbf{P}$, and $\mathbf{W}$ must be sparse as well by definition. Additionally, we only compute $\mathbf{P}$ and $\mathbf{W}$ from the non-zero elements of $\mathbf{H}$ 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 forward solution. For a given random walk permutation ν, we define the weight of that permutation on the m^{th} step to be:

$$W_m = w_{i_0,i_1} w_{i_1,i_2} \cdots w_{i_{m-1},i_m} \,, \tag{2.16}$$

such that the weight of each transition event contributes to the total through multiplication 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*:

$$X_{i_0=i}(\nu) = \sum_{m=0}^{k} W_m b_{i_m} \,, \tag{2.17}$$

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:

$$P_\nu = p_{i,i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} \,. \tag{2.18}$$

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

$$E\{X_i\} = \sum_{\nu} P_\nu X_i(\nu) \,, \tag{2.19}$$

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

$$\begin{aligned} E\{X_i\} &= \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} p_{i,i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} w_{i,i_1} w_{i_1,i_2} \cdots w_{i_{k-1},i_k} b_{i_k} \\ &= x_i \,, \end{aligned} \tag{2.20}$$

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

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 (2.14) and (2.15) will create a random walk weight in Eq (2.16) that approaches, but never reaches zero. We do this by noticing that the factors added to Eq (2.16) will become diminishingly small due to their definition in Eq (2.15) 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.

2.3.1 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:

$$\sigma_i{}^2 = E\{X_i - (\mathbf{A}^{-1}\mathbf{b})_i\}^2 = E\{X_i^2\} - x_i^2 \,, \tag{2.21}$$

where the vector exponentials are computed element-wise. Inserting Eq (2.19) gives:

$$\sigma_i^2 = \sum_\nu P_\nu X_i(\nu)^2 - x_i^2 \,, \tag{2.22}$$

and applying Eq (2.17):

$$\sigma_i^2 = \sum_\nu P_\nu \Big(\sum_{m=0}^{k} W_m^2 b_{i_m}^2 + 2 \sum_{\substack{j=0 \\ m<j}}^{k} W_m W_j b_{i_m} b_{i_j} \Big) - x_i^2 \,. \tag{2.23}$$

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

$$\sigma_i^2 = \sum_\nu P_\nu \sum_{m=0}^{k} W_m^2 b_{i_m}^2 + 2 \sum_\nu P_\nu \sum_{\substack{j=0 \\ m<j}}^{k} W_m W_j b_{i_m} b_{i_j} - x_i^2 \,. \tag{2.24}$$

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

$$\sum_\nu P_\nu \sum_{m=0}^{k} W_m^2 b_{i_m}^2 = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} p_{i,i_1} p_{i_1,i_2} \ldots p_{i_{k-1},i_k} w_{i,i_1}^2 w_{i_1,i_2}^2 \ldots w_{i_{k-1},i_k}^2 b_{i_k}^2 \,. \tag{2.25}$$

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 (2.25):

$$(\mathbf{P} \circ \mathbf{W} \circ \mathbf{W})^k = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} p_{i,i_1} p_{i_1,i_2} \cdots p_{i_{k-1},i_k} w_{i,i_1}^2 w_{i_1,i_2}^2 \cdots w_{i_{k-1},i_k}^2 \,. \tag{2.26}$$

If we assign $\mathbf{G} = \mathbf{P} \circ \mathbf{W} \circ \mathbf{W}$ as in Eq (2.13), we then have:

$$(\mathbf{P} \circ \mathbf{W} \circ \mathbf{W})^k = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} g_{i,i_1} g_{i_1,i_2} \cdots g_{i_{k-1},i_k} \,, \tag{2.27}$$

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

$$\mathbf{T} = \sum_{k=0}^{\infty} \mathbf{G}^k \,, \tag{2.28}$$

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

$$\sigma_i^2 = (\mathbf{T}\mathbf{b}^2)_i + 2 \sum_{\nu} P_\nu \sum_{\substack{j=0 \\ m<j}}^{k} W_m W_j b_{i_m} b_{i_j} - x_i^2 \,. \tag{2.29}$$

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:

$$|\mathbf{H} \circ \mathbf{W}| \geq |\mathbf{H}| \, |\mathbf{W}| \,. \tag{2.30}$$

Using the norm property of the Hadamard product and Eq (2.13), we can define the norm of $\mathbf{W}$ as:

$$\frac{|\mathbf{H}|}{|\mathbf{P}|} \geq |\mathbf{W}| \,. \tag{2.31}$$

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

$$|\mathbf{G}| \geq |\mathbf{H}|^2 \tag{2.32}$$

Using these relations to analyze Eq (2.29), we see that if $\rho(\mathbf{G}) > 1$, then the first sum in Eq (2.28) will not converge and an infinite variance will arise as the elements of $\mathbf{T}$ become infinite in Eq (2.29). We must restrict $\mathbf{G}$ to alleviate this and therefore restrict $\mathbf{H}$ due to Eq (2.32) with $\rho(\mathbf{H}) < 1$ so that our expectation values for the solution may have a finite variance.

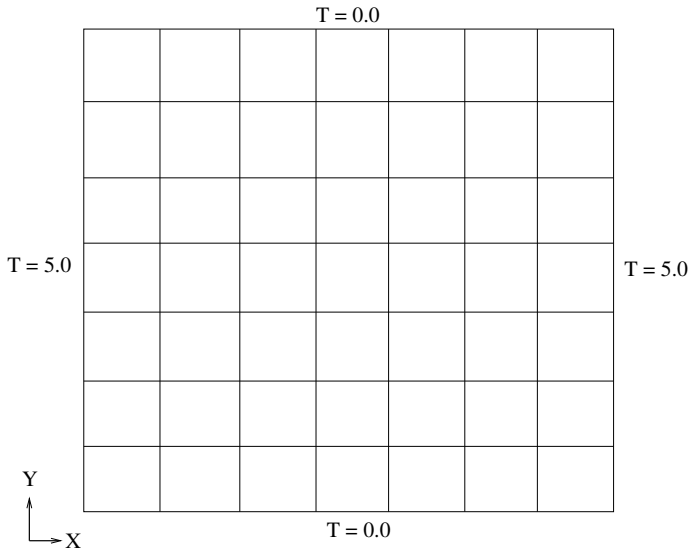

Figure 2.1: **Problem setup for 2D heat equation.** *Dirichlet conditions are set for the temperature on all 4 boundaries of the Cartesian grid. Background source of 1/5 the value of the boundary sources present.* 50×50 *grid.*

2.3.2 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 2.1. 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 nature of the method. Figure 2.2 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 a 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, analogous to the convergence of a traditional iterative method.

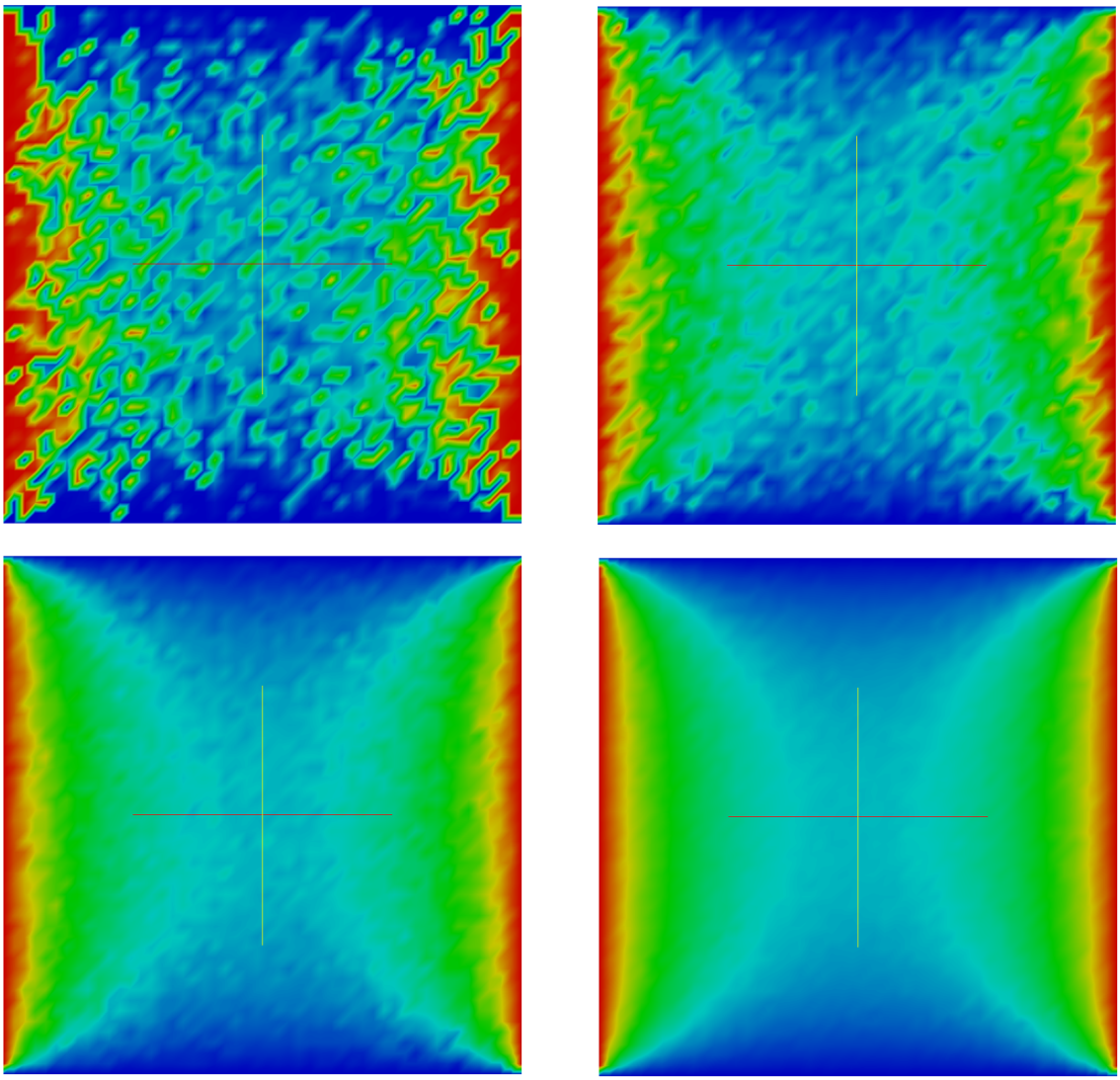

Figure 2.2: **Direct Monte Carlo solution to the heat equation with varying numbers of histories.** *Top left: 1 history per state. Top right: 10 histories per state. Bottom left: 100 histories per state. Bottom right: 1000 histories per state.*

2.4 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):

$$\mathbf{A}^T\mathbf{y} = \mathbf{d}\,, \tag{2.33}$$

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 (2.8):

$$\mathbf{y} = \mathbf{H}^T\mathbf{y} + \mathbf{d}\,. \tag{2.34}$$

As was required for convergence with the direct method using Eq (2.8), 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):

$$\langle \mathbf{A}^T\mathbf{x}, \mathbf{y}\rangle = \langle \mathbf{x}, \mathbf{A}\mathbf{y}\rangle\,. \tag{2.35}$$

it follows that:

$$\langle \mathbf{x}, \mathbf{d}\rangle = \langle \mathbf{y}, \mathbf{b}\rangle\,. \tag{2.36}$$

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 (2.34) as:

$$\mathbf{y} = (\mathbf{I} - \mathbf{H}^T)^{-1}\mathbf{d}\,, \tag{2.37}$$

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

$$\mathbf{y} = \sum_{k=0}^{\infty}(\mathbf{H}^T)^k\mathbf{d}\,. \tag{2.38}$$

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:

$$y_i = \sum_{k=0}^{\infty}\sum_{i_1}^{N}\sum_{i_2}^{N}\ldots\sum_{i_k}^{N} h_{i_k,i_{k-1}}\ldots h_{i_2,i_1}h_{i_1,i}d_{i_k}\,. \tag{2.39}$$

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 (2.36) as the first constraint and as a second constraint we select:

$$\mathbf{d} = \boldsymbol{\delta}_j \,, \tag{2.40}$$

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 (2.40) to our first constraint Eq (2.36), we get the following convenient outcome:

$$\langle \mathbf{y}, \mathbf{b} \rangle = \langle \mathbf{x}, \boldsymbol{\delta}_j \rangle = x_j \,, \tag{2.41}$$

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:

$$P_{i_0=i}(\nu) = \frac{|b_i|}{||\mathbf{b}||_1} \,. \tag{2.42}$$

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 $\mathbf{H}$:

$$\mathbf{H}^T = \mathbf{P} \circ \mathbf{W} \,, \tag{2.43}$$

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:

$$p_{ij} = \frac{|h_{ji}|}{\sum_j |h_{ji}|} \,, \tag{2.44}$$

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 (2.43), the transition weight is then defined as:

$$w_{ij} = \frac{h_{ji}}{p_{ij}} \, . \tag{2.45}$$

Using the decomposition we can then define an expectation value for the adjoint method. Given Eq (2.16) as the weight generated for a particular random walk permutation as in Eq (2.12) and our result from Eq (2.41) 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*:

$$X_j(\nu) = \sum_{m=0}^{k} W_m \delta_{i_m,j} \, , \tag{2.46}$$

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 (2.46) does not have a dependency on the source state as in Eq (2.46), 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:

$$E\{X_j\} = \sum_{\nu} P_\nu X_j(\nu) \tag{2.47}$$

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

$$\begin{aligned} E\{X_j\} &= \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \dots \sum_{i_k}^{N} b_{i_0} h_{i_0,i_1} h_{i_1,i_2} \dots h_{i_{k-1},i_k} \delta_{i_k,j} \\ &= x_j \, , \end{aligned} \tag{2.48}$$

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 (2.48) 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*:

$$W_f = W_c b_{i_0} \,, \tag{2.49}$$

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.

2.4.1 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:

$$\sigma_j^2 = \sum_\nu P_\nu \sum_{m=0}^{k} W_m^2 \delta_{i_m,j} + 2\sum_\nu P_\nu \sum_{\substack{l=0 \\ m<l}}^{k} W_m W_l \delta_{i_m,j}\delta_{i_l,j} - x_j^2 \,. \tag{2.50}$$

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 Neumann 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:

$$\sigma_j^2 = \sum_{k=0}^{\infty}\sum_{i_1}^{N}\sum_{i_2}^{N}\cdots\sum_{i_k}^{N} p_{i_k,i_{k-1}}\cdots p_{i_2,i_1}p_{i_1,i_0} w_{i_k,i_{k-1}}^2 \cdots w_{i_2,i_1}^2 w_{i_1,i_0}^2 b_{i_0}^2 \delta_{i_k,j} + \\ 2\sum_\nu P_\nu \sum_{\substack{l=0 \\ m<l}}^{k} W_m W_l \delta_{i_m,j}\delta_{i_l,j} - x_j^2 \,. \tag{2.51}$$

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

$$\sigma_j^2 = \sum_{k=0}^{\infty} \sum_{i_1}^{N} \sum_{i_2}^{N} \cdots \sum_{i_k}^{N} g_{i_k,i_{k-1}} \cdots g_{i_2,i_1} g_{i_1,i_0} b_{i_0} \delta_{i_k,j} + 2 \sum_{\nu} P_\nu \sum_{\substack{l=0 \\ m<l}}^{k} W_m W_l \delta_{i_m,j} \delta_{i_l,j} - x_j^2 \,, \quad (2.52)$$

where now $\mathbf{G}$ in this case is the transpose of that used in Eq (2.27). 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 (2.52) relative to its formulation in the forward estimator, then Eq (2.52) and Eq (2.27) 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.

2.4.2 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:

$$X_j(\nu) = b_j + \sum_{m=0}^{k} W_m h_{j,i_m} \quad (2.53)$$

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 (2.53):

$$X_j(\nu) = b_j + \sum_{m=0}^{k} \sum_{i=1}^{N} W_m \delta_{i_m,i} h_{ji} \quad (2.54)$$

where N is the number of states in the system. Immediately, we see the collision estimator as defined by Eq (2.46) and can therefore write the expectation value as:

$$E\{X_j\} = b_j + \sum_{i=1}^{N} E\{X_i\} h_{ji} \tag{2.55}$$

which is equivalently is the j^{th} component of Eq (2.8):

$$E\{X_j\} = b_j + \sum_{i=1}^{N} x_i h_{ji} \tag{2.56}$$

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.

2.4.2.1 Expected Value Estimator Variance

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

$$\sigma_j^2 = \sum_{\nu} P_\nu b_j^2 + 2 \sum_{\nu} P_\nu b_j \sum_{m=0}^{k} W_m h_{j,i_m} + \sum_{\nu} P_\nu \sum_{m=0}^{k} W_m^2 h_{j,i_m}^2 + 2 \sum_{\nu} P_\nu \sum_{\substack{l=0 \\ m<l}}^{k} W_m W_l h_{j,i_m} h_{j,i_l} - x_j^2 \,. \tag{2.57}$$

As the delta functions are no longer present, the final summation in Eq (2.57) 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.

2.4.3 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

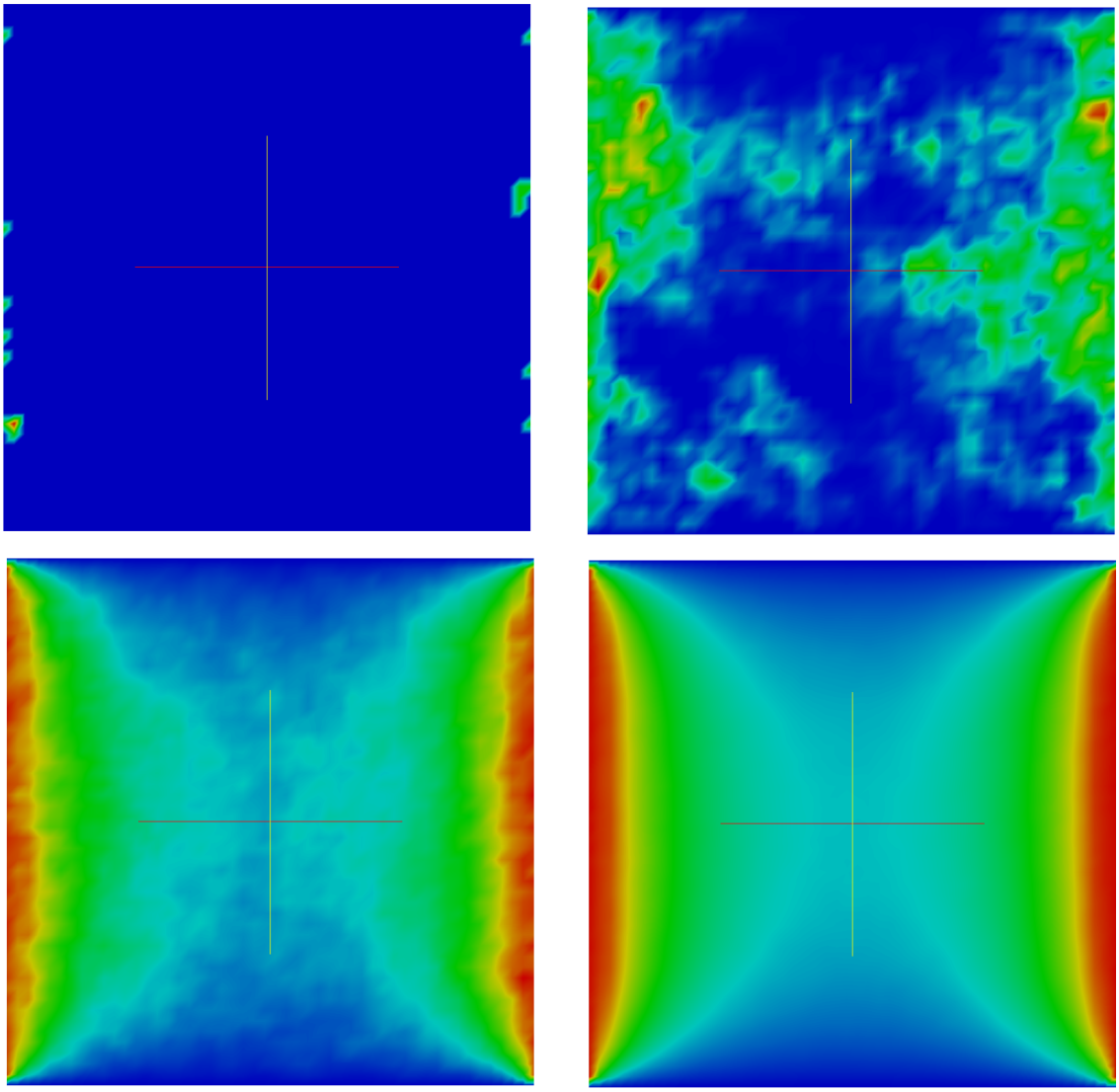

Figure 2.3: **Adjoint Monte Carlo solution to the heat equation with varying numbers of histories.** *Top left: 10 histories per state. Top right: 1,000 histories per state. Bottom left: 100,000 histories per state. Bottom right: 10,000,000 histories per state.*

sides of the domain and a smaller uniform source as shown in Figure 2.1. 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 2.3 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 2.2 and 2.3 is precisely that determined by their mathematics. As the adjoint solution evolves with the addition of histories, more histories emanate from the boundary the smaller uniform source in the domain with more penetrating from the boundary into the domain and making contributions to the tallies in those states as they are transported.

2.5 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 the projection methods outlined in Appendix A, 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)

$$\mathbf{r}^k = \mathbf{b} - \mathbf{A}\mathbf{x}^k \,, \tag{2.58a}$$

$$\mathbf{A}\boldsymbol{\delta}^k = \mathbf{r}^k \,, \tag{2.58b}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \boldsymbol{\delta}^k \,, \tag{2.58c}$$

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.

2.6 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 as 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)

$$\mathbf{x} = (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b} \,. \tag{2.59}$$

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

$$\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}^{k} + \mathbf{b} \,, \tag{2.60}$$

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:

$$\delta\mathbf{x}^{k} = \mathbf{x} - \mathbf{x}^{k} \,. \tag{2.61}$$

Subtracting Eq (2.60) from Eq (2.59) we get:

$$\delta\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\delta\mathbf{x}^{k} \,. \tag{2.62}$$

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

$$\begin{aligned} \mathbf{A}\delta\mathbf{x}^{k+1} &= (\mathbf{I} - \mathbf{A})(\mathbf{x}^{k+1} - \mathbf{x}^{k}) \\ &= \mathbf{r}^{k+1} \,. \end{aligned} \tag{2.63}$$

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

$$\mathbf{x}^{k+1} = (\mathbf{I} - \mathbf{A})\mathbf{x}^{k} + \mathbf{b} \,, \tag{2.64a}$$

$$\mathbf{A}\delta\mathbf{x}^{k+1} = \mathbf{r}^{k+1} \,, \tag{2.64b}$$

$$\mathbf{x} = \mathbf{x}^{k+1} + \delta\mathbf{x}^{k+1} \,. \tag{2.64c}$$

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:

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + \mathbf{r}^{k} \,, \tag{2.65a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \,, \tag{2.65b}$$

$$\mathbf{A}\delta\mathbf{x}^{k+1/2} = \mathbf{r}^{k+1/2} \,, \tag{2.65c}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2} \,, \tag{2.65d}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \,, \tag{2.65e}$$

where a Neumann-Ulam 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 Neumann-Ulam method to generate the correction. In addition, the fact that the scheme in Eq (2.64) 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:

$$||\mathbf{r}||_{\infty} < \epsilon \, ||\mathbf{b}||_{\infty} \,, \tag{2.66}$$

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 (2.66). We therefore have 3 parameters to tune in an MCSA implementation: the number of Monte Carlo histories computed in the Neumann-Ulam solve during each MCSA iteration, the weight cutoff for those histories, and the total MCSA convergence tolerance as specified by ϵ.

2.6.1 Alternative Fixed Point Iterations

In addition to the basic Richardson iteration, the MCSA algorithm presented in Eq (2.65) can be used to accelerate any fixed point iteration which only depends on the state (typically the residual) of the last iteration. As outlined in Appendix A, subspace methods take on a general form using the Petrov-Galerkin conditions as constraints for extracting a correction from the search subspace as given by Eq (A.16). Those that are one-dimensional may be used with fixed-point MCSA as they only depend the previous iteration for information. As an example, for positive-definite but not necessarily symmetric problems, the minimal residual (MINRES) iteration (Saad, 2003) can be used in conjunction with MCSA where the residual vector, $\mathbf{r}$, defines the search subspace and the action of the linear operator on the residual vector, $\mathbf{Ar}$, defines the constraint subspace. Using this, we can then define an MCSA scheme that accelerates the MINRES iteration:

$$\alpha = \frac{\langle \mathbf{A}\mathbf{r}^k, \mathbf{r}^k \rangle}{\langle \mathbf{A}\mathbf{r}^k, \mathbf{A}\mathbf{r}^k \rangle} , \tag{2.67a}$$

$$\mathbf{x}^{k+1/2} = \mathbf{x}^k + \alpha \mathbf{r}^k , \tag{2.67b}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} , \tag{2.67c}$$

$$\mathbf{A}\delta\mathbf{x}^{k+1/2} = \mathbf{r}^{k+1/2} , \tag{2.67d}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2} , \tag{2.67e}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} . \tag{2.67f}$$

where α is an optimal extrapolation parameter generated from the constraints. Interestingly, this scheme is nearly identical to the Richardson iteration version in Eq (2.65) except now the additional extrapolation parameter, α, is applied to the residual correction in Eq (2.67b) as a means of selecting a more optimal search direction at each iteration, potentially further accelerating convergence.

2.6.2 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.

2.6.2.1 Basic Preconditioning

As an example of basic preconditioning, 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:

$$\mathbf{M} = diag(\mathbf{A}) \,, \tag{2.68}$$

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

$$\mathbf{M}^{-1}\mathbf{A}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b} \,. \tag{2.69}$$

Next, we can apply MCSA to solve Eq (2.69):

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + \mathbf{M}^{-1}\mathbf{r}^{k} \,, \tag{2.70a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \,, \tag{2.70b}$$

$$\mathbf{M}^{-1}\mathbf{A}\delta\mathbf{x}^{k+1/2} = \mathbf{M}^{-1}\mathbf{r}^{k+1/2} \,, \tag{2.70c}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2} \,, \tag{2.70d}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \,, \tag{2.70e}$$

where the Neumann-Ulam 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 Monte Carlo method used within MCSA to compute the correction operates on a linear problem with the preconditioned operator, then $\mathbf{H}$ in the Neumann-Ulam 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.

2.6.2.2 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:

$$\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\mathbf{M}_R\mathbf{x} = \mathbf{M}_L^{-1}\mathbf{b}\,. \tag{2.71}$$

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

$$\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\mathbf{u} = \mathbf{M}_L^{-1}\mathbf{b}\,, \tag{2.72}$$

with

$$\mathbf{x} = \mathbf{M}_R^{-1}\mathbf{u}\,. \tag{2.73}$$

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

$$\mathbf{u}^{k+1/2} = \mathbf{u}^k + \mathbf{r}^k\,, \tag{2.74a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{M}_L^{-1}(\mathbf{b} - \mathbf{A}\mathbf{M}_R^{-1}\mathbf{u}^{k+1/2})\,, \tag{2.74b}$$

$$\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\delta\mathbf{u}^{k+1/2} = \mathbf{r}^{k+1/2}\,, \tag{2.74c}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^{k+1/2} + \delta\mathbf{u}^{k+1/2}\,, \tag{2.74d}$$

$$\mathbf{r}^{k+1} = \mathbf{M}_L^{-1}(\mathbf{b} - \mathbf{A}\mathbf{M}_R^{-1}\mathbf{u}^{k+1})\,, \tag{2.74e}$$

and then recover the original solution vector with Eq (2.73). For the Monte Carlo problem, we isolate the generation of the correction:

$$\mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\delta\mathbf{u}^{k+1/2} = \mathbf{r}^{k+1/2}\,, \tag{2.75}$$

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

$$\mathbf{H} = \mathbf{I} - \mathbf{M}_L^{-1}\mathbf{A}\mathbf{M}_R^{-1}\,. \tag{2.76}$$

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 (2.75) 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 (2.76) 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.

2.7 Monte Carlo Method Selection

The MCSA method defined in Eq. (2.65) uses the adjoint method to estimate the error in a residual Monte Carlo solve instead of the direct method outlined in §2.3. 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 transport problem[3]:

$$\frac{\partial \mathbf{u}}{\partial t} = \nabla^2 \mathbf{u} \,. \tag{2.77}$$

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-

[3]The Poisson equation is simple form of the diffusion equation and has a similar elliptic character in its time-dependent form to the neutron transport equations that will be solved in the next chapter.

order five-point stencil,

$$\nabla_5^2 = \frac{1}{\Delta^2}[u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - 4u_{i,j}] \,, \tag{2.78}$$

and a fourth-order nine-point stencil,

$$\nabla_9^2 = \frac{1}{6\Delta^2}[4u_{i-1,j} + 4u_{i+1,j} + 4u_{i,j-1} + 4u_{i,j+1} + u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{i,j}] \,, \tag{2.79}$$

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:

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{u}^n \,. \tag{2.80}$$

Both the stencils will be used to vary the size and density of the sparse linear system in Eq. (2.80).

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 2.4 gives the CPU time needed for each case to converge in seconds and Figure 2.5 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 the remaining sections of this chapter 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 2.4 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 2.5. 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

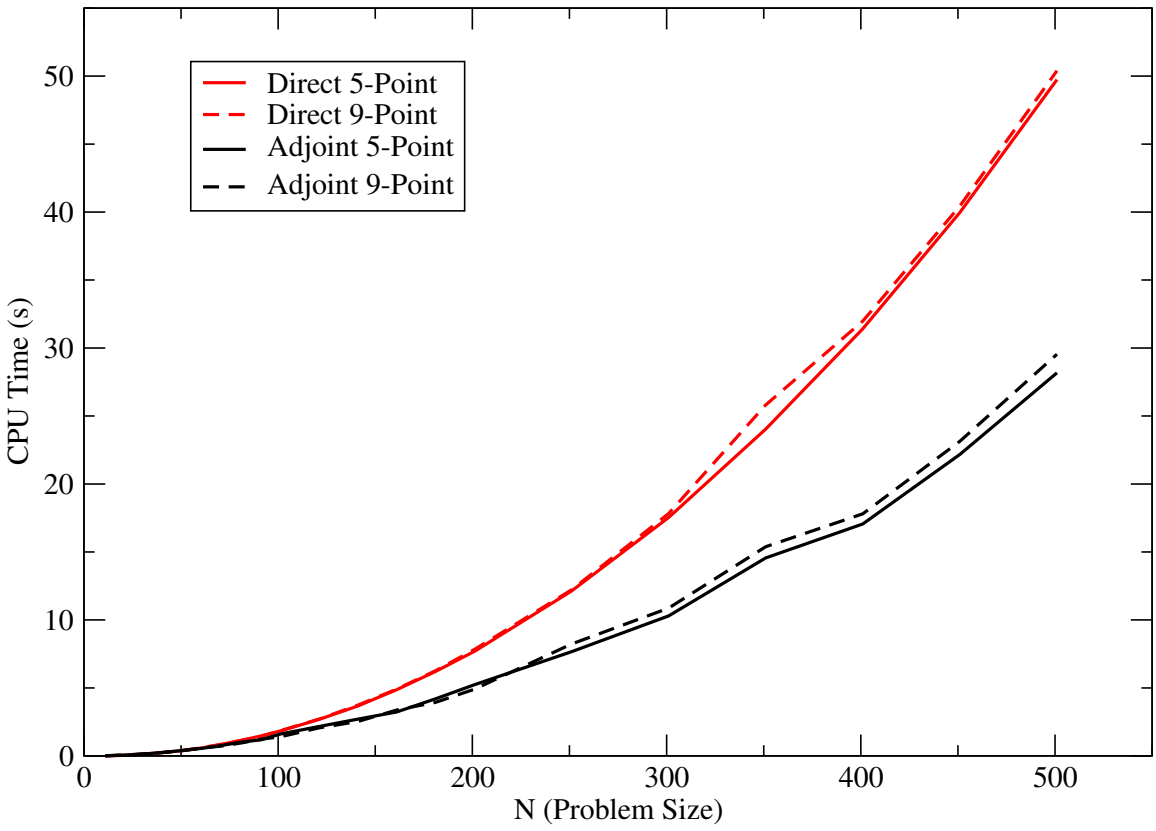

Figure 2.4: **CPU Time (s) to converge vs. Problem Size (N for an $N \times N$ square mesh).** *Both the adjoint and direct solvers are used with the five point and nine point stencils. A CPU time speedup is noted with the adjoint method due to the higher density of random walk events in regions with a large residual.*

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 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 2.5 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

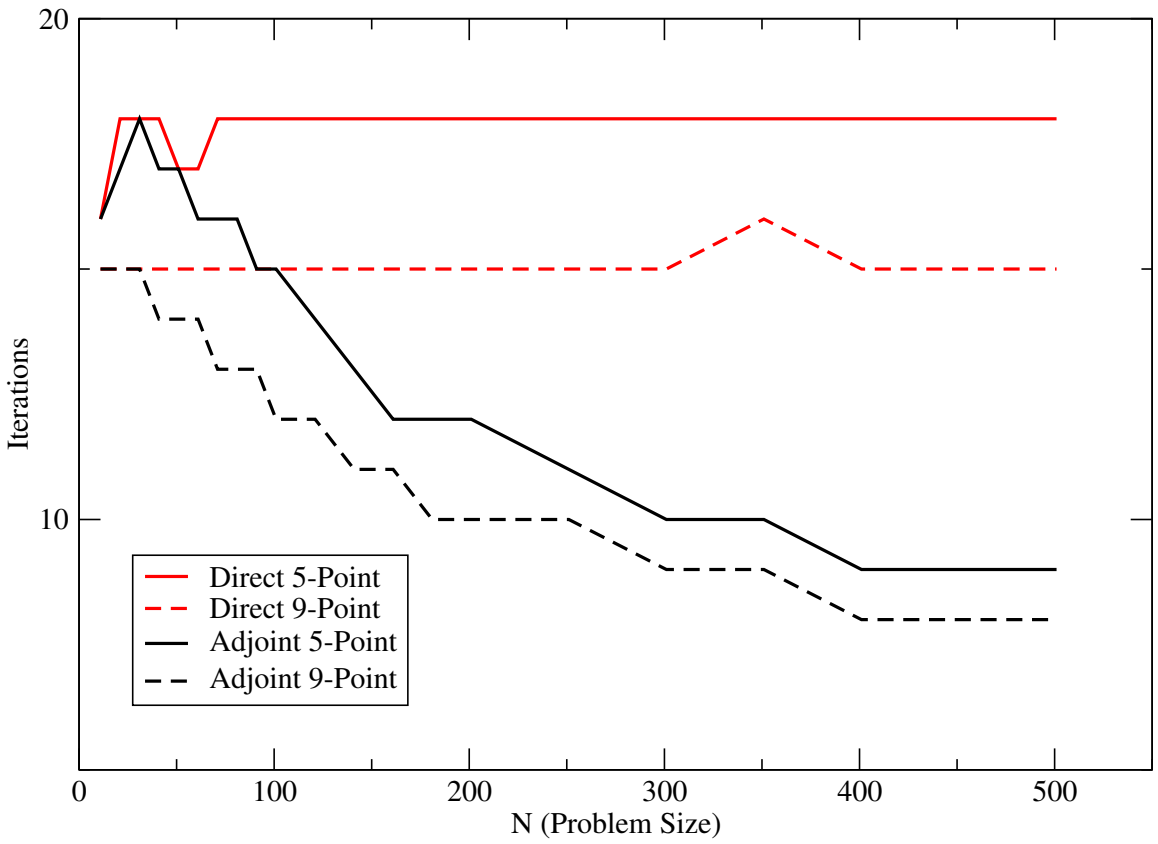

Figure 2.5: **Iterations to converge vs. Problem Size (N for an $N \times N$ square mesh).** *Both the adjoint and direct solvers are used with the five-point and nine-point stencils.*

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.

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. (2.65) was collected at each iteration for a fixed problem size of $N = 500$. Figure 2.6 gives the results of these computations. First, it is worthy to note on the semi-log 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 histories 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

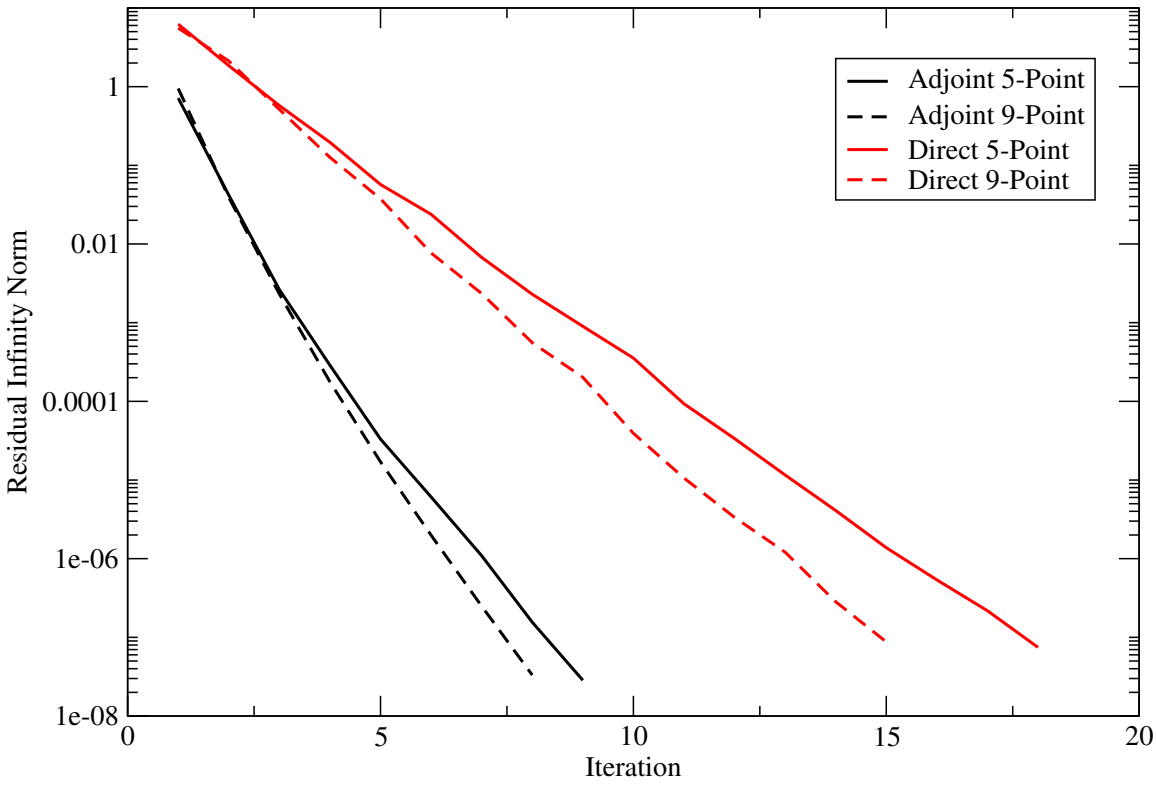

Figure 2.6: **Infinity norm of the solution residual vs. iteration number for a problem of size** $N = 500$**.** *Both the adjoint and direct solvers are used with the five point and nine point stencils. A higher rate of convergence is observed for MCSA using the adjoint Monte Carlo solver as compared to the direct method when both solvers compute the same number of random walks per iteration.*

is used to discretize the Laplacian operator (although at no gain in speed as given by the results in Figure 2.4). This is due to the fact that the smaller discretization error directly corresponds to a more well defined residual source generated by the 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 accurate correction vector from the Monte Carlo calculation.

2.8 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 2.7 gives the CPU time results for this comparison as a function of problem size while Figure 2.8 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 2.9 and 2.10 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

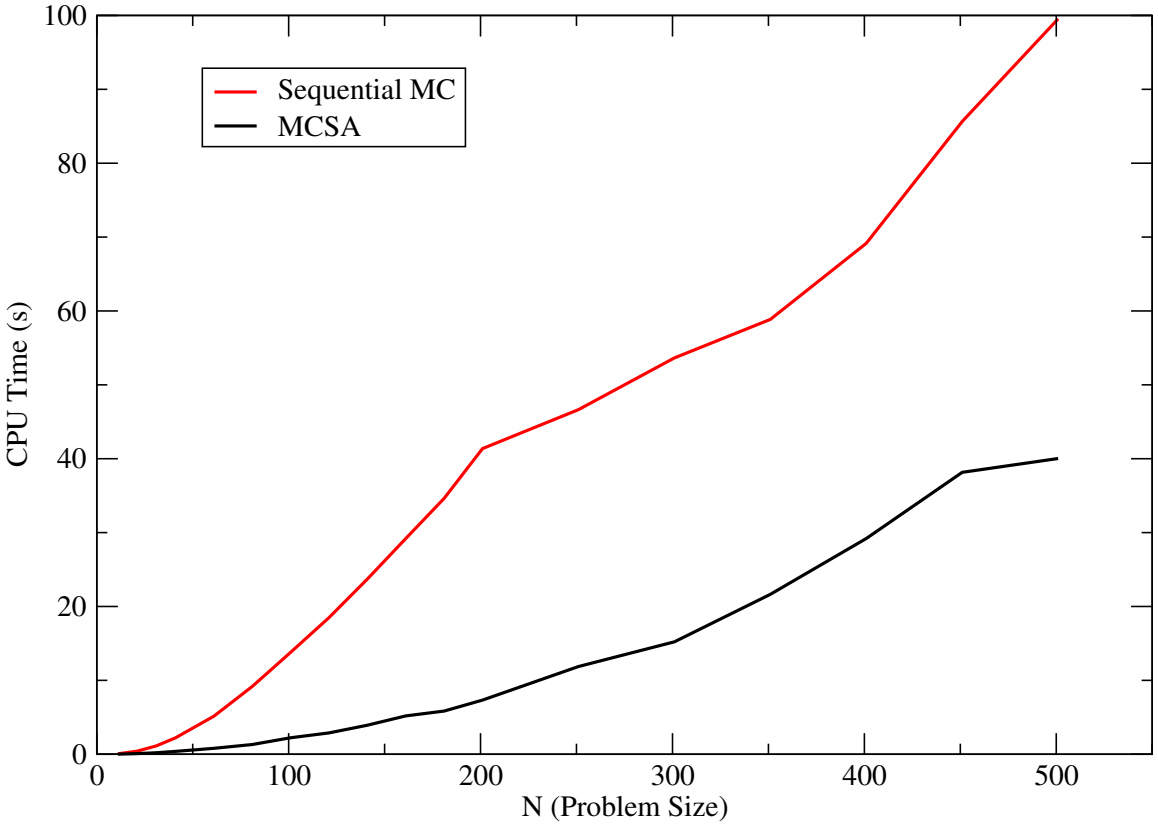

Figure 2.7: **CPU Time (s) to converge vs. Problem Size (N for an $N \times N$ square mesh).** *Both the Sequential Monte Carlo and MCSA solvers are used with the five point stencils and the adjoint Monte Carlo solver. The number of random walks was twice the number of discrete states in the system in 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.

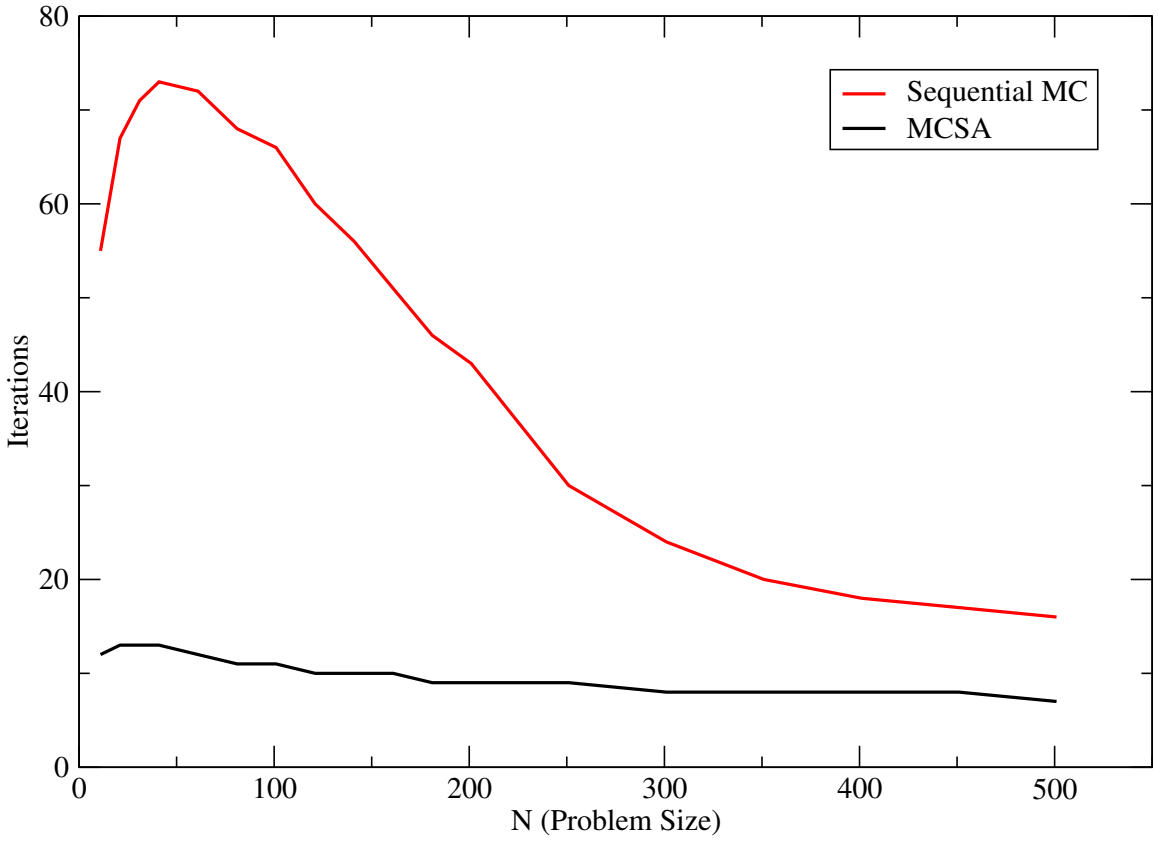

Figure 2.8: **Iterations to converge vs. Problem Size (N for an $N \times N$ square mesh).** *Both the Sequential Monte Carlo and MCSA solvers are used with the five point stencils and the adjoint Monte Carlo solver.*

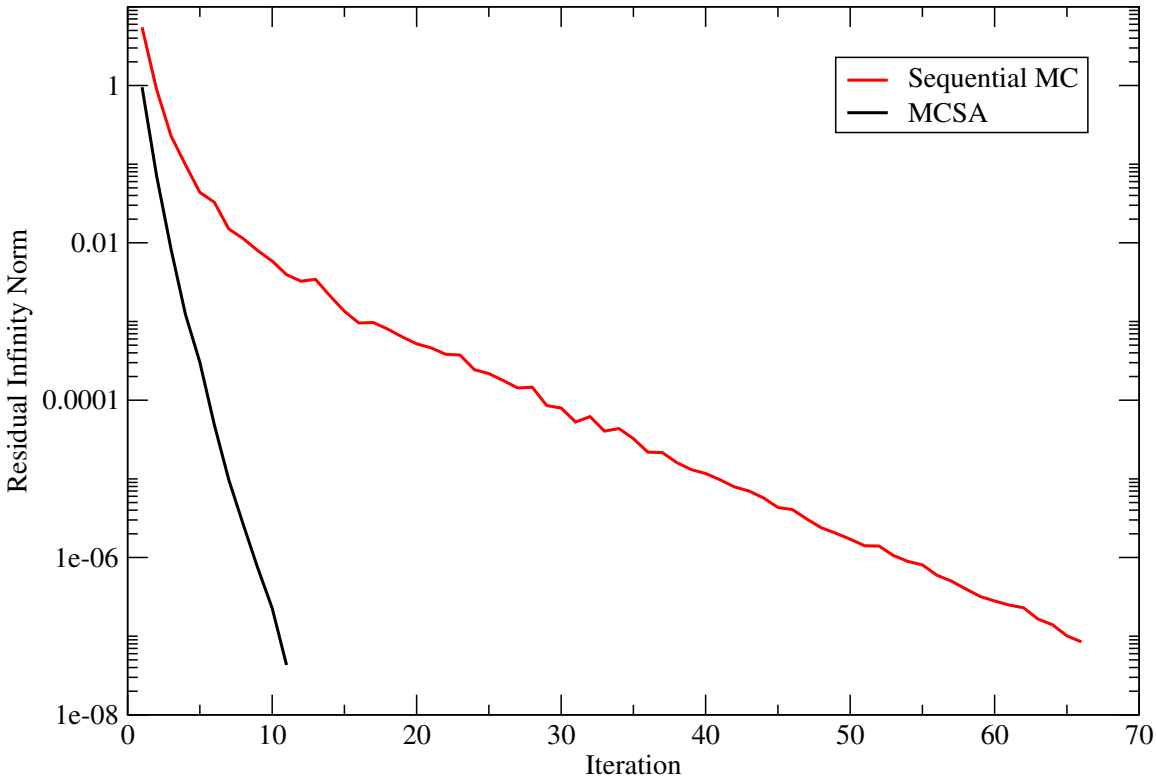

Figure 2.9: **Infinity norm of the solution residual vs. iteration number for a problem of size** $N = 100$**.** *Both the Sequential Monte Carlo and MCSA solvers are used with the five point stencils and the adjoint Monte Carlo solver.*

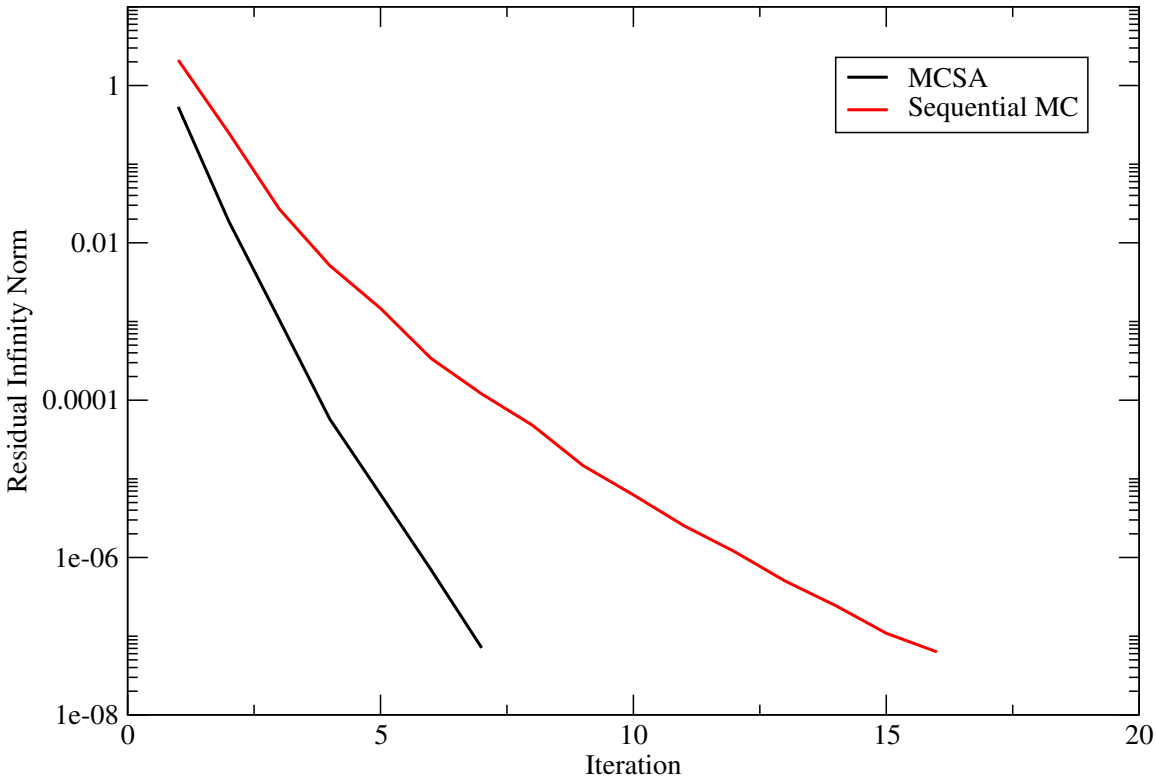

Figure 2.10: **Infinity norm of the solution residual vs. iteration number for a problem of size** $N = 500$**.** *Both the Sequential Monte Carlo and MCSA solvers are used with the five point stencils and the adjoint Monte Carlo solver.*

2.9 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 2.11 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). Interestingly, 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 2.12 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 2.13 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

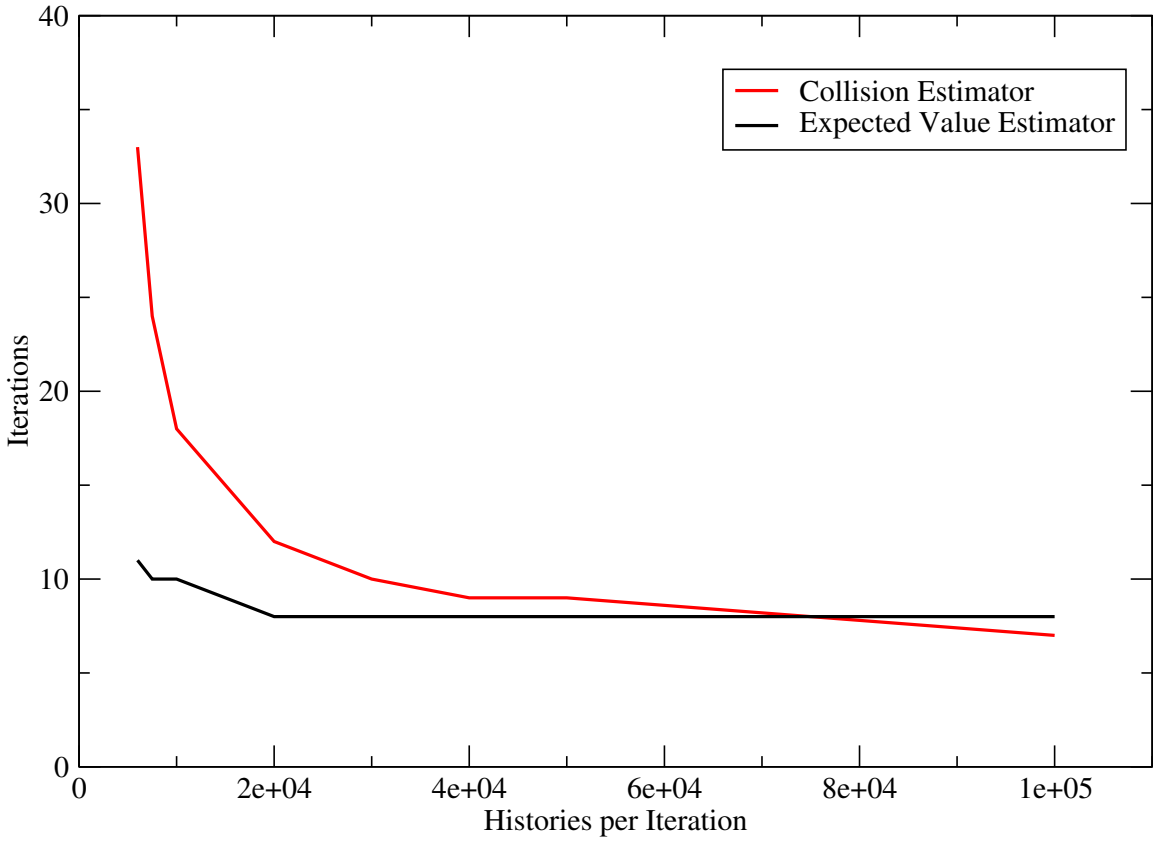

Figure 2.11: **Iterations (s) to converge vs. Monte Carlo histories per MCSA iteration for a 200×200 square mesh and a weight cutoff of 1×10^{-4}.** *For low numbers of histories, the expected value estimator performance is significantly better than the collision estimator. At higher numbers of histories, the estimators become roughly equivalent.*

more important to MCSA convergence than those that occur later and with lower weight contributions. To further motivate using a larger weight cutoff, Figure 2.14 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.

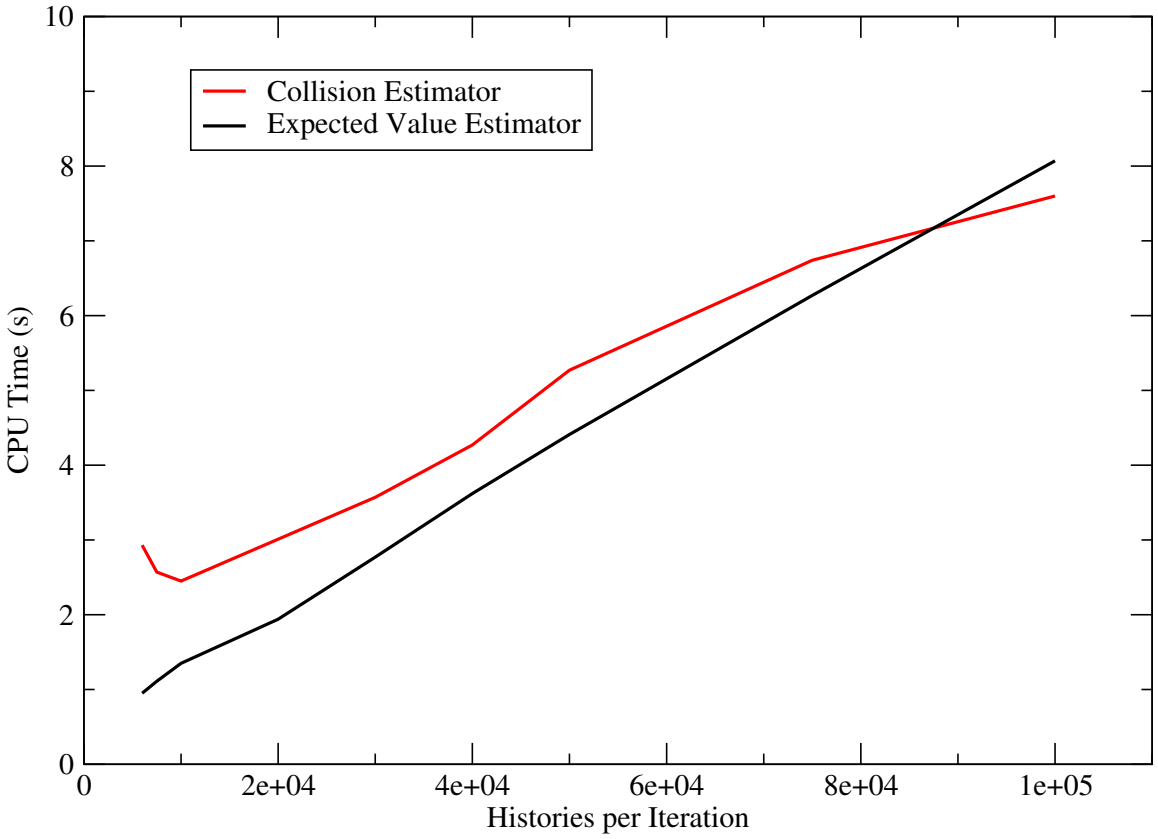

Figure 2.12: **CPU Time (s) to converge vs. Monte Carlo histories per MCSA iteration for a 200×200 square mesh and a weight cutoff of 1×10^{-4}.** *For low numbers of histories, the expected value estimator performance is better than the collision estimator due to a lower iteration count while the actual compute time per iteration is higher.*

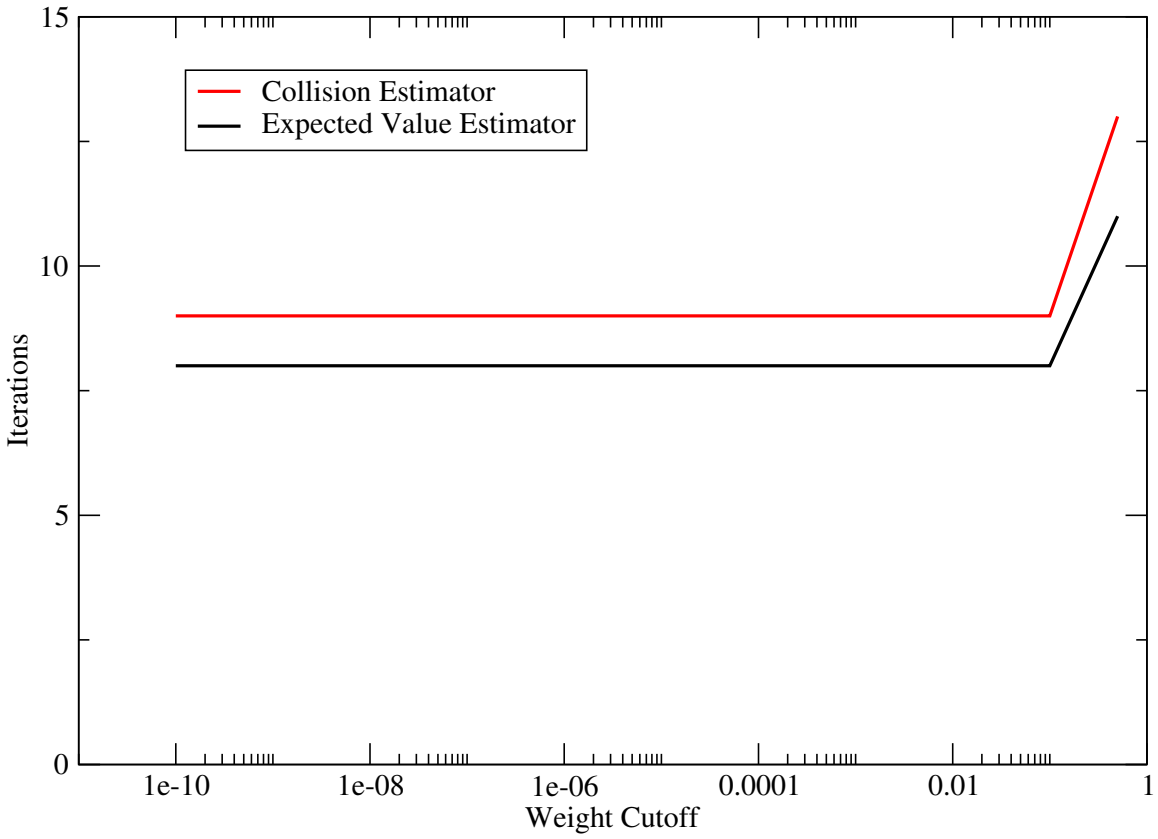

Figure 2.13: **Iterations (s) to converge vs. history weight cutoff for a** 200×200 **square mesh and 40,000 histories.** *For low numbers of histories, the expected value estimator performance is significantly better than the collision estimator. At higher numbers of histories, the estimators become roughly equivalent.*

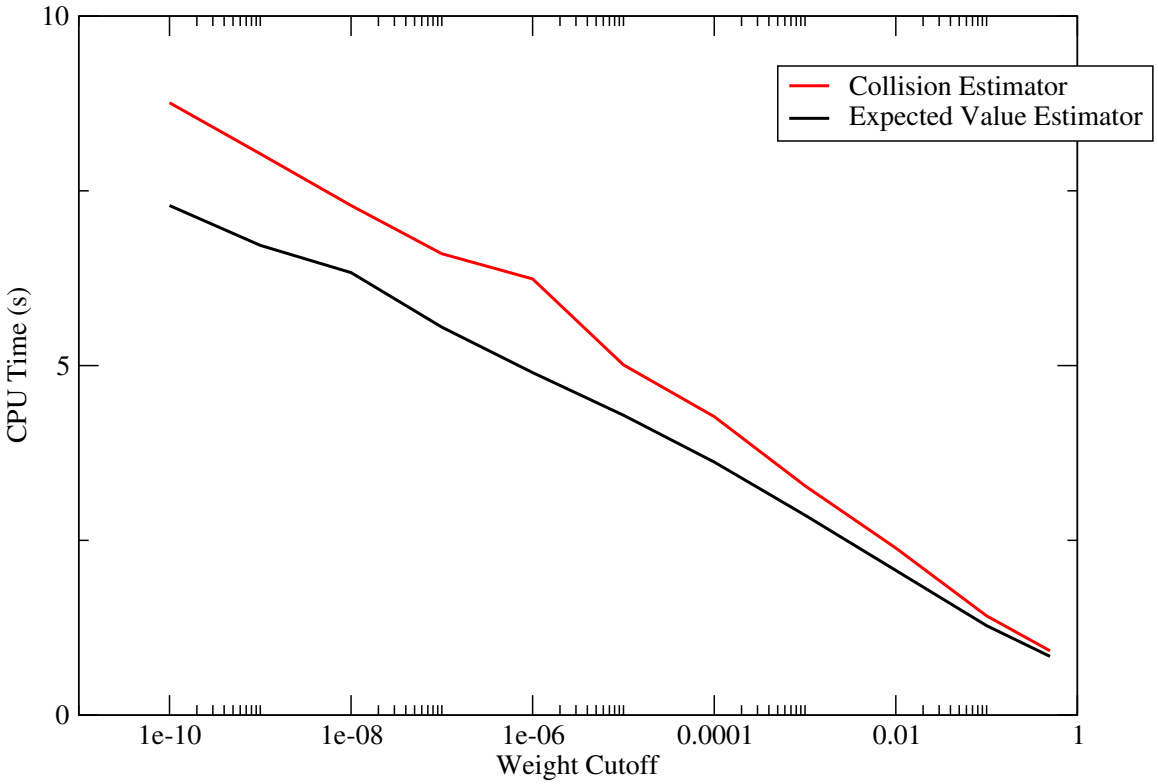

Figure 2.14: **CPU Time (s) to converge vs. history weight cutoff for a 200×200 square mesh and 40,000 histories.** *For low numbers of histories, the expected value estimator performance is better than the collision estimator due to a lower iteration count while the actual compute time per iteration is higher.*

2.10 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. For this work, any method that alters the weights and probabilities of the Monte Carlo method with the aim of improving either iterative performance or time to convergence will be defined as a variance reduction scheme.

2.10.1 Artificial Absorption

The random walk sequences generated by the adjoint Neumann Ulam decomposition in Eqs (2.44) and (2.45) 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 *artificial 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:

$$p_{ij} = \frac{|h_{ji}|}{\sum_j |h_{ji}|}(1 - p_{abs}) \,, \tag{2.81}$$

and

$$w_{ij} = \frac{sign(h_{ji})}{(1 - p_{abs})} \sum_j |h_{ji}| \,. \tag{2.82}$$

At each transition step, a history at state i will then sample the probability distribution function generated by Eq (2.81) 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 significantly more likely that a history will be terminated by absorption rather than weight cutoff. Figure 2.15 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 significantly 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 2.16 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.

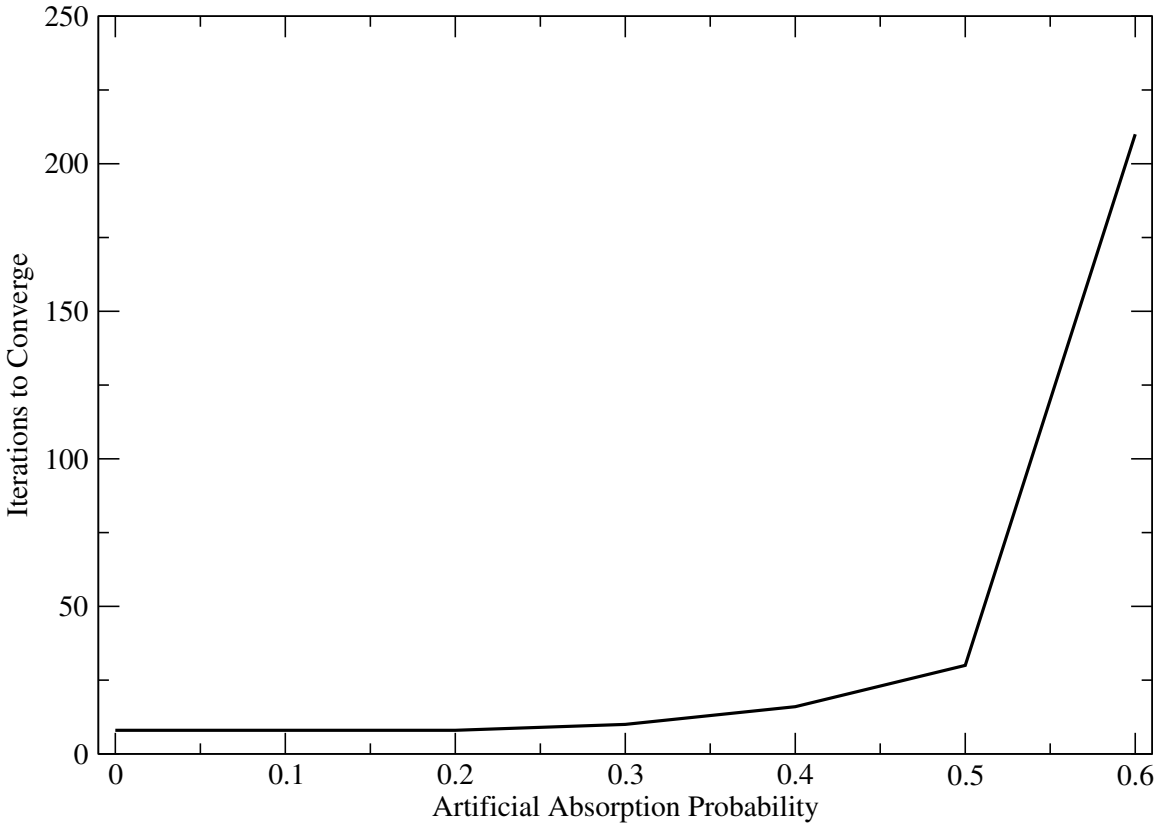

Figure 2.15: **Iterations (s) to converge vs. artificial absorption probability for a 200×200 square mesh and 40,000 histories.** *The addition of absorption is detrimental to iterative performance due to both the shortening of the random walks as well as the modification of the transition weight.*

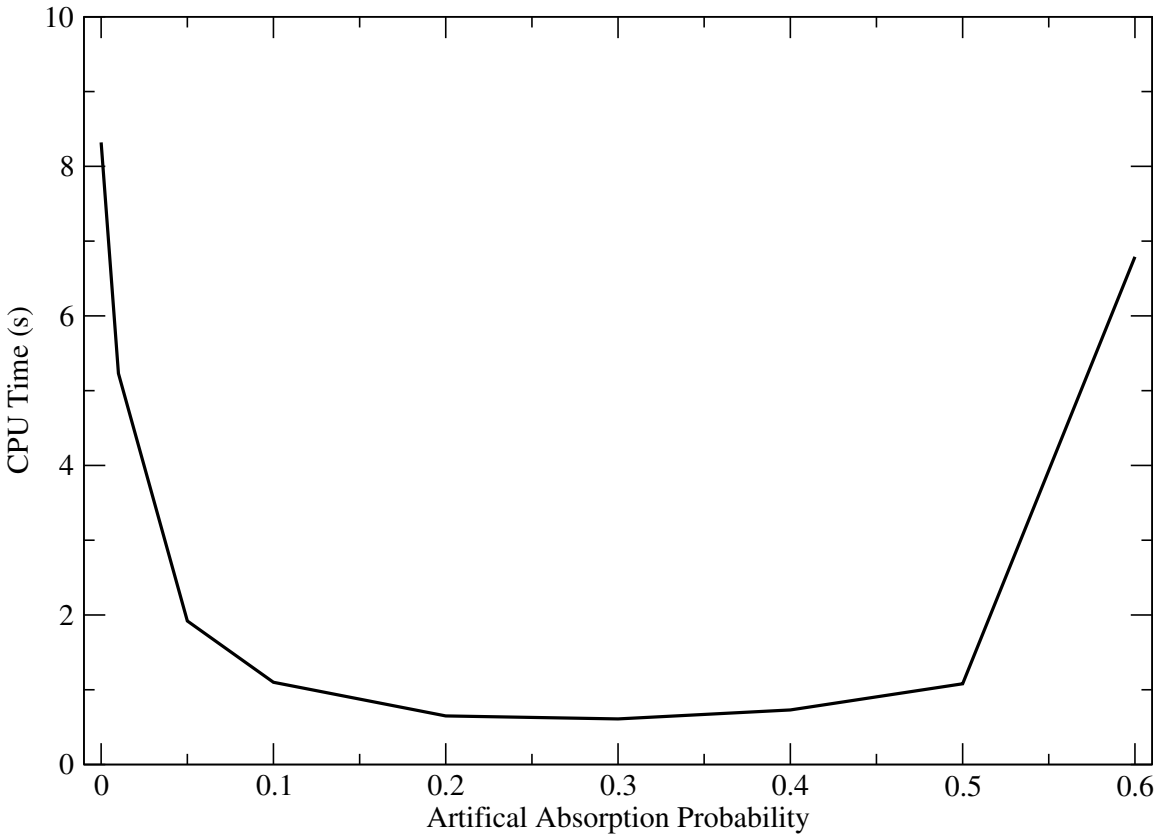

Figure 2.16: **CPU Time (s) to converge vs. artificial absorption probability for a** 200×200 **square mesh and 40,000 histories.** *The reduced random walk length speeds up the calculation with a fixed number of histories. Too much artificial absorption increases iterations rapidly giving an increasing CPU time.*

2.10.2 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 (2.75) 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:

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + \mathbf{r}^{k} \,, \tag{2.83a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \,, \tag{2.83b}$$

$$\hat{\mathbf{A}}\delta\mathbf{x}^{k+1/2} = \mathbf{r}^{k+1/2} \,, \tag{2.83c}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta\mathbf{x}^{k+1/2} \,, \tag{2.83d}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \,, \tag{2.83e}$$

where $\hat{\mathbf{A}}$ in Eq (2.83c) 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 neutron transport systems where it is required due to the explicit preconditioning strategy will be presented in Chapter 3.

Chapter 3

Monte Carlo Synthetic Acceleration Methods for the SP_N Equations

In this chapter, we briefly derive the simplified P_N (SP_N) equations, closely following the work of Evans (Evans, 2013)[1], in order to gain full understanding of the underlying system and its potential behavior in a Monte Carlo context. From the P_N equations, we apply a set of approximations to yield the SP_N equations for fixed source and criticality problems. Using the fully-formed linear operator for the transport problem, we explore solutions to the SP_N equations with Monte Carlo Synthetic Acceleration using a light water reactor fuel assembly criticality calculation as the driving problem. Several difficulties arise when applying MCSA to these problems that were not observed when investigating the simpler transport systems.

For Jacobi-based preconditioners, convergence with MCSA is difficult if not impossible for ill-conditioned systems with this behavior demonstrated using a simple neutron diffusion problem. In order to effectively solve the SP_N equations for the fuel assembly problem, a suite of preconditioners along with a relaxation scheme is developed and studied within the context of MCSA. Using these preconditioners, several additional issues are observed and alleviated to a certain extent by applying the reduced domain approximation. MCSA solutions for the fuel assembly problem are then verified by comparing the solutions against production Krylov solvers for problems with varying energy groups. Finally performance is analyzed through comparison with those same Krylov solvers in terms of both iterative performance and CPU timing using the same set of problems utilized for the verification.

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

3.1 The Neutron Transport Equation

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

$$\hat{\Omega} \cdot \vec{\nabla}\psi(\vec{r}, \hat{\Omega}, E) + \sigma(\vec{r}, E)\psi(\vec{r}, \hat{\Omega}, E) = \iint \sigma_s(\vec{r}, E' \to E, \hat{\Omega}' \cdot \hat{\Omega})\psi(\vec{r}, \hat{\Omega}', E')d\Omega' dE' + q(\vec{r}, \hat{\Omega}, E)\,, \quad (3.1)$$

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
- $\sigma_s(\vec{r}, E' \to E, \hat{\Omega}')$ - 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 (3.1) in 1-dimensional Cartesian geometry:

$$\mu\frac{\partial}{\partial x}\psi(x, \mu, E) + \sigma(x, E)\psi(x, \mu, E) = \iint \sigma_s(x, E' \to E, \hat{\Omega}' \cdot \hat{\Omega})\psi(x, \hat{\Omega}', E')d\Omega' dE' + \frac{q(x, E)}{4\pi}\,, \quad (3.2)$$

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:

$$\hat{\Omega}\cdot\vec{\nabla}\psi(\vec{r},\hat{\Omega},E)+\sigma(\vec{r},E)\psi(\vec{r},\hat{\Omega},E)= \iint\sigma_s(\vec{r},E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega})\psi(\vec{r},\hat{\Omega}',E')d\Omega'dE'+ \frac{1}{k}\chi(E)\iint\nu\sigma_f(\hat{r},E')\psi(\vec{r},\hat{\Omega}',E')d\Omega'dE'+q(\vec{r},\hat{\Omega},E)\,, \quad (3.3)$$

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 (3.3) becomes:

$$\mu\frac{\partial}{\partial x}\psi(x,\mu,E)+\sigma(x,E)\psi(x,\mu,E)= \iint\sigma_s(x,E'\rightarrow E,\hat{\Omega}'\cdot\hat{\Omega})\psi(x,\hat{\Omega}',E')d\Omega'dE'+ \frac{1}{k}\chi(E)\iint\nu\sigma_f(x,E')\psi(x,\hat{\Omega}',E')d\Omega'dE'+\frac{q(x,E)}{4\pi}\,. \quad (3.4)$$

3.2 Derivation of the Monoenergetic SP_N Equations

The P_N equations as derived in Appendix B 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:

$$\phi_n = \frac{1}{\Sigma_n}\left[q\delta_{no} - \frac{\partial}{\partial x}\left(\frac{n}{2n+1}\phi_{n-1} + \frac{n+1}{2n+1}\phi_{n+1}\right)\right], \tag{3.5}$$

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

$$\begin{aligned} -\frac{\partial}{\partial x}\Bigg[\frac{n}{2n+1}\frac{1}{\Sigma_{n-1}}\frac{\partial}{\partial x}\left(\frac{n-1}{2n-1}\phi_{n-2} + \frac{n}{2n-1}\phi_n\right) \\ + \frac{n+1}{2n+1}\frac{1}{\Sigma_{n+1}}\frac{\partial}{\partial x}\left(\frac{n+1}{2n+3}\phi_n + \frac{n+2}{2n+3}\phi_{n+2}\right)\Bigg] \\ + \Sigma_n\phi_n = q\delta_{n0} \qquad n = 0, 2, 4, \cdots, N\,. \end{aligned} \tag{3.6}$$

Immediately, we note the diffusion-like nature of Eq (3.6) 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:

$$\begin{aligned} -\nabla\cdot\Bigg[\frac{n}{2n+1}\frac{1}{\Sigma_{n-1}}\nabla\left(\frac{n-1}{2n-1}\phi_{n-2} + \frac{n}{2n-1}\phi_n\right) \\ + \frac{n+1}{2n+1}\frac{1}{\Sigma_{n+1}}\nabla\left(\frac{n+1}{2n+3}\phi_n + \frac{n+2}{2n+3}\phi_{n+2}\right)\Bigg] \\ + \Sigma_n\phi_n = q\delta_{n0} \qquad n = 0, 2, 4, \cdots, N\,, \end{aligned} \tag{3.7}$$

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:

$$-\nabla \cdot \frac{1}{3\Sigma_1}\nabla(\phi_0 + 2\phi_2) + \Sigma_0\phi_0 = q \tag{3.8a}$$

$$-\nabla \cdot \left[\frac{2}{15\Sigma_1}\nabla(\phi_0 + 2\phi_2) + \frac{3}{35\Sigma_3}\nabla(3\phi_2 + 4\phi_4)\right] + \Sigma_2\phi_2 = 0 \tag{3.8b}$$

$$-\nabla \cdot \left[\frac{4}{63\Sigma_3}\nabla(3\phi_2 + 4\phi_4) + \frac{5}{99\Sigma_5}\nabla(5\phi_4 + 6\phi_6)\right] + \Sigma_4\phi_4 = 0 \tag{3.8c}$$

$$-\nabla \cdot \left[\frac{6}{143\Sigma_5}\nabla(5\phi_4 + 6\phi_6) + \frac{7}{195\Sigma_7}\nabla(7\phi_6)\right] + \Sigma_6\phi_6 = 0\,. \tag{3.8d}$$

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:

$$u_1 = \phi_0 + 2\phi_2 \tag{3.9a}$$

$$u_2 = 3\phi_2 + 4\phi_4 \tag{3.9b}$$

$$u_3 = 5\phi_4 + 6\phi_6 \tag{3.9c}$$

$$u_4 = 7\phi_6\,. \tag{3.9d}$$

When substituted into Eq (3.8), these terms give:

$$-\nabla \cdot \frac{1}{3\Sigma_1}\nabla u_1 + \Sigma_0\left[u_1 - \frac{2}{3}u_2 + \frac{8}{15}u_3 - \frac{16}{35}u_4\right] = -q \tag{3.10a}$$

$$-\nabla \cdot \left[\frac{2}{15\Sigma_1}\nabla u_1 + \frac{3}{35\Sigma_3}\nabla u_2\right] + \Sigma_2\left[\frac{1}{3}u_2 - \frac{4}{15}u_3 + \frac{8}{35}u_4\right] = 0 \tag{3.10b}$$

$$-\nabla \cdot \left[\frac{4}{63\Sigma_3}\nabla u_2 + \frac{5}{99\Sigma_5}\nabla u_3\right] + \Sigma_4\left[\frac{1}{5}u_3 - \frac{6}{35}u_4\right] = 0 \tag{3.10c}$$

$$-\nabla \cdot \left[\frac{6}{143\Sigma_5}\nabla u_3 + \frac{7}{195\Sigma_7}\nabla u_4\right] + \Sigma_6\left[\frac{1}{7}u_4\right] = 0\,. \tag{3.10d}$$

If we rearrange Eq (3.11) 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:

$$-\nabla \cdot D_n\nabla u_n + \sum_{m=1}^{4} A_{nm}u_m = q_n \qquad n = 1, 2, 3, 4\,, \tag{3.11}$$

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

$$\mathbf{u} = (u_1 \;\; u_2 \;\; u_3 \;\; u_4)^T \,, \tag{3.12}$$

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

$$\mathbf{D} = \left(\frac{1}{3\Sigma_1} \;\; \frac{1}{7\Sigma_3} \;\; \frac{1}{11\Sigma_5} \;\; \frac{1}{15\Sigma_7} \right)^T \,, \tag{3.13}$$

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

$$\mathbf{q} = (q \;\; -\frac{2}{3}q \;\; \frac{8}{15}q \;\; -\frac{16}{35}q)^T \,, \tag{3.14}$$

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

$$\mathbf{A} = \begin{bmatrix} (\Sigma_0) & (-\frac{2}{3}\Sigma_0) & (\frac{8}{15}\Sigma_0) & (-\frac{16}{35}\Sigma_0) \\ (-\frac{2}{3}\Sigma_0) & (\frac{4}{9}\Sigma_0 + \frac{5}{9}\Sigma_2) & (-\frac{16}{45}\Sigma_0 - \frac{4}{9}\Sigma_2) & (\frac{32}{105}\Sigma_0 + \frac{8}{21}\Sigma_2) \\ (\frac{8}{15}\Sigma_0) & (-\frac{16}{45}\Sigma_0 - \frac{4}{9}\Sigma_2) & (\frac{64}{225}\Sigma_0 + \frac{16}{45}\Sigma_2 + \frac{9}{25}\Sigma_4) & (-\frac{128}{525}\Sigma_0 - \frac{32}{105}\Sigma_2 - \frac{54}{175}\Sigma_4) \\ (-\frac{16}{35}\Sigma_0) & (\frac{32}{105}\Sigma_0 + \frac{8}{21}\Sigma_2) & (-\frac{128}{525}\Sigma_0 - \frac{32}{105}\Sigma_2 - \frac{54}{175}\Sigma_4) & (\frac{256}{1225}\Sigma_0 + \frac{64}{245}\Sigma_2 + \frac{324}{1225}\Sigma_4 + \frac{13}{49}\Sigma_6) \end{bmatrix} . \tag{3.15}$$

Note that the term $\sum_{m=1}^{4} A_{nm} u_m$ in Eq (3.12) 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 D, the multigroup form is presented in Appendix C, and the spatial discretization in Appendix E.

3.2.1 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 form of the transport equation given by Eq (3.3). We can readily replace the fixed source in Eq (C.7) with the fission

source derived in § B.4:

$$
\begin{aligned}
-\nabla\cdot\Bigg[&\frac{n}{2n+1}\boldsymbol{\Sigma}_{\mathbf{n-1}}{}^{-1}\nabla\Big(\frac{n-1}{2n-1}\boldsymbol{\Phi}_{\mathbf{n-2}}+\frac{n}{2n-1}\boldsymbol{\Phi}_{\mathbf{n}}\Big)\\
&+\frac{n+1}{2n+1}\boldsymbol{\Sigma}_{\mathbf{n+1}}{}^{-1}\nabla\Big(\frac{n+1}{2n+3}\boldsymbol{\Phi}_{\mathbf{n}}+\frac{n+2}{2n+3}\boldsymbol{\Phi}_{\mathbf{n+2}}\Big)\Bigg]\\
&+\boldsymbol{\Sigma}_{\mathbf{n}}\boldsymbol{\Phi}_{\mathbf{n}}=\frac{1}{k}\mathbf{F}\boldsymbol{\Phi}_{\mathbf{n}}\delta_{n0}\qquad n=0,2,4,\cdots,N\,. \quad (3.16)
\end{aligned}
$$

with the fission matrix $\mathbf{F}$ given by Eq (B.46). We again apply the change of variables to yield the set of multigroup 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:

$$
-\nabla\cdot\mathbb{D}_n\nabla\mathbb{U}_n+\sum_{m=1}^{4}\mathbb{A}_{nm}\mathbb{U}_m=\frac{1}{k}\sum_{m=1}^{4}\mathbb{F}_{nm}\mathbb{U}_n\qquad n=1,2,3,4\,. \quad (3.17)
$$

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.

3.3 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 2, 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 3.1 were fixed for each spatial location and energy group. Mesh element sizes were the same in all cardinal directions with 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 3.1 while for the upscatter case all groups could scatter to all

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

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 (C.8). Figure 3.1 gives the sparsity pattern for a monoenergetic SP_7 discretization while Figures 3.2 and 3.3 give the sparsity patterns for the same problem for 10 energy groups without and with upscatter respectively. For these figures, the parameters in Table 3.1 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 3.2), 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 Eq (C.9) are readily apparent. Our first attempt at preconditioning this system will be to use the point Jacobi preconditioning from §2.6 due to this diagonal form.

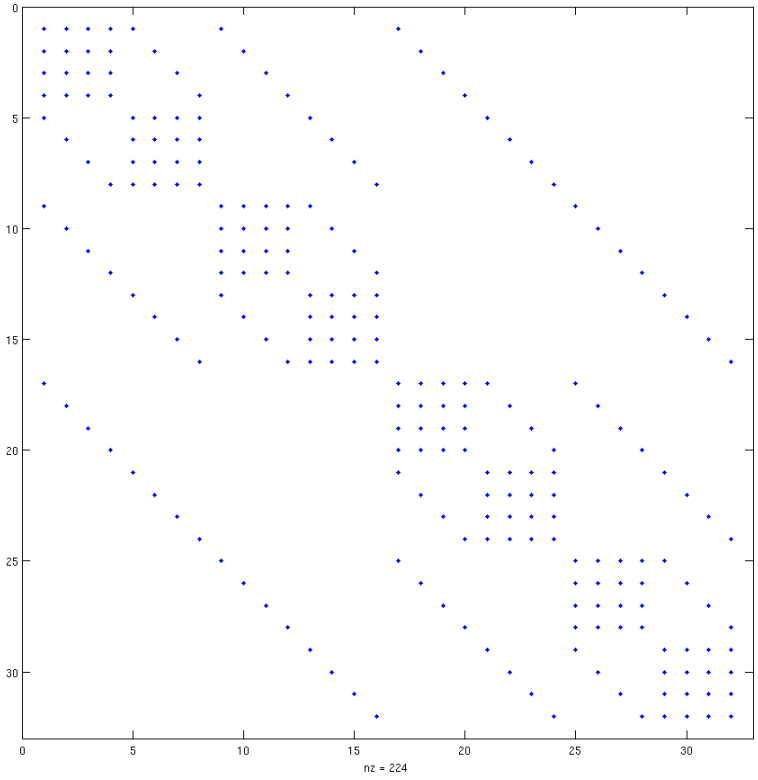

Figure 3.1: **Sparsity pattern for 1-group SP_7 discretization.** *A $2\times2\times2$ element mesh was used to show detail of the blocks formed by the discretization.*

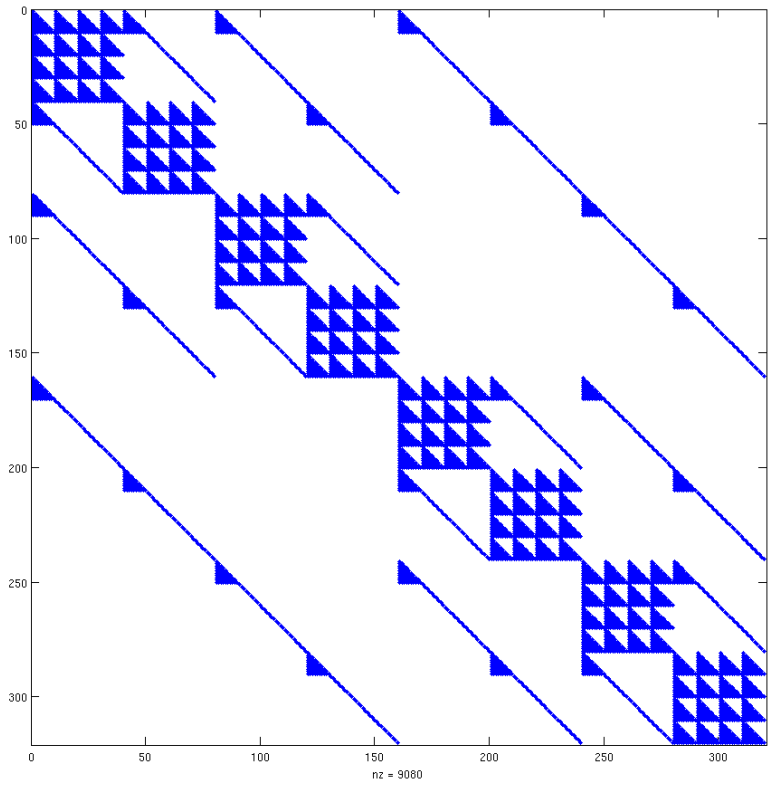

Figure 3.2: **Sparsity pattern for 10-group SP_7 discretization with downscatter only.** *A $2 \times 2 \times 2$ element mesh was used to show detail of the blocks formed by the discretization.*

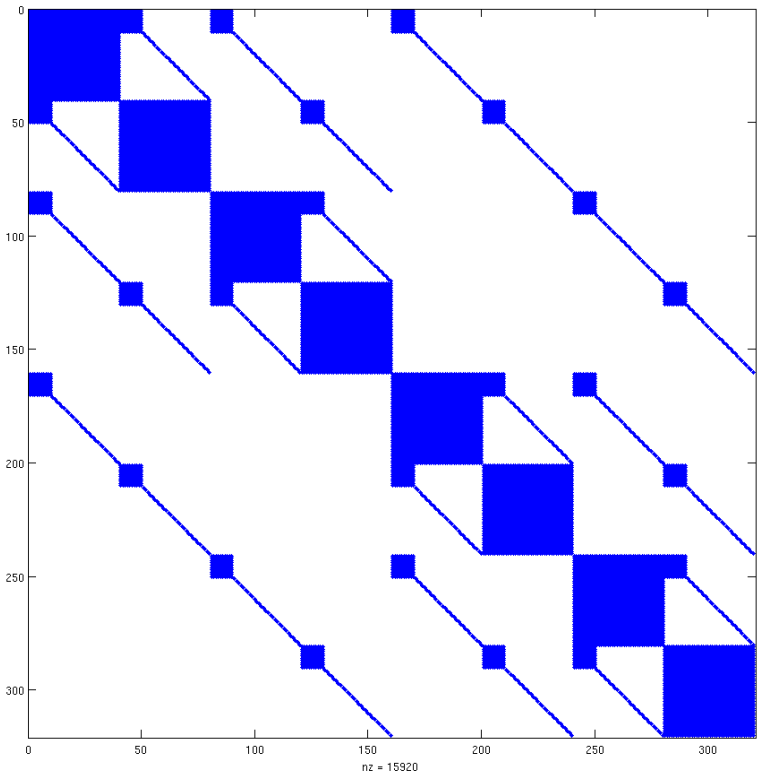

Figure 3.3: **Sparsity pattern for 10-group SP_7 discretization with downscatter and upscatter.** *A $2 \times 2 \times 2$ element mesh was used to show detail of the blocks formed by the discretization.*

3.3.1 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 3.2 gives the results for the 1-group case, Table 3.3 for the 10-group case with full downscatter only and Table 3.4 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 3.2: **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 3.3: **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.

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

3.3.2 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 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 3.1, 3.2 and 3.3 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 (C.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 3.2.

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 3.4. Shown on the right side of Figure 3.4, inversion of the preconditioner is trivial with each diagonal block inverted separately. For the SP_N equations, Eq (C.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 3.4. 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 (C.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 transport problems meaning that established dense matrix methods can be used for fast inversion.

3.3.3 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 3.5

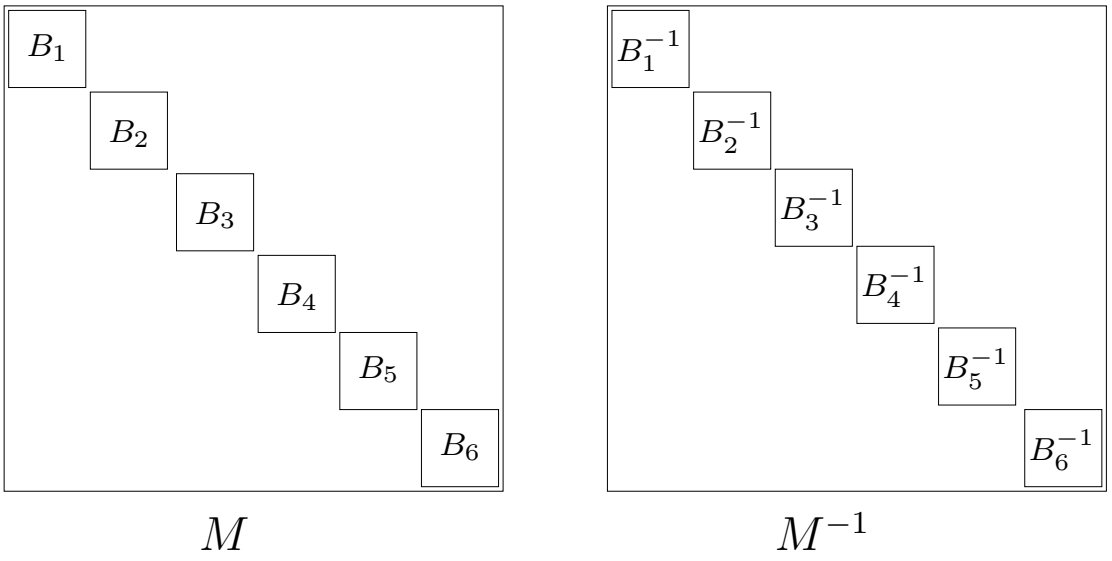

Figure 3.4: **Block Jacobi preconditioning strategy used for the SP_N equations.** *Left: The preconditioner is formed by the diagonal blocks of the matrix. Right: Inversion of the preconditioner is trivial and decoupled by block.*

gives the results for the 1-group case, Table 3.6 for the 10-group case with full downscatter only and Table 3.7 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.

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

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

		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 3.6: **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 3.7: **Spectral radius results for the block Jacobi preconditioned iteration matrix with 10 energy groups, full downscatter and full upscatter.**

strategy applied to more complicated problems representative of physical systems.

3.4 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 3.5 while a cross section of the materials configuration along the horizontal axis is given by Figure 3.6. A detailed view of the assembly bottom is given in Figure 3.7. On the top and bottom of the assembly, vacuum conditions are used as well as on the top and right boundaries in Figure 3.5. Reflecting conditions are used on the left and bottom boundaries of Figure 3.5, effectively giving a representation of one quarter of the assembly. For the spatial discretization, each fuel pin (gray regions in Figure 3.5) is resolved by a 2×2 mesh with materials and cross sections homogenized over this region.

Significant geometric details are 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 3.8 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 5.1 presents the use of MCSA within a power iteration strategy to find the multiplication factor. Here, $\mathbf{M}$ is the

Algorithm 3.1 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
  $\mathbf{M}\mathbf{\Phi}^{n+1} = \frac{1}{k^n}\mathbf{F}\mathbf{\Phi}^n$ ▷ Solve for the new flux state with MCSA
  $k^{n+1} = k^n \frac{\int \mathbf{F}\mathbf{\Phi}^{n+1} d\mathbf{r}}{\int \mathbf{F}\mathbf{\Phi}^n d\mathbf{r}}$ ▷ Update the multiplication factor
  $n = n + 1$
end while

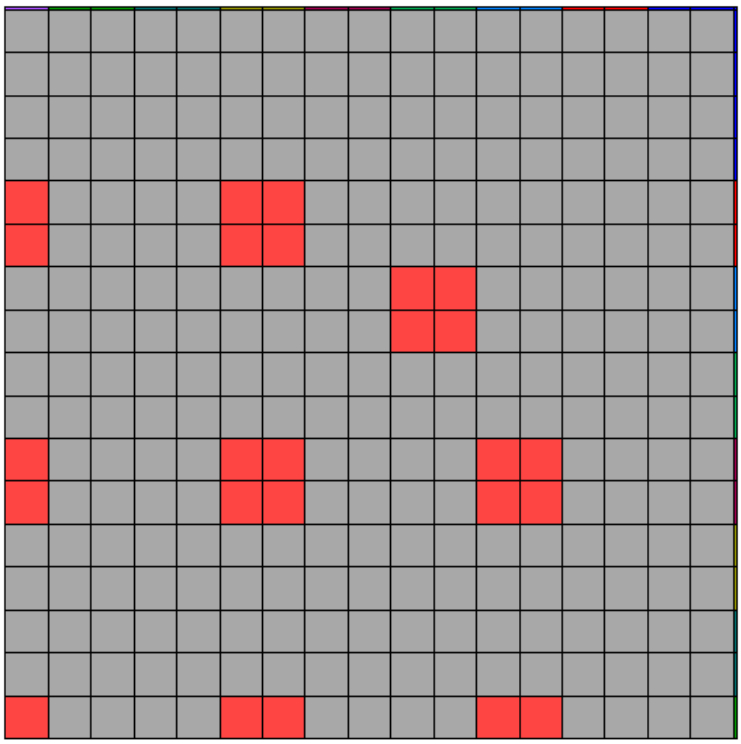

Figure 3.5: **Fuel assembly mesh and geometry cross section.** *Reflecting boundaries are used on the left and lower boundaries to create a complete* 17×17 *assembly geometry. Gray regions are homogenized fuel and red regions are homogenized moderator. Each fuel pin is resolved by a* 2×2 *mesh.*

Figure 3.6: **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.*

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.

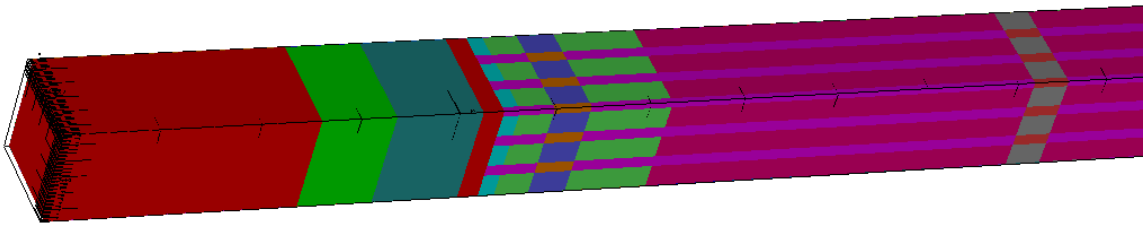

Figure 3.7: **Fuel assembly geometry end detail.** *Reactor core structure including spacer grids and plenum has been included. Lighter purple material on the right of the figure is moderator and darker purple/red material is fuel.*

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

When using MCSA, the linear operator applied to $\mathbf{\Phi}^{n+1}$ at each eigenvalue iteration 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.

[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 $\mathbf{M}$.

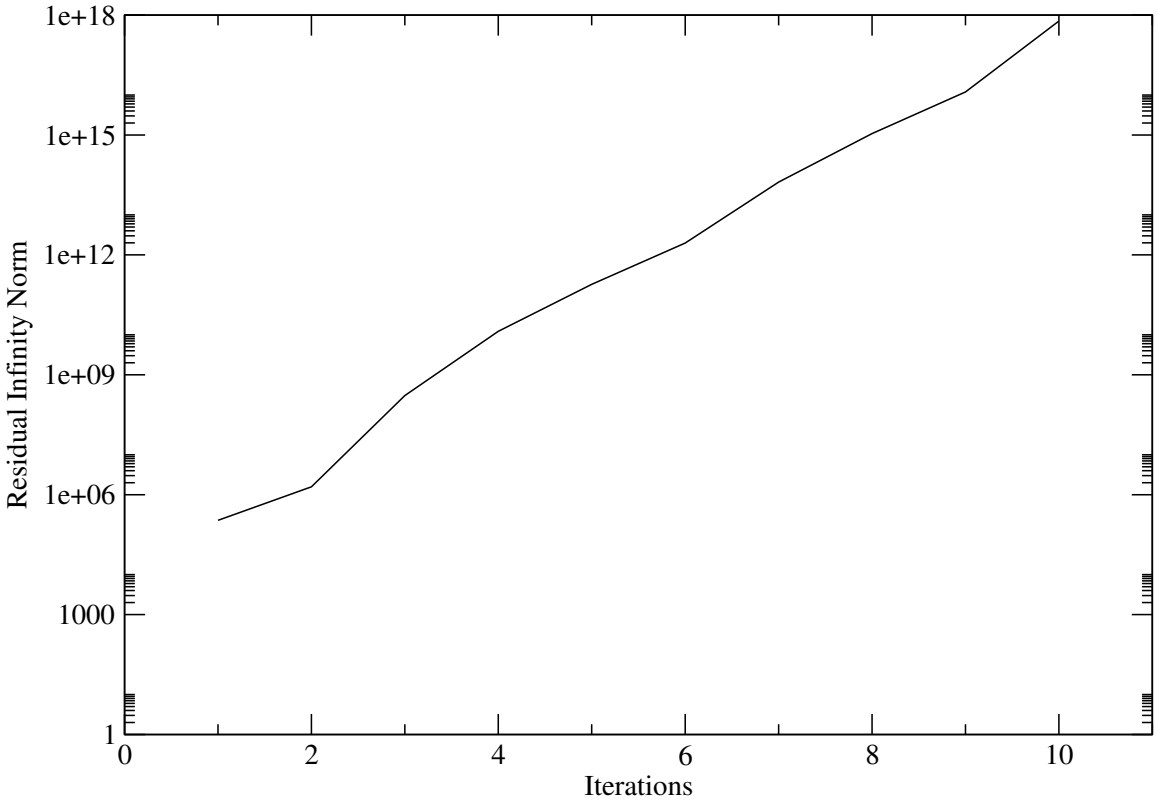

Figure 3.8: **Residual infinity norm vs. iteration for the block Jacobi preconditioned MCSA solve during the first eigenvalue iteration of the 1-group 17×17 fuel assembly problem.** *Convergence was not achieved with the block Jacobi preconditioned method.*

3.4.1 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 3.8 gives the residual infinity norm as a function of iteration for the MCSA linear solve in the first eigenvalue iteration 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

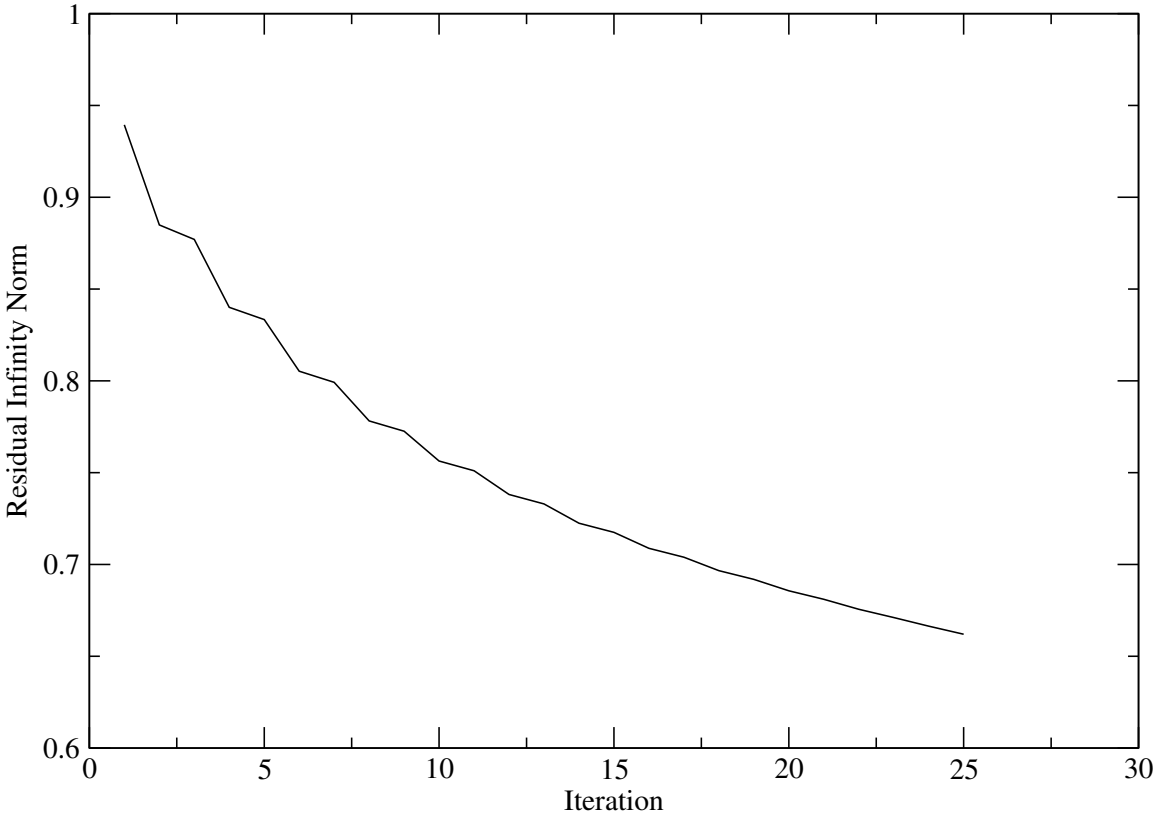

Figure 3.9: **Residual infinity norm vs. iteration for the block Jacobi preconditioned Richardson solve during the first eigenvalue iteration of the 1-group 17×17 fuel assembly problem.** *A spectral radius near 1 was observed for the iteration matrix.*

do expect convergence of MCSA. To investigate further, a block Jacobi preconditioned Richardson iteration was used to solve the same problem. Figure 3.9 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 eigenvalue 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 (A.10). 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.

3.4.2 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 the simpler homogeneous 2-dimensional one-speed neutron diffusion system presented in Appendix **??** 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.

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 3.11 gives the number of iterations required to converge for the chosen number of histories per iteration given by Figure 3.12 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

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

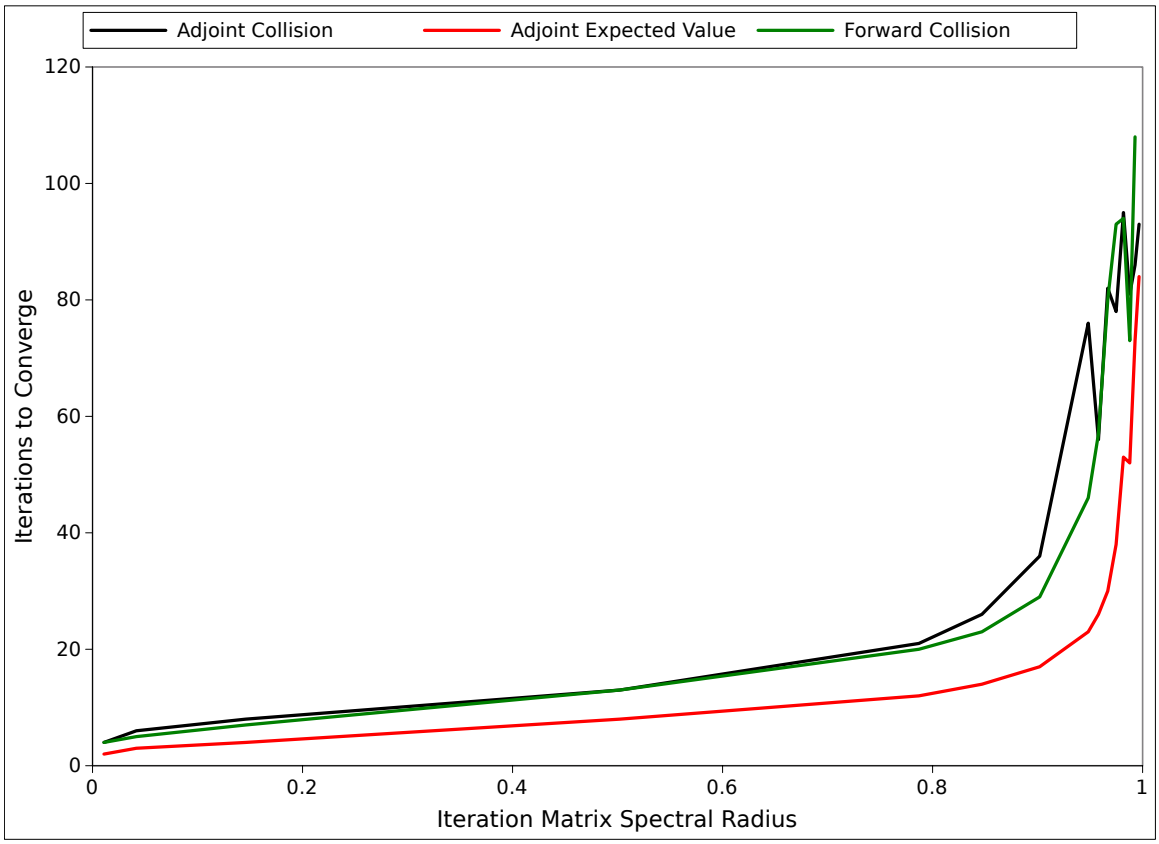

Figure 3.10: **Iterations required to converge as a function of spectral radius for the neutron diffusion problem.** *The number of histories was increased to achieve convergence in less than 100 iterations. At least 10,000 histories were used for each calculation.*

spectral radius approaches 1, the number of histories required becomes significant 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 to achieve convergence for ill-conditioned problems another penalty is paid due to histories that take longer to compute. Figure 3.13 gives the CPU time in seconds required to

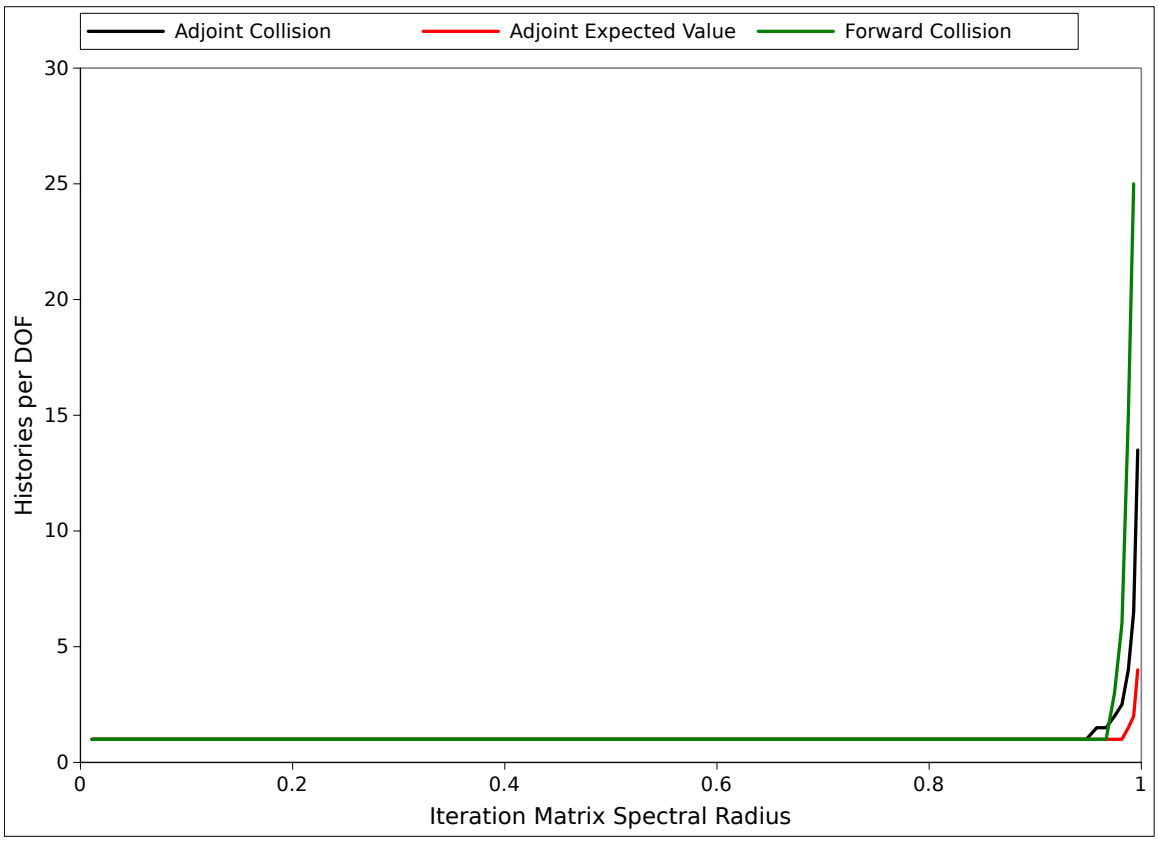

Figure 3.11: **Histories per DOF required to converge as a function of spectral radius for the neutron diffusion problem.** *The number of histories was increased to achieve convergence in less than 100 iterations. At least 10,000 histories were used for each calculation with 10,000 DOFs in the problem.*

compute a single random walk averaged over the entire set of histories run in the 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.

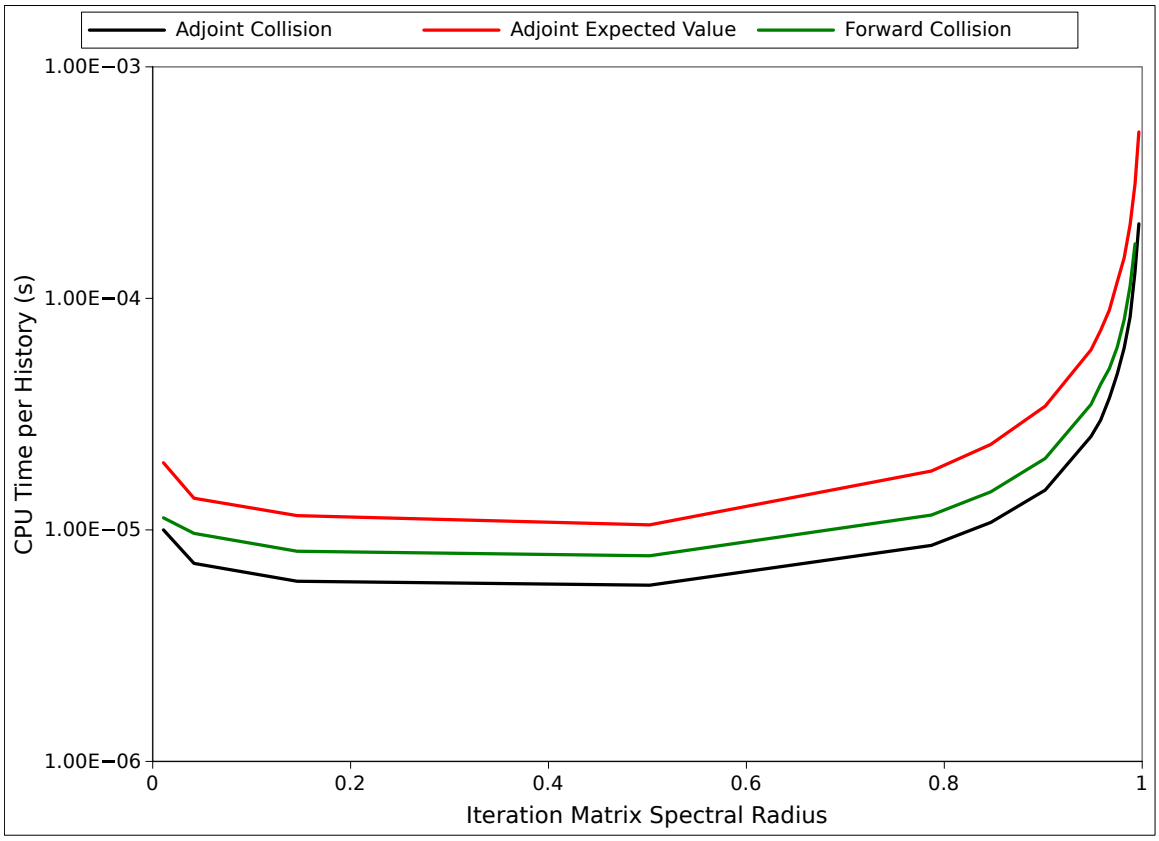

Figure 3.12: **CPU time per history as a function of spectral radius for the neutron diffusion problem.** *As the spectral radius grows, so do the length of the random walks. Longer random walks require more CPU time to compute.*

3.5 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 (2.74). Data showing their effects on MCSA solutions of the fuel assembly criticality problem will be presented.

3.5.1 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):

$$\mathbf{R} = \mathbf{LU} - \mathbf{A}\,, \tag{3.18}$$

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 (2.74), 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 3.14 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

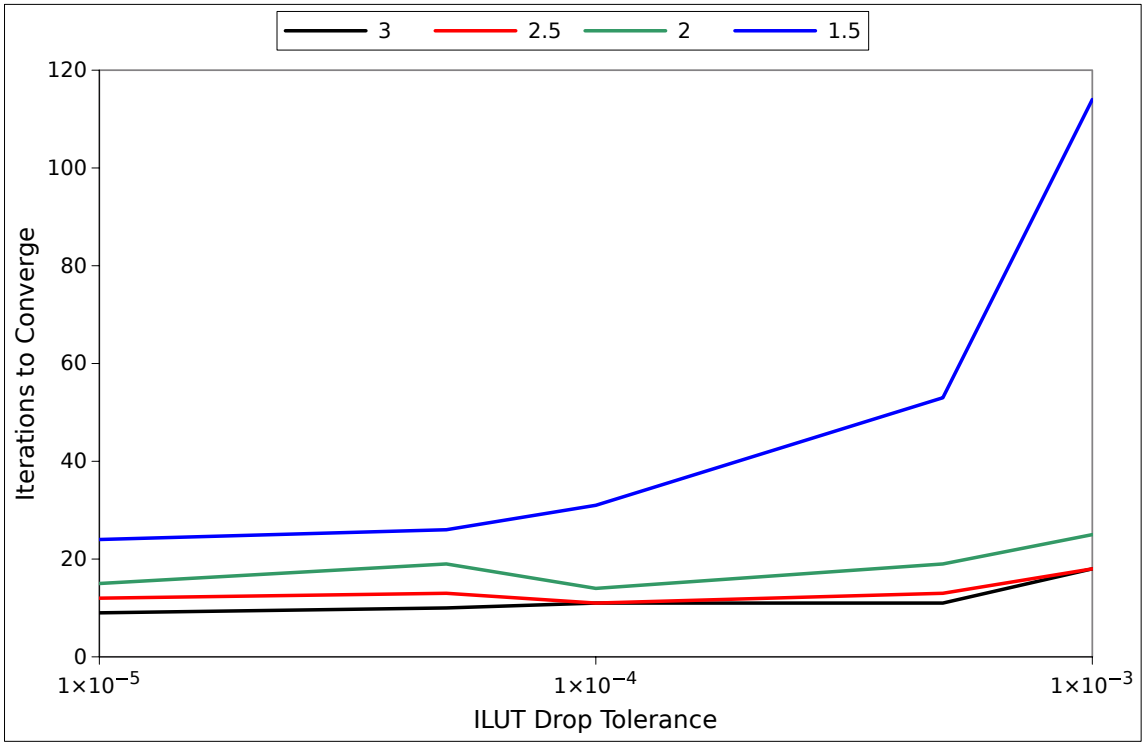

Figure 3.13: **Number of MCSA iterations required to converge an eigenvalue iteration for the fuel assembly problem with ILUT preconditioning as a function of ILUT drop tolerance.** *Each colored curve represents the iteration behavior for a different ILUT fill level. Fill levels of 1.5, 2.0, 2.5, and 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 3.15 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 3.15, 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

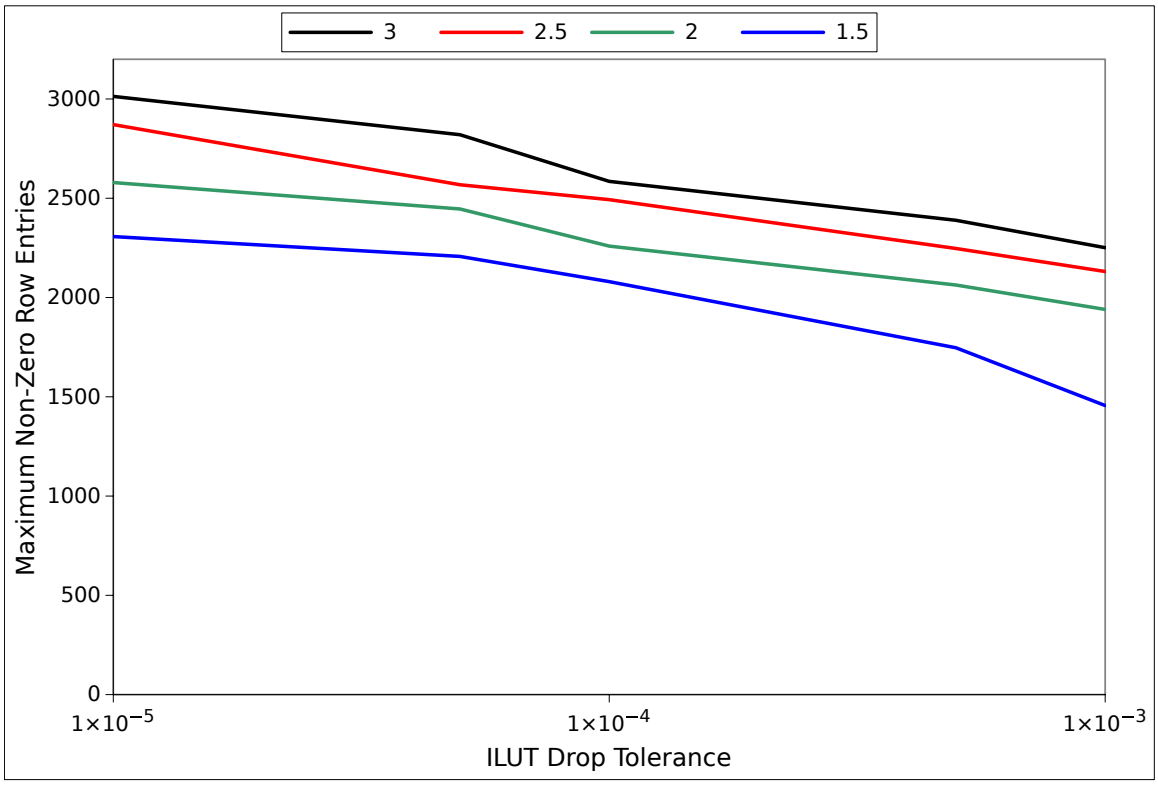

Figure 3.14: **Maximum number of non-zero entries observed for all rows in the composite linear operator for the fuel assembly problem with ILUT preconditioning given as a function of ILUT drop tolerance.** *Each colored curve represents the row size for a different ILUT fill level. Fill levels of 1.5, 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. Second, 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

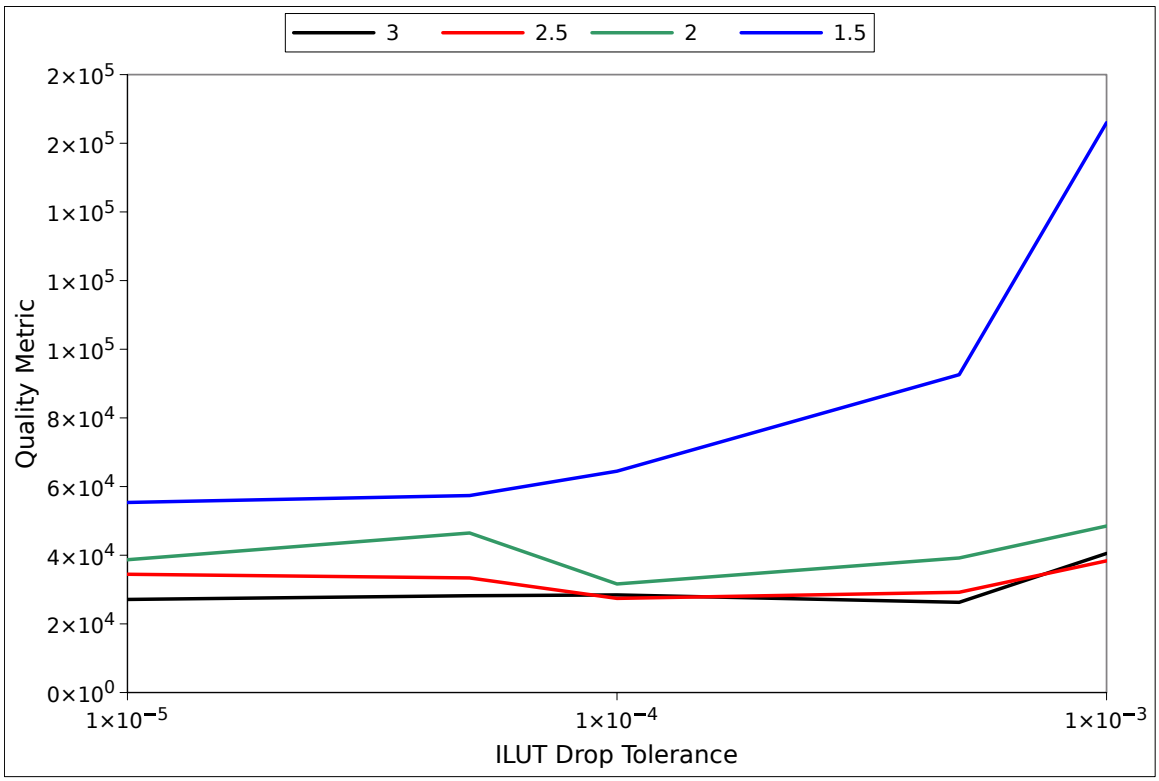

Figure 3.15: **ILUT preconditioning quality metric for the fuel assembly problem given as a function of ILUT drop tolerance.** *Each colored curve represents the quality metric behavior for a different ILUT fill level. Fill levels of 1.5, 2.0, 2.5, and 3.0 were used.*

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:

$$\text{Quality Metric} = (\#\text{ iterations}) \times (\text{maximum } \#\text{ of non-zero values}) \,, \tag{3.19}$$

where the highest quality preconditioning is one that minimizes this metric. For the ILUT preconditioned data provided, Figure 3.16 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 $||\mathbf{b}||_1 \neq 0$.

3.5.2 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):

$$||\mathbf{I} - \mathbf{AM}||_F^2 = \sum_{j=1}^{N} ||\mathbf{e}_j - \mathbf{Am}_j||_2^2 \,, \tag{3.20}$$

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 (3.28), 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 3.17 gives the number of MCSA iterations needed to converge the fuel assembly problem and Figure 3.18 the maximum number of non-zero entries per row observed 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 3.19 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.

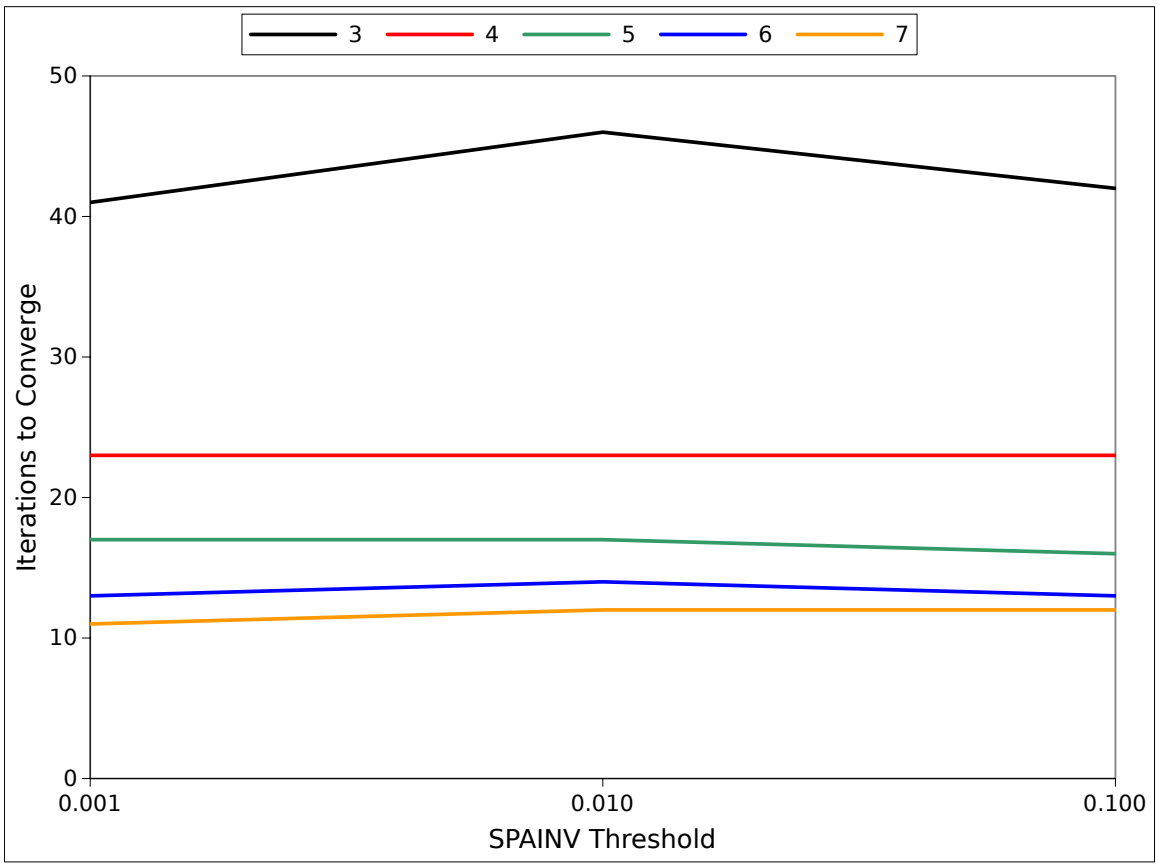

Figure 3.16: **Number of MCSA iterations required to converge a single eigenvalue iteration for the fuel assembly problem with SPAINV preconditioning as a function of SPAINV threshold.** *Each colored curve represents the iteration behavior for a different SPAINV level pattern. Levels of 3, 4, 5, 6, and 7 were used.*

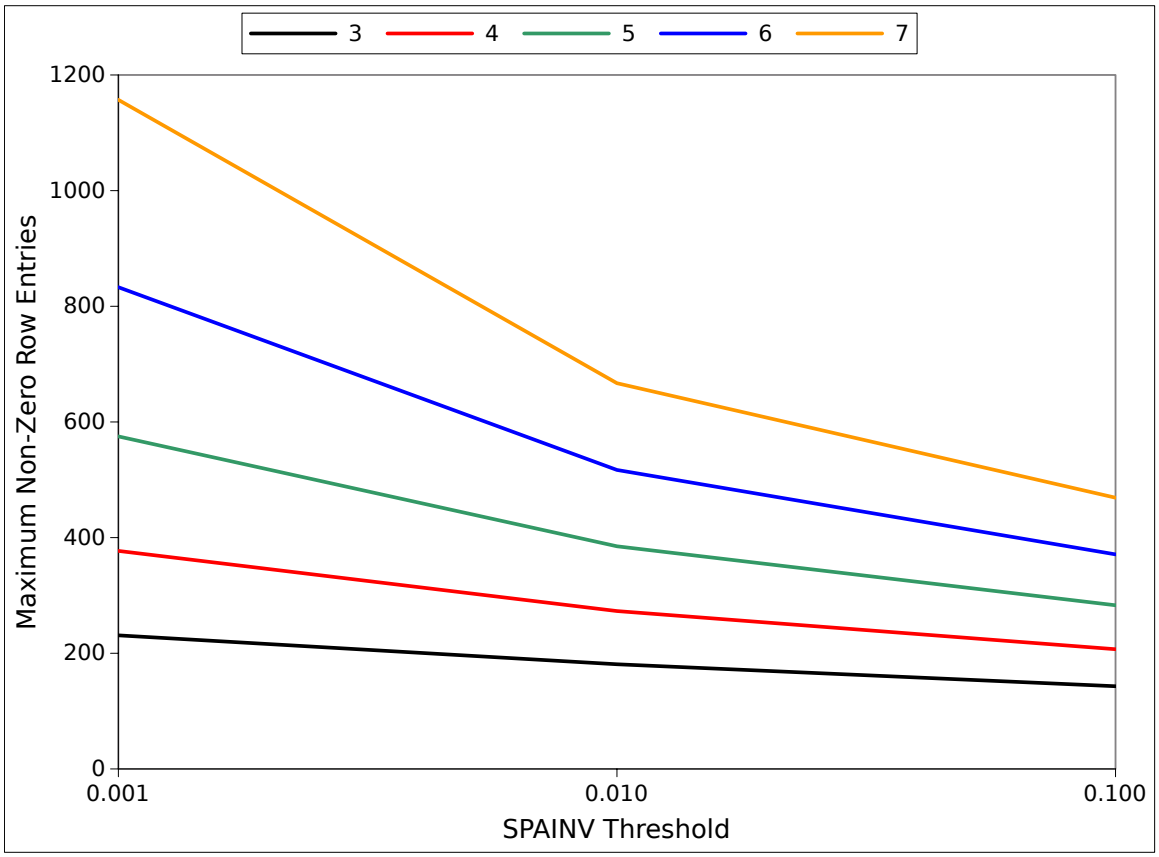

Figure 3.17: **Maximum number of non-zero entries observed for all rows in the composite linear operator for the fuel assembly problem with SPAINV preconditioning given as a function of SPAINV threshold.** *Each colored curve represents the row size for a different SPAINV level pattern. Levels of 3, 4, 5, 6, and 7 were used.*

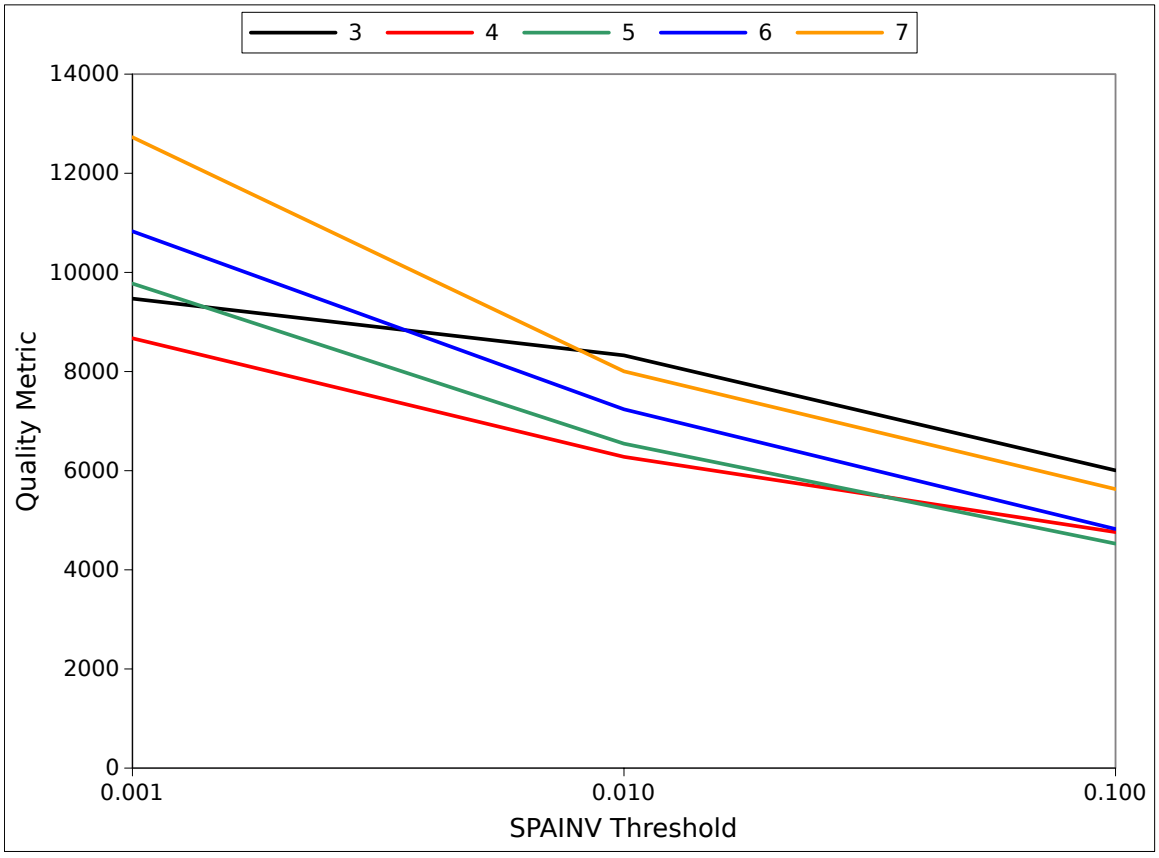

Figure 3.18: **SPAINV preconditioning quality metric for the fuel assembly problem given as a function of SPAINV threshold.** *Each colored curve represents the quality metric behavior for a different SPAINV level pattern. Levels of 3, 4, 5, 6, and 7 were used.*

3.5.3 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:

$$\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}\mathbf{u} = \mathbf{L}^{-1}\mathbf{b}\,. \tag{3.21}$$

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

$$||\mathbf{I} - \mathbf{M}\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}||_F^2\,, \tag{3.22}$$

giving a new preconditioned system:

$$\mathbf{M}\mathbf{L}^{-1}\mathbf{A}\mathbf{U}^{-1}\mathbf{u} = \mathbf{M}\mathbf{L}^{-1}\mathbf{b}\,. \tag{3.23}$$

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 insensitive 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 3.20 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 3.21, 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 3.22 are only marginally better than those observed for ILUT preconditioning and much larger than the SPAINV quality metric values.

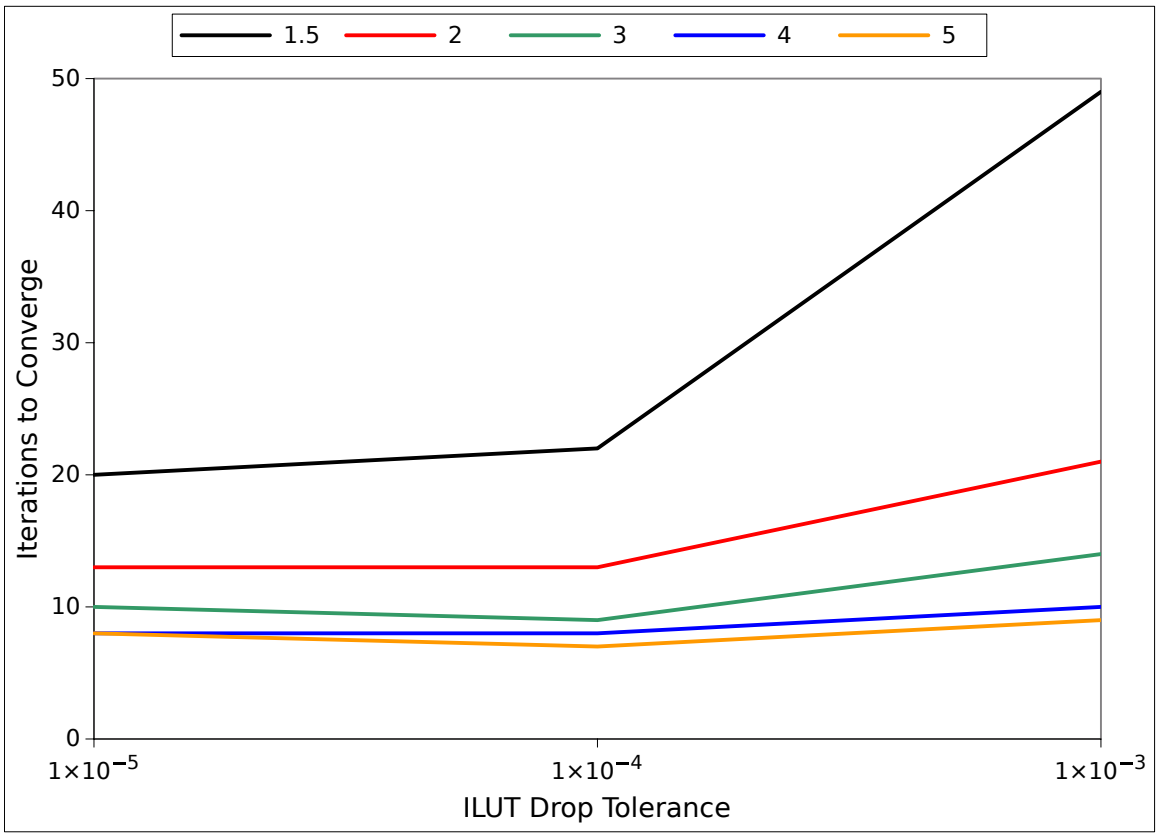

Figure 3.19: **Number of MCSA iterations required to converge a single eigenvalue iteration for the fuel assembly problem with SPAINV-ILUT preconditioning as a function of ILUT drop tolerance.** *Each colored curve represents the row size for a different ILUT fill level. Fill levels of 1.5, 2.0, 3.0, 4.0, and 5.0 were used.*

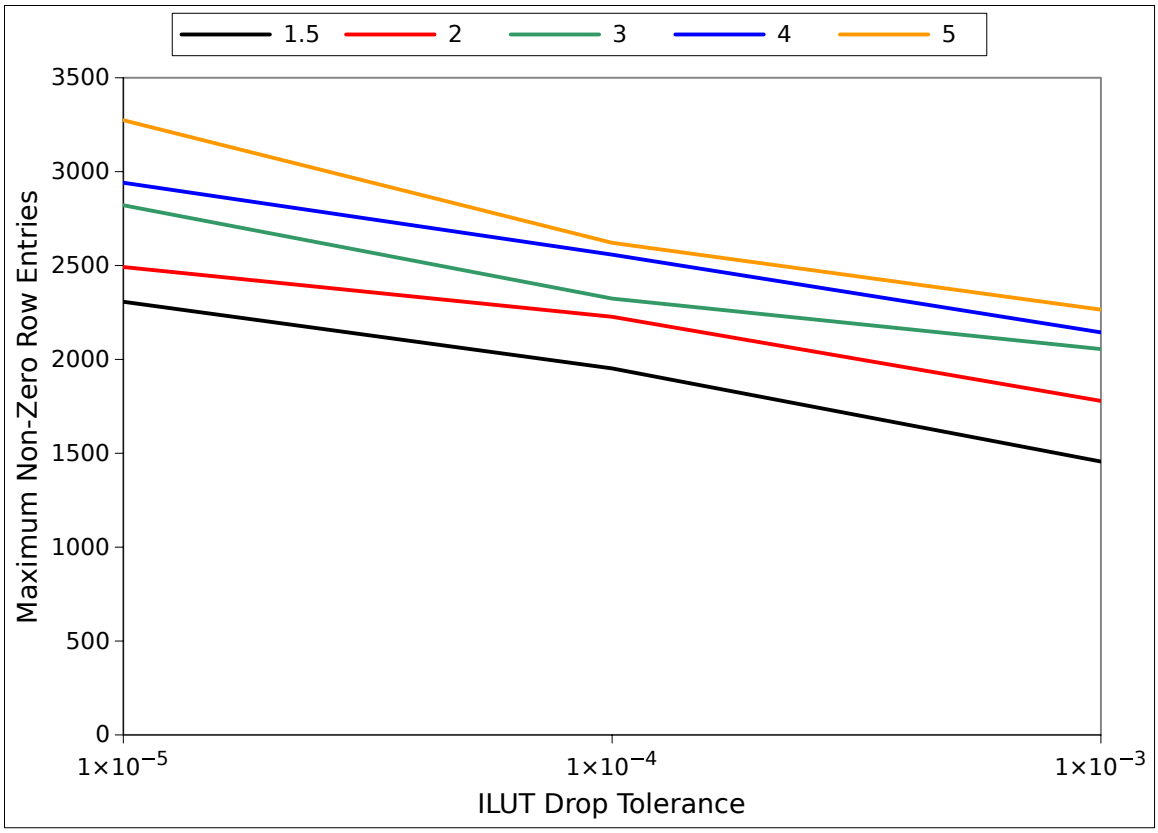

Figure 3.20: **Maximum number of non-zero entries observed for all rows in the composite linear operator for the fuel assembly problem with SPAINV-ILUT preconditioning given as a function of ILUT drop tolerance.** *Each colored curve represents the row size for a different ILUT fill level. Fill levels of 1.5, 2.0, 3.0, 4.0 and 5.0 were used.*

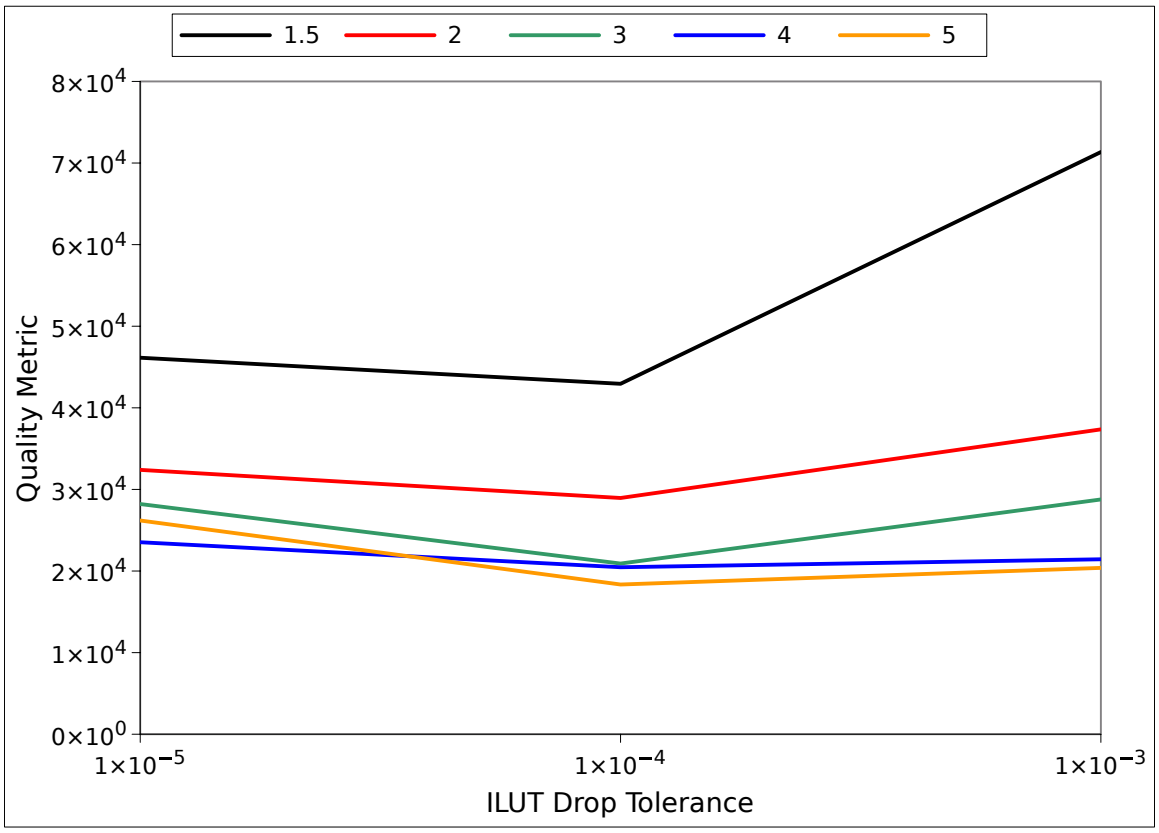

Figure 3.21: **SPAINV-ILUT preconditioning quality metric for the fuel assembly problem given as a function of ILUT drop tolerance.** *Each colored curve represents the row size for a different ILUT fill level. Fill levels of 1.5, 2.0, 3.0, 4.0 and 5.0 were used.*

3.5.4 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 § 2.10, 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 3.23 gives the number of iterations required to converge for each preconditioner type as a function of reduced domain fill level. Figure 3.24 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 that SPAINV preconditioning alone is significantly 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.

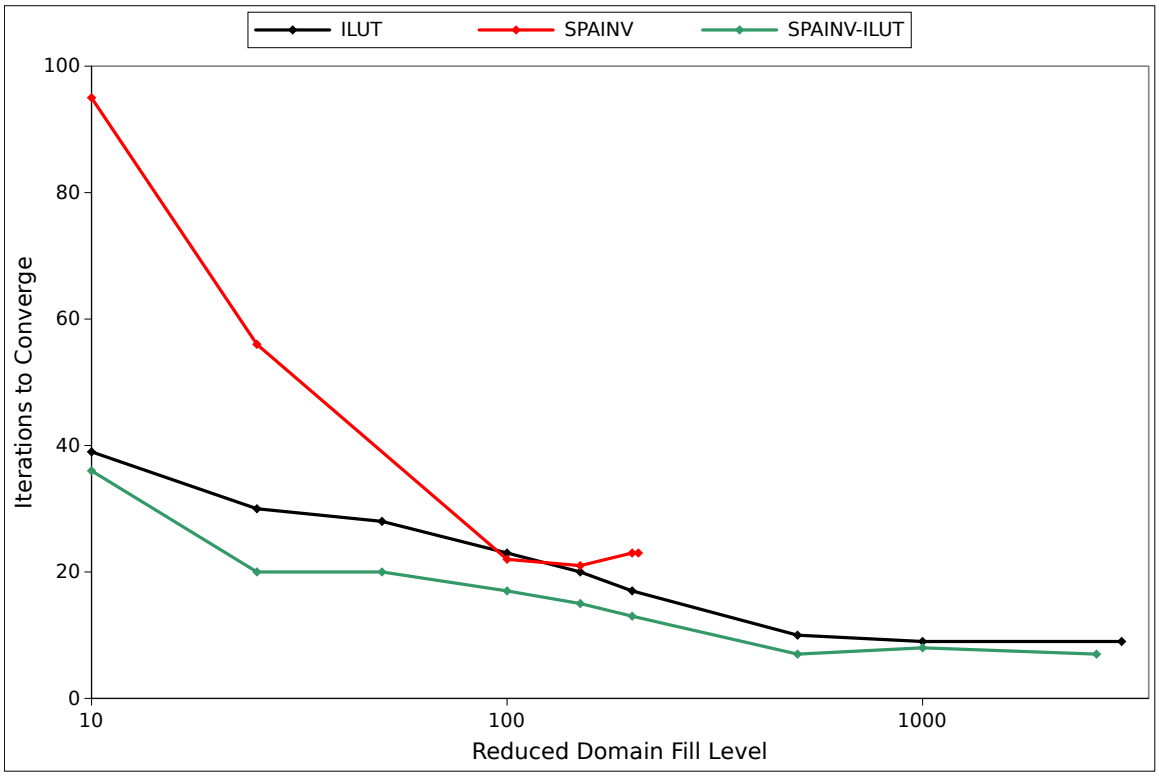

Figure 3.22: **Number of MCSA iterations required to converge a single eigenvalue iteration for the fuel assembly problem with each preconditioning as a function of reduced domain approximation fill level.** *The largest fill level for each preconditioning presented is that using the parameters that gave the best results without the approximation.*

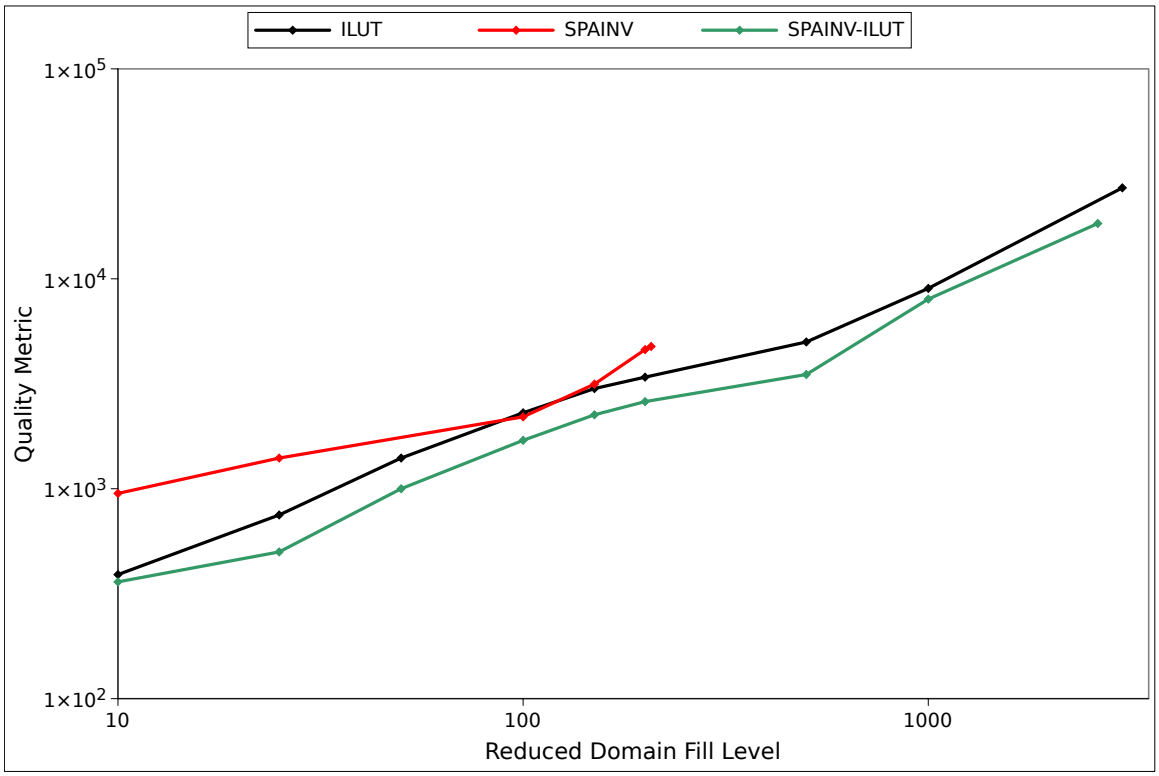

Figure 3.23: **Preconditioning quality metric for the fuel assembly problem given as a function of reduced domain approximation fill level.** *The largest fill level for each preconditioning presented is that using the parameters that gave the best results without the approximation.*

3.5.5 MCSA Relaxation Parameters

As another means of preconditioning (or variance reduction), 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:

$$\mathbf{x} = \mathbf{x} + \omega \mathbf{r} \,, \tag{3.24}$$

where ω is the relaxation parameter. This is very similar to point Jacobi 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 (3.32) gives:

$$\mathbf{H} = \mathbf{I} - \omega \mathbf{A} \,, \tag{3.25}$$

with the probabilities and weights for the Monte Carlo procedure 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:

$$\mathbf{x}^{k+1/2} = \mathbf{x}^{k} + \omega_R \mathbf{r}^{k} \,, \tag{3.26a}$$

$$\mathbf{r}^{k+1/2} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1/2} \,, \tag{3.26b}$$

$$\omega_N \mathbf{A} \delta \mathbf{x}^{k+1/2} = \omega_N \mathbf{r}^{k+1/2} \,, \tag{3.26c}$$

$$\mathbf{x}^{k+1} = \mathbf{x}^{k+1/2} + \delta \mathbf{x}^{k+1/2} \,, \tag{3.26d}$$

$$\mathbf{r}^{k+1} = \mathbf{b} - \mathbf{A}\mathbf{x}^{k+1} \,, \tag{3.26e}$$

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. ILUT preconditioning with a drop tolerance of 1×10^{-5} and fill level of 5 is

used as well.

Figure 3.25 gives the number of iterations required to converge the fuel assembly problem for a 1-group SP1 discretization using varying combinations of the relaxation parameters. First, both parameters were fixed at the base case of 1 while the other parameter was varied as shown in the left-hand side plots of Figure 3.25. For the Richardson relaxation parameter, using a value larger than 1 gave better iterative performance up to a point, effectively providing a stronger extrapolation using the residual at each iteration. For the Neumann relaxation parameter, a value of less than 1 is observed to be ideal. Although initially counter-intuitive, the fact that the correction computed by the Monte Carlo solver has a stochastic error associated with it means that by using a Neumann relaxation parameter less than 1, the correction and its error are effectively dampened to improve iterative performance. Considering the CPU time required to converge a single eigenvalue iteration, similar results are also observed for the relaxation parameters as given by Figure 3.26. For the base cases given by the plots on the left, it was found the a Richardson relaxation parameter of 1.1 and a Neumann relaxation parameter of 0.7 provided the fastest CPU time for convergence. Fixing each parameter at these new values, the calculations were repeated as shown for the plots on the right hand sides of both Figures 3.25 and 3.26. For each repeated timing calculation, it was found that the same combination of relaxation parameters found in the base cases performed the best although they did not necessarily have the best iterative performance.

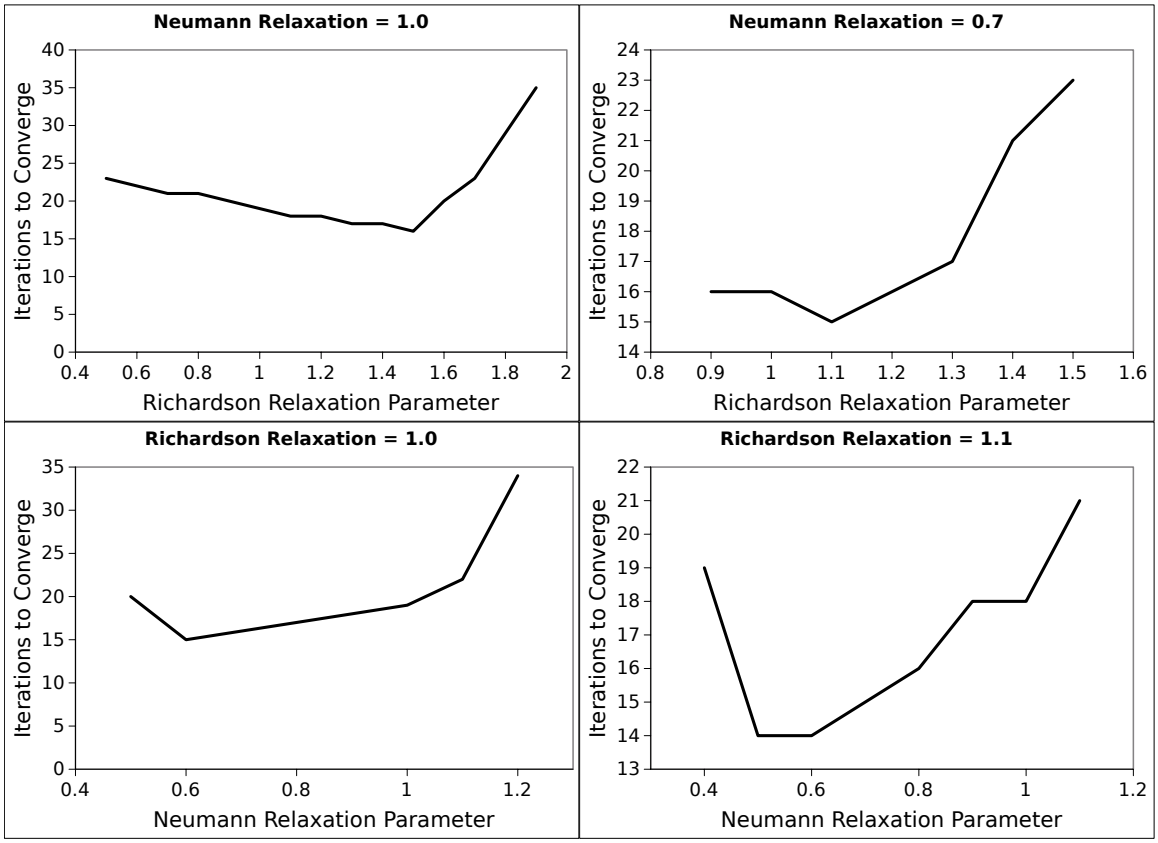

Figure 3.24: **Number of iterations to converge a single eigenvalue iteration of the fuel assembly problem as a function of the relaxation parameters.** *Starting in upper left and moving counter-clockwise: Neumann relaxation parameter fixed at 1.0, Richardson relaxation parameter fixed at 1.0, Neumann relaxation parameter fixed at 0.7, Richardson relaxation parameter fixed at 1.1. For each calculation* 3×10^4 *stochastic histories were used to compute the MCSA correction at each iteration.*

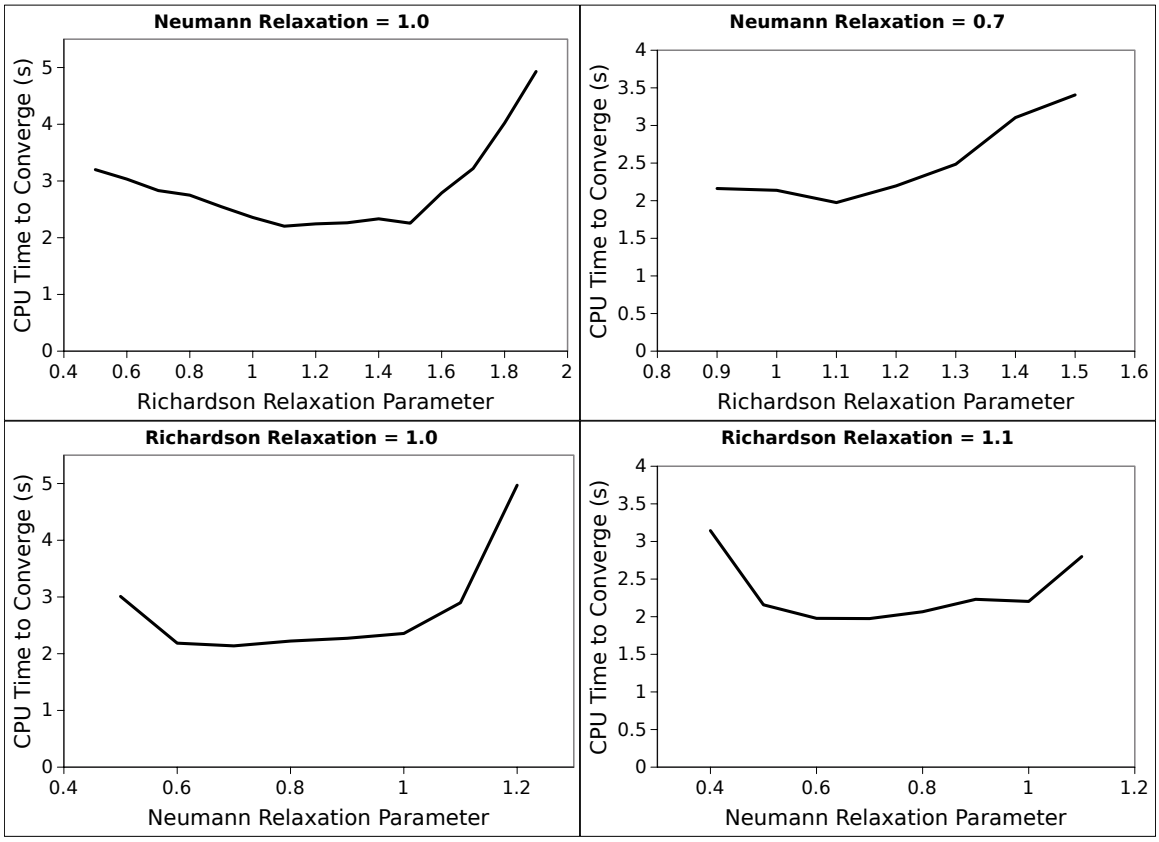

Figure 3.25: **CPU time in seconds to converge a single eigenvalue iteration of the fuel assembly problem as a function of the relaxation parameters.** *Starting in upper left and moving counter-clockwise: Neumann relaxation parameter fixed at 1.0, Richardson relaxation parameter fixed at 1.0, Neumann relaxation parameter fixed at 0.7, Richardson relaxation parameter fixed at 1.1. For each calculation* 3×10^4 *stochastic histories were used to compute the MCSA correction at each iteration.*

3.5.6 Monte Carlo Estimator Comparison

With convergence obtained for the fuel assembly problem, the advanced preconditioners will next be studied with both the collision and expected value estimators using the best preconditioning and relaxation parameter combinations found by the previous analysis. Additionally, both the Richardson-based MCSA iteration given by Eq (2.65) and the MINRES-based MCSA iteration given by Eq (2.67) will be used. Although the expected value estimator achieved the best iterative performance in the previous comparison and analysis for the simple Poisson problem in § 2.9, it is possible that the collision estimator 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.

For this study, the fuel assembly problem with a 1-group SP_1 discretization was solved using MCSA with the adjoint Monte Carlo solver using both the collision and expected value estimators and both the Richardson and MINRES fixed point iterations. Using the results from the previous section, a Neumann relaxation parameter of 0.7 was used with both estimators and fixed point iterations. For the collision estimator, a Richardson relaxation parameter of 1.1 was used with the Richardson iteration while it was found the expected value estimator had the best performance when a Richardson relaxation parameter of 1.0 was used instead. In addition, ILUT preconditioning with a drop tolerance of 1×10^{-5} and a fill level of 5 was used with a reduced domain approximation fill level of 100 and a threshold of 1×10^{-10} as in the previous analysis. Artificial absorption was also introduced with a value of 0.2 for the MCSA calculations that used the collision estimator as CPU timing performance was improved without a loss in iterative performance while no absorption was used for calculations using the expected value estimator. For each estimator, the number of stochastic histories used to compute the MCSA correction was varied from 25 to 50,000 with the number of MCSA iterations required to converge to a tolerance of 1×10^{-8} recorded for a single eigenvalue iteration along with the CPU time required to achieve convergence.

Figure 3.27 gives the iteration count results and Figure 3.28 gives the timing results. Remarkably, although the fuel assembly problem is significantly more complicated than the simple Poisson problem studied in § 2.9, the results here are largely the same with Figure 3.27 and Figure 2.11 having identical qualitative behavior. Using the expected value estimator with both fixed point iterations creates a relative insensitivity

to the number of histories used to compute the correction. Marginally better iterative and timing performance was observed for the expected value estimator when using the Richardson iteration. Better timing performance is expected as computing the extrapolation parameter in the MINRES iteration requires several additional parallel operations. However, we expect that because the MINRES iteration will converge faster on its own when compared to the Richardson iteration that better iterative performance will be observed when it is accelerated with MCSA. The fact that this was not observed signals that an MCSA scheme leveraging the expected value estimator is also effectively insensitive to the fixed point iteration used. For the collision estimator, the MINRES iteration permits fewer stochastic histories to be used with each MCSA iteration while still maintaining good iterative performance. With respect to CPU time, there is a more pronounced effect due to the larger density of states created by the explicit ILUT preconditioning. Using a reduced domain fill level of 100 creates a situation where the expected value estimator is significantly more expensive per history to compute that the collision estimator due to the coupling of states.

From both Figure 3.27 and Figure 3.28 we see that both estimators and fixed point iterations have comparable iterative and timing performance when 1,000 histories are used at each MCSA iteration. Using 1,000 histories per MCSA iteration, the infinity norm of the residual vector is presented at every MCSA iteration for both estimators and both fixed point iterations in Figure 3.29 as an additional means of comparison. At this number of histories, all MCSA combinations converge the fuel assembly problem monotonically at and approximately the same rate, giving little reason to choose one over the other. The end result is that both estimators provide a viable method for getting good MCSA iterative performance with comparable timing performance. For the collision estimator, using around 20,000 stochastic histories at every iteration gave the best iterative performance while 500 histories per iteration gave the best iterative performance when using the expected value estimator. For purely serial computational performance, there is not a distinguishing feature for the fuel assembly problem presented that would cause one to choose one of the estimators and fixed point iterations over the other as they all have similar minimum CPU times over the range of values tested.

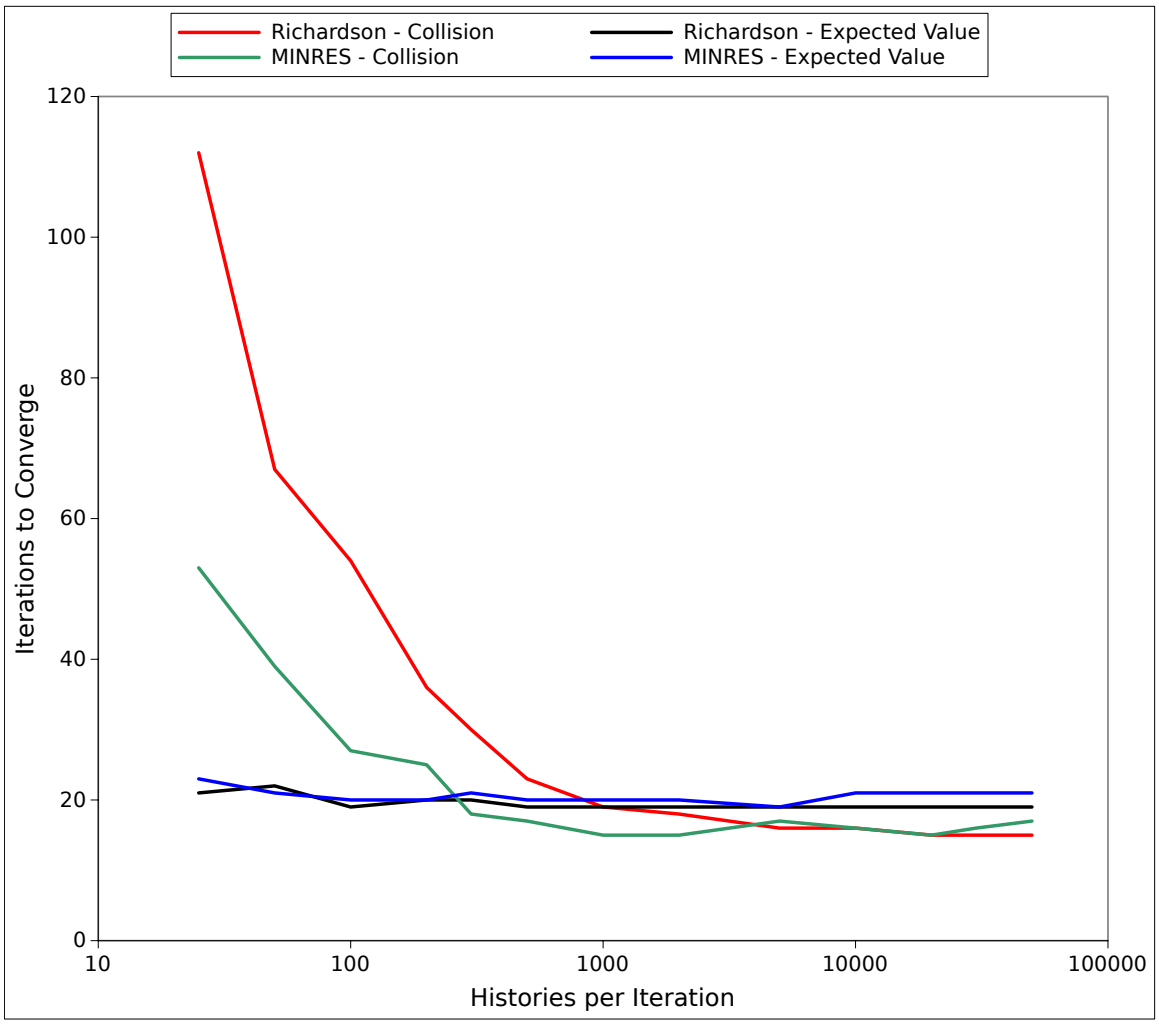

Figure 3.26: **MCSA iterations required to converge the fuel assembly problem to a tolerance of 1×10^{-8}.** *Both the collision and expected value estimators were used with the adjoint Monte Carlo solver to compute the correction at each iteration. Each estimator was used with the Richardson and MINRES fixed point iterations within an MCSA iteration.*

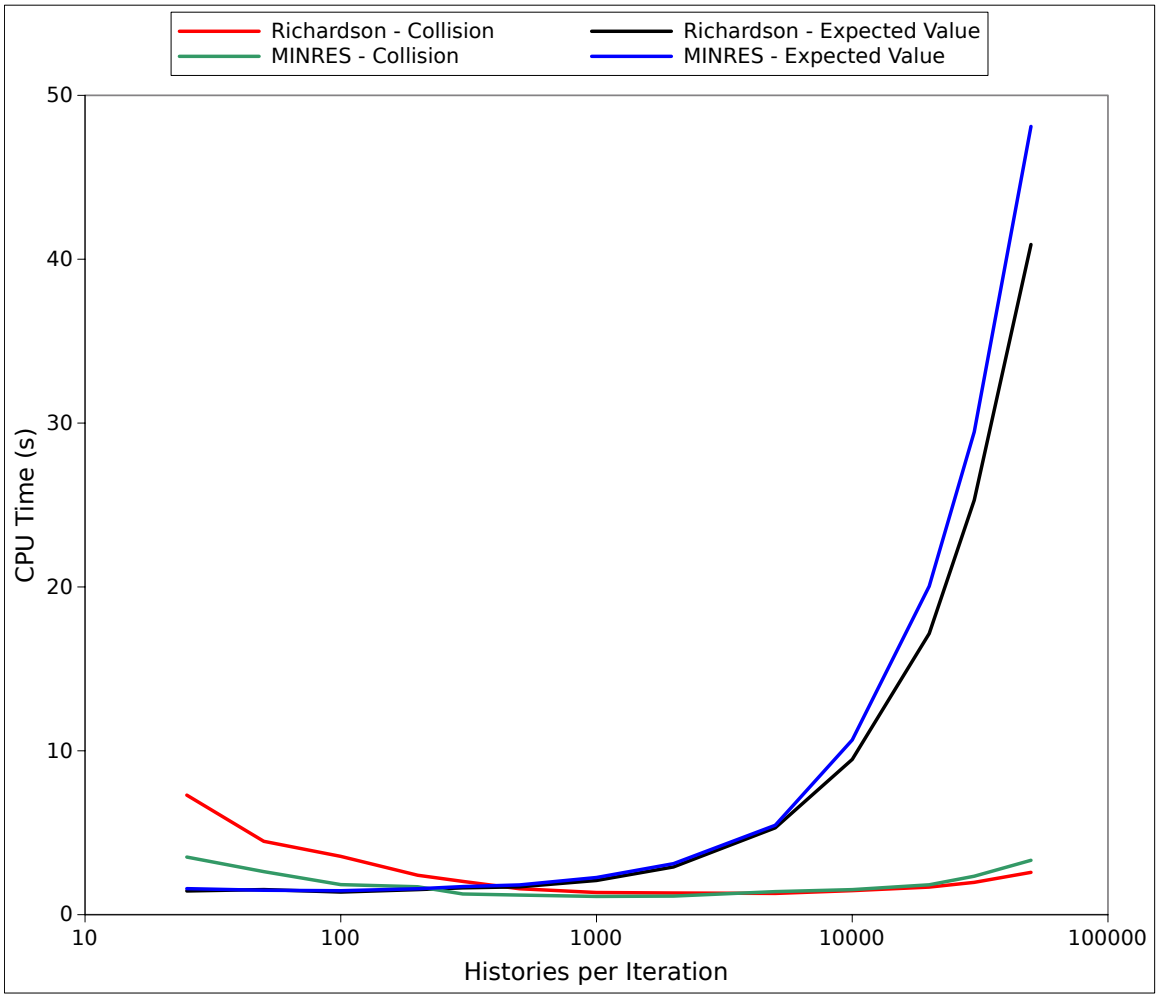

Figure 3.27: **Total CPU time in seconds required to converge the fuel assembly problem to a tolerance of 1×10^{-8}.** *Both the collision and expected value estimators were used with the adjoint Monte Carlo solver to compute the correction at each iteration. Each estimator was used with the Richardson and MINRES fixed point iterations within an MCSA iteration.*

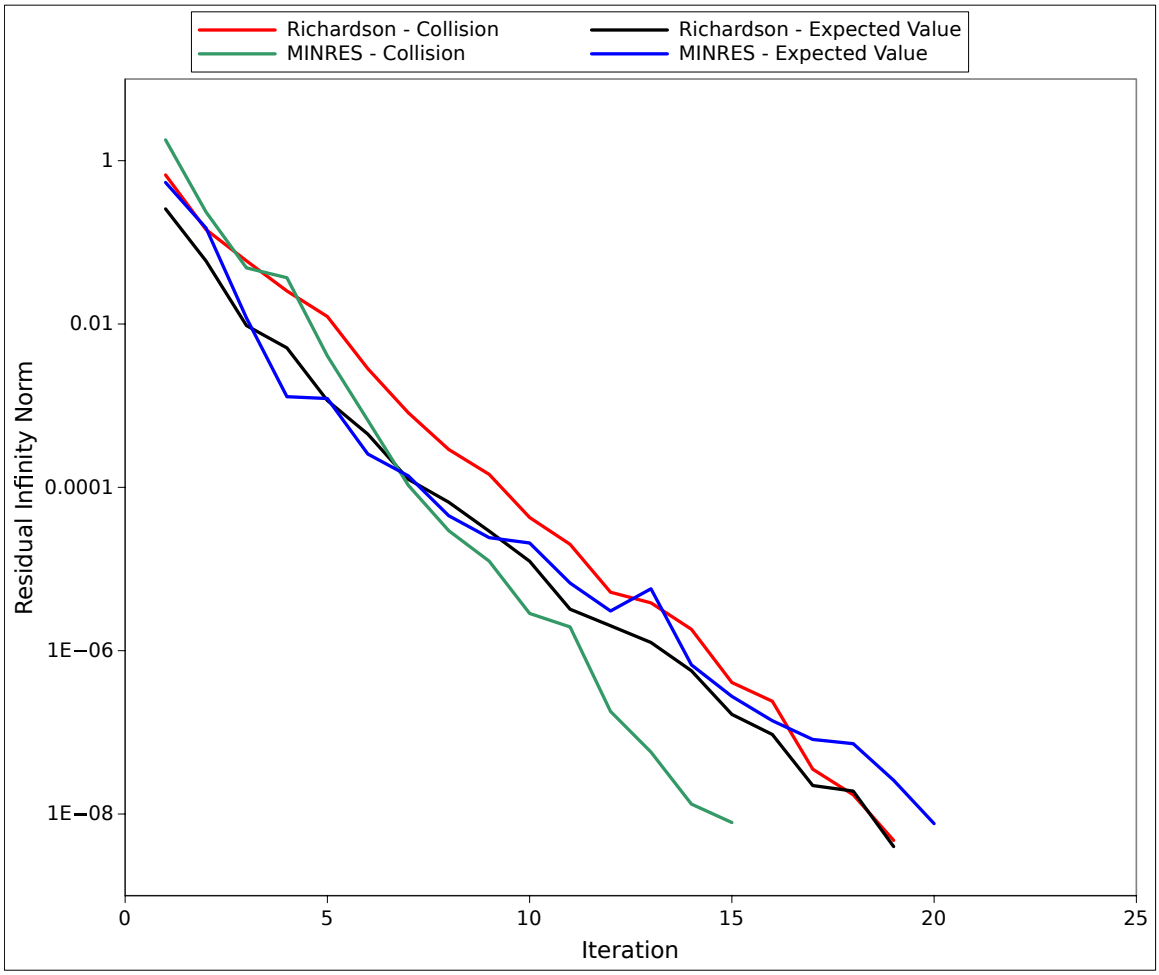

Figure 3.28: **Residual infinity norm as a function of MCSA iteration with 1,000 stochastic histories per iteration.** *Both the collision and expected value estimators were used with the adjoint Monte Carlo solver to compute the correction at each iteration. Each estimator was used with the Richardson and MINRES fixed point iterations within an MCSA iteration.*

3.6 MCSA Verification

Through the numerical studies presented in this chapter, we have developed a set of preconditioning techniques that permit MCSA to be used with the SP_N form of the neutron transport equation and applied to a difficult fuel assembly criticality problem. Along with these preconditioning techniques, a set a varying MCSA parameters were analyzed to find those that yielded the best performance for this particular problem. In addition, several steps were taken to mitigate the dense composite linear operators that arise when performing explicit MCSA preconditioning in order to achieve convergence. Using these results, we compare MCSA directly to conventional linear solvers that would be typically used to solve the SP_N form of the transport equation as presented here. In particular, we will use a pair of production Krylov solvers with which we will compare direct numerical results in order to verify MCSA solutions in this section.

In order to verify the correctness of the MCSA method and implementation used for this work, MCSA solutions to the fuel assembly problem with 1, 2, and 4 energy groups using SP_1 discretization will be directly compared over all group and cell fluxes computed by converging the criticality problem using both BiCGStab and GMRES production implementations from the Trilinos scientific computing library Aztec (Heroux et al., 2005). In addition to comparing the cell and group fluxes, the number of eigenvalue iterations required to converge the problem along with the eigenvalue itself will be compared as a means of verification. If all values computed using MCSA agree with the results from the production Krylov solvers within the solution tolerance, then the MCSA solutions will be deemed correct for these problems. Additionally, this verification is only applicable to serial results with parallel MCSA verification for this problem provided in Chapter 5.

For the verification calculations, all solvers will be preconditioned with ILUT using a drop tolerance of 1×10^{-5} and a fill level of 5. For GMRES, no restrictions were placed on the size of the subspace and therefore no restarts occurred. When the collision estimator was used with MCSA, 2×10^4 stochastic histories were used to compute the correction for every energy group in the problem (4×10^4 and 8×10^4 histories total for the 2 and 4 group calculations respectively) corresponding to approximately 1 stochastic history per DOF. When the expected value estimator was used, 5×10^2 stochastic histories used for each energy group (1×10^3 and 1.5×10^3 histories total for the 2 and 4 group calculations) giving approximately 1 stochastic history for every 5 DOFs. The reduced domain approximation was also applied in conjunction with the ILUT preconditioning using a fill level of 100 and a threshold value of 1×10^{-10} to

reduce the density of states in the Monte Carlo problem. For relaxation parameters, all MCSA computations used a Neumann relaxation parameter of 0.7 while a Richardson relaxation of 1.1 was used with the collision estimator and 1.0 with the expected value estimator as determined by the previous analysis of the relaxation parameters. Table 3.9 gives definitions for the solvers used to generate the results in the remainder of this section. For the energy group structures, Table 3.10 gives the lower bounds of each group in eV (an implicit upper bound of 2×10^6 eV is assumed for group 0).

Name	**Definition**
BiCGStab-ILUT	BiCGStab preconditioned with ILUT
GMRES-ILUT	GMRES preconditioned with ILUT
MCSA-ILUT-R-C	MCSA preconditioned with ILUT using Richardson fixed point iteration and collision estimator
MCSA-ILUT-MR-C	MCSA preconditioned with ILUT using MINRES fixed point iteration and collision estimator
MCSA-ILUT-R-EV	MCSA preconditioned with ILUT using Richardson fixed point iteration and expected value estimator
MCSA-ILUT-MR-EV	MCSA preconditioned with ILUT using MINRES fixed point iteration and expected value estimator

Table 3.9: **Solver definitions used for MCSA verification and performance analysis.** *The ILUT preconditioner parameters were identical for all calculations and solvers.*

Number of Groups	**Lower Bounds (eV)**
1	$\{ 1 \times 10^{-5} \}$
2	$\{ 1 \times 10^{-1}, 1 \times 10^{-5} \}$
4	$\{ 1 \times 10^{1}, 1 \times 10^{0}, 1 \times 10^{-1}, 1 \times 10^{-5} \}$

Table 3.10: **Energy group lower bounds for the multigroup fuel assembly criticality problem in electron volts.** *An implicit upper bound of* 2×10^6 *eV is assumed for group zero.*

For every combination of energy group structure and solver, the eigenvalue problem was converged with an eigensolver tolerance of 1×10^{-6} and a linear solver tolerance of 1×10^{-8} in a serial calculation. Table **??** gives the eigenvalue computed by each solver for each problem as well as the number of eigenvalue iterations required to converge. For every problem, each solver converged to the same eigenvalue within the tolerance of the eigensolver in the same number of eigenvalue iterations.