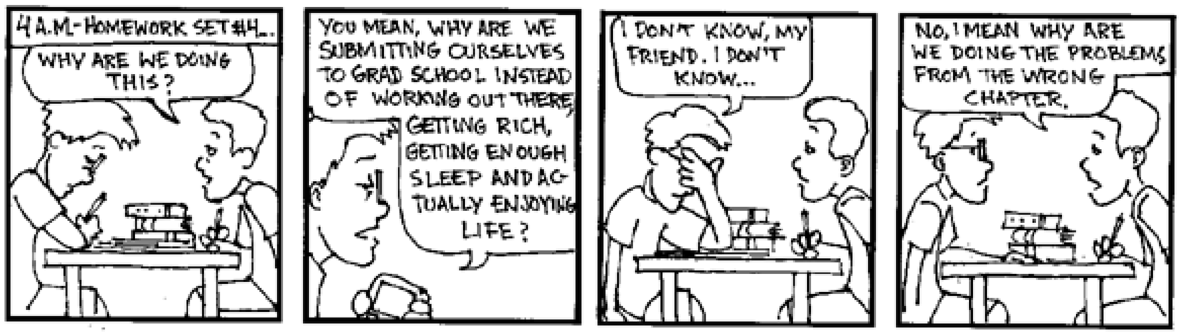
A Multilevel in Space and Energy Solver for Multigroup Diffusion and Coarse Mesh Finite Difference Eigenvalue Problems

by

Ben C. Yee

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Nuclear Engineering and Radiological Sciences)
in the University of Michigan
2018


4 A.M. - HOMEWORK SET #4..
WHY ARE WE DOING THIS?
YOU MEAN, WHY ARE WE SUBMITTING OURSELVES TO GRAD SCHOOL INSTEAD OF WORKING OUT THERE, GETTING RICH, GETTING ENOUGH SLEEP AND ACTUALLY ENJOYING LIFE?
I DON'T KNOW, MY FRIEND. I DON'T KNOW...
NO, I MEAN WHY ARE WE DOING THE PROBLEMS FROM THE WRONG CHAPTER.

— Jorge Cham, *PhD Comics*, 1997

"If you can't fly, run; if you can't run, walk; if you can't walk, crawl; but by all means keep moving."

— Dr. Martin Luther King Jr., "Keep Moving from this Mountain," 1960

For Mom, Dad, Winnie, and grandma

ACKNOWLEDGMENTS

"You are never strong enough that you don't need help."

— César Chávez

First and foremost, I would like to thank my co-advisors, Edward Larsen and Brendan Kochunas, for all of their guidance and patience. Without them, this dissertation would be an apology letter for a poorly conceived method that does not work and has no real world applications. Brendan and Ed, I cannot express in words how much I appreciate all that you have done for me.

Second, I would like to thank my committee for their invaluable support. In particular, I would like to thank Professor Downar for taking me into the MPACT group and ensuring that I never had to worry about project funding, and I would like to thank Professor Veerapaneni for his mathematical expertise and for all that I have learned in his courses.

I would also like to express gratitude to the U.S. Department of Energy's Nuclear Energy University Program (NEUP), which supported me in the form of a generous fellowship, and to the Consortium for the Advanced Simulation of Light Water Reactors (CASL) (`www.casl.gov`), an Energy Innovation Hub (`http://www.energy.gov/hubs`) for Modeling and Simulation of Nuclear Reactors under U.S. Department of Energy Contract No. DE-AC05-00OR22725. Moreover, this research used resources of the Oak Ridge Leadership Computing Facility (OLCF) at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Next, I would like to thank all the friends and colleagues I have gained in my five years here. Graduate school was orders of magnitude more bearable thanks to all the fun times we shared, either at school and outside of school. In particular, thank you Daniel for being such a great roommate and friend. To those of you on the 4th floor of ERB – thank you for listening to all of my frustrations and complaints. I hope that you will forgive me for all of the bugs that my commits may cause in MPACT.

To all of my friends back home, thank you for all of the support and encouragement you have given me in spite of the thousands of miles that separate us. Some of you were even crazy enough to fly across the country to visit me! Aaron, Jimmy, Lawrence, thank you for all of the fun, stress-relieving times we shared. Boris, thank you for being a great friend, and for being the best homework and research buddy I could ask for. Norman, thank you for being there for me ever since kindergarten.

This acknowledgments section would be far too long if I pointed out all the specific ways in which all of my friends, in Ann Arbor, back home, or elsewhere in the world, have made my life better. To those that I have not explicitly mentioned here, please know that I truly appreciate every favor you have done for me, every time you have made me laugh, and every moment we have shared together.

Finally, and most importantly, I would like to thank my family. They moved across an ocean so that I could have a better future, spent decades working difficult low-wage jobs that make graduate school seem like a breeze, and still managed to put my needs ahead of their own. In a manner that would leave even a time-traveling wizard in awe, my mom and dad always managed to prepare delicious, homemade meals each and every night in spite of their grueling work hours. They took every opportunity to make sure that I was fed, clothed, healthy, and loved. Even now, every phone call I have with my parents invariably begins, "Have you eaten yet?" Despite a difficult life that should leave one wanting nothing but rest and relaxation, my grandma spent her post-retirement years watching over me while my parents were at work, showering me with love, and constantly reminding me of how much she believed in me. My family inspires me to keep moving forward, and I cannot say enough about what they have sacrificed for me. I owe them everything.

余家 – from left to right: Winnie, Mom, grandma, Dad, me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

LIST OF APPENDICES

LIST OF ABBREVIATIONS

BiCGSTAB	Biconjugate Gradient Stabilized
BOC	beginning of cycle
BWR	boiling water reactor
CASL	Consortium for the Advanced Simulation of Light Water Reactors
CMFD	Coarse Mesh Finite Difference
CMR	Coarse Mesh Rebalance
CR	CMFD runtime
DSA	Diffusion Synthetic Acceleration
DNC	did not converge
EOC	end of cycle
EXSC	equivalence cross section calculations
FLOP	floating-point operation
GD	Generalized Davidson
GLSI	grey LSI
GMRES	Generalized Minimal Residual
GPI	grey PI
GR	grey runtime
GS	Gauss-Seidel
HOLO	High-Order Low-Order
HFP	hot full power
HZP	hot zero power

ILU	Incomplete LU
LA	local absorption
LAPS	LA Positive Source
LE	local eigenvalue
LEPS	LE Positive Source
LONDA	low order nonlinear diffusion acceleration
LSI	linear solver iteration
LU	“lower-upper”
LSI	linear solver iteration
LWR	light water reactor
MLCMR	Multilevel CMR
MLSR	Multilevel Surface Rebalance
MPI	Message Passing Interface
MPACT	Michigan Parallel Characteristics Transport
MOC	method of characteristics
odCMFD	optimally diffusive CMFD
MED	Multilevel-in-Energy Diffusion
MGLSI	multigroup LSI
MGPI	multigroup PI
MSED	Multilevel-in-Space-and-Energy Diffusion
OLCF	Oak Ridge Leadership Computing Facility
PARCS	Purdue Advanced Reactor Core Simulator
PB	Peach Bottom
PDE	partial differential equation
PLAPS	PARCS LAPS
PLEPS	PARCS LEPS
PI	power iteration

RBBJ	“red-black” block Jacobi
SDWS	space-dependent Wielandt shift
SOR	successive over-relaxation
TCP$_0$	Transport-Corrected P_0
TR	total runtime
TS	transport sweep
VERA	Virtual Environment for Reactor Applications
WS	Wielandt shift

ABSTRACT

In reactor physics, the efficient solution of the multigroup neutron diffusion eigenvalue problem is desired for various applications. The diffusion problem is a lower-order but reasonably accurate approximation to the higher-fidelity multigroup neutron transport eigenvalue problem. In cases where the full-fidelity of the transport solution is needed, the solution of the diffusion problem can be used to accelerate the convergence of transport solvers via methods such as Coarse Mesh Finite Difference (CMFD). The diffusion problem can have $O(10^8)$ unknowns, and, despite being orders of magnitude smaller than a typical transport problem, obtaining its solution is still not a trivial task. In the Michigan Parallel Characteristics Transport (MPACT) code, the lack of an efficient CMFD solver has resulted in a computational bottleneck at the CMFD step. Solving the CMFD system can comprise 50% or more of the overall runtime in MPACT when the de facto default CMFD solver is used; addressing this bottleneck is the motivation for our work.

The primary focus of this thesis is the theory, development, implementation, and testing of a new Multilevel-in-Space-and-Energy Diffusion (MSED) method for efficiently solving multigroup diffusion and CMFD eigenvalue problems. As its name suggests, MSED efficiently converges multigroup diffusion and CMFD problems by leveraging lower-order systems with coarsened energy and/or spatial grids. The efficiency of MSED is verified via various Fourier analyses of its components and via testing in a 1-D diffusion code. In the later chapters of this thesis, the MSED method is tested on a variety of reactor problems in MPACT. *Compared to the default CMFD solver, our implementation of MSED in MPACT has resulted in an ∼8-12x reduction in the CMFD runtime required by MPACT for single statepoint calculations on 3-D, full-core, 51-group reactor models.* The number of transport sweeps is also typically reduced by the use of MSED, which is able to better converge the CMFD system than the default CMFD solver. This leads to a further savings in overall runtime that is not captured by the differences in CMFD runtime.

This thesis also includes related work on the development of space-dependent Wielandt shifts (SDWSs). These shift methods were an attempt to improve upon the Wielandt-shifted power iteration (PI) schemes that are commonly used to solve diffusion eigenvalue problems. These efforts fell short of our goal of significantly reducing the computational burden of CMFD in MPACT, but

they provided insight regarding the impact of Wielandt shift (WS) on the conditioning of diffusion operators and ultimately motivated our development of MSED.

The thesis is a fairly lengthy document, and we understand that most readers may not wish to read the entirety of it in detail. For readers who wish to understand the "big picture" and see the final results, the gist of the method is described in Chapter 5, and the most important results are given in Section 7.4 and Chapter 8. An outline of the thesis is provided in Section 1.5. For the convenience of the reader, a list of acronyms is provided at the beginning of this document, and detailed descriptions of the MPACT problems used in this thesis are given in Appendix C. For those viewing this document as an electronic PDF, we note that all equation numbers, acronyms, table numbers, figure numbers, algorithm numbers, and citations are hyperlinked to their source or definition.

CHAPTER 1

Introduction

The goal of this thesis is to develop a highly efficient Multilevel-in-Space-and-Energy Diffusion (MSED) method for solving multigroup diffusion and Coarse Mesh Finite Difference eigenvalue problems. In this chapter, we motivate the development of MSED, provide a historical overview of multilevel and multigrid techniques, and describe the differences between MSED and previous multilevel techniques. At the end of this chapter, we preview the remaining chapters of this thesis.

1.1 Motivation

Computing the distribution of neutrons within a nuclear reactor is essential to its design and analysis. Knowledge of this distribution allows for the calculation and estimation of important quantities, including but not limited to: the power distribution, the critical boron concentration, and the fuel depletion rates. The neutron distribution can be obtained by solving a 6-dimensional steady-state neutron transport eigenvalue problem that describes the interaction, generation, and loss of neutrons inside the reactor. Because of its high-dimensionality, solving the neutron transport equation is a difficult task that constitutes a large percentage of the computational resources required for the design and analysis of nuclear reactors.

Often, an approximate 4-dimensional steady-state neutron diffusion eigenvalue problem is solved instead of the transport problem. One can generate diffusion coefficients and cross sections from smaller transport problems with appropriate boundary conditions, and then solve a diffusion problem for the full reactor [1]. The resulting solution is an acceptable approximation to the transport equation in many useful applications. When the full fidelity of the neutron transport equation is required, techniques such as Diffusion Synthetic Acceleration (DSA) [2] and Coarse Mesh Finite Difference (CMFD) [3] can leverage solutions of the neutron diffusion equation to accelerate the convergence of the iterative methods used to solve the neutron transport equation.

The work in this thesis is implemented primarily for the purpose of improving the efficiency of the Michigan Parallel Characteristics Transport (MPACT) code. MPACT is a state-of-the-art neutron transport code developed jointly by the University of Michigan and the Oak Ridge National Laboratory, and it is capable of providing pin-resolved accuracy for reactor physics applications [4]. In MPACT, optimally diffusive CMFD (odCMFD) – an unconditionally stable variant of CMFD developed by Zhu et al [5] – is used to reduce the number of iterations required on the 6-dimensional transport problem by leveraging the solution of approximate diffusion systems. Because of the success of odCMFD in minimizing the effort required by the transport solver as well as optimizations to non-CMFD components of MPACT, the efficiency of MPACT has significantly improved over the past few years.

However, there has not been much work in improving the efficiency of the CMFD solver in MPACT. As a result, it has become a bottleneck in the MPACT code, and it is one of the primary areas in which the efficiency of MPACT can be significantly improved. Although the CMFD problem has fewer dimensions (no angular dependence) and a coarser spatial grid than the original transport problem, solving the CMFD problem is not trivial because it still has $O(10^8)$ unknowns for full-core problems. Presently, despite the fact that the CMFD problem is several orders of magnitude smaller than the transport problem, CMFD can constitute 50% or more of the overall runtime in MPACT. Furthermore, because of the ineffectiveness of the de-facto CMFD solver in MPACT, we are often unable to achieve the low spectral radii predicted by [5] for the outer odCMFD-accelerated transport scheme, and additional costly transport sweeps are needed as a result. The extra costs stemming from the use of an inefficient CMFD solver in MPACT can be seen in [6] as well as the results in Chapters 6, 7, and 8. Given its relative size and its role as a "low-order" accelerator, we expect that fully converging the CMFD problem should constitute no more than 10% of the overall runtime.

The computational bottleneck in the CMFD component of MPACT is the motivation for the work in this thesis. Our overarching goal is to develop a method that can efficiently solve multigroup diffusion and CMFD eigenvalue problems. For MPACT specifically, we wish to spend $\sim$10% or less of the overall runtime in the CMFD solver, but still produce sufficiently converged CMFD solutions so that we can achieve the spectral radii predicted by [5] for the outer odCMFD-accelerated transport scheme. That is, we would like odCMFD-acceleration of transport in MPACT to be both effective and computationally inexpensive. With the de-facto CMFD solver in MPACT, it is typically neither. Although the work in this thesis focuses on the implementation of MSED in the MPACT code, MSED can be used in the same manner for any CMFD or diffusion eigenvalue problem on a Cartesian grid with a finite-differenced spatial discretization. MSED should also work for CMFD or diffusion problems with non-Cartesian grids and/or other spatial discretizations, but some modifications will be needed to the multigrid-in-space component.

Our development of the MSED method for solving multigroup diffusion/CMFD problems was driven by two primary principles. First, we wanted a method that takes full advantage of the specific physics of the CMFD problem. Before the implementation of the work in this thesis, the default approach for solving the CMFD system in MPACT relied heavily on “black-box” solvers from external libraries – in particular, MPACT had relied on Krylov solvers from libraries such as PETSc [7] and Trilinos [8]. These Krylov solvers are convenient because they absolve developers from the burden of writing their own optimized solver, but this convenience comes at a price. Krylov solvers are designed for general problems, and, unless a well-designed problem-dependent preconditioner is provided, they are agnostic to the physics and geometry of the problem. In applications outside reactor physics with complicated geometries, unstructured grids, and/or nonlinear phenomena, it is often difficult to develop an efficient solution scheme from the physics of the problem, and any “black-box” solver that works reasonably well is a welcome sight. However, the CMFD problem in MPACT is not such a problem. It is defined on a relatively structured Cartesian grid, and its parameters are constrained by well-understood physical processes. This makes it possible to analyze the equations in depth and study the convergence of the error modes of the problem in the context of the physics at hand.

Second, we wanted a method that extends the multilevel approach of CMFD. For many decades, mathematicians have used a multilevel approach known as multigrid to solve elliptic partial differential equations (PDEs) with varying scales of physical phenomena [9]. The idea behind these multilevel approaches is similar in all applications: iteratively solve a system with many unknowns by leveraging the solution of a smaller approximate system. In CMFD, a smaller 4-dimensional diffusion system (with correction factors) is leveraged to solve the larger 6-dimensional transport system. With MSED, we extend this idea further by leveraging smaller, approximate versions of the diffusion problem. Whereas CMFD leverages smaller diffusion problems to solve the larger transport problem, MSED leverages systems of equations that are even smaller than the CMFD diffusion problem to solve the CMFD diffusion problem. An illustration of this multilevel hierarchy of equations is shown in Figure 1.1, which has been reproduced from [10, 11]. The components shown in this figure are elucidated later in this thesis.

1.2 A Historical Review of Multigrid Methods for Elliptic PDEs

The numerical solution of PDEs is of great interest to many different fields of research. Linear PDEs occur in one of three categories: elliptic, hyperbolic, and parabolic. Typically, elliptic PDEs describe physical phenomena in which every point in space is relatively tightly coupled, hyperbolic PDEs describe physical phenomena in which the coupling is mostly constrained to “characteristic” paths or streamlines, and parabolic PDEs typically describe time-dependent versions of elliptic problems.

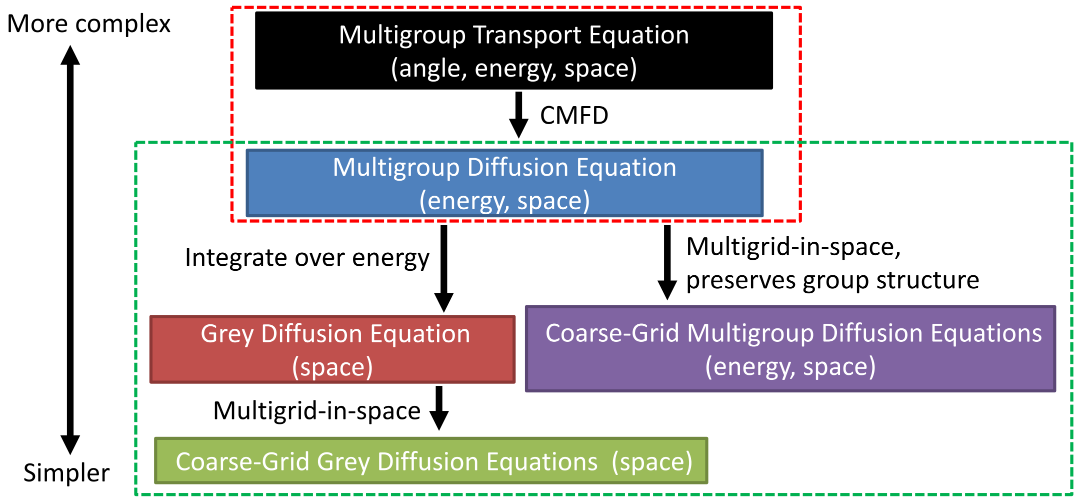

More complex
Simpler
Multigroup Transport Equation
(angle, energy, space)
CMFD
Multigroup Diffusion Equation
(energy, space)
Integrate over energy
Multigrid-in-space,
preserves group structure
Grey Diffusion Equation
(space)
Coarse-Grid Multigroup Diffusion Equations
(energy, space)
Multigrid-in-space
Coarse-Grid Grey Diffusion Equations (space)

Figure 1.1: A visual of the multilevel hierarchy of equations in CMFD and MSED. The upper (red) dashed box encloses the equations used in CMFD, while the lower (green) dashed box encloses the equations used in MSED. The levels in this hierarchy are elucidated later in this thesis, and we suggest that the reader revisit this figure at those junctures.

This is not a complete description of the different types of PDEs, and the reader may find more information in textbooks such as [12]. Although the transport equation is formally hyperbolic, the diffusion equation used to accelerate its convergence is elliptic. Thus, we are primarily concerned with elliptic PDEs in this thesis.

Devising efficient solution strategies for discretized elliptic PDEs is a challenging task, the difficulty of which is compounded by the increasingly parallel nature of modern computing architectures. Because the phenomena described by elliptic PDEs tend to be "global," one cannot simply use a "divide-and-conquer" approach. This poses a natural limit on how much one can parallelize the problem. Every solution point in space has a nontrivial dependence on every other solution point, and any solver must find a way to propagate information throughout the entire domain. In an ideal world, one would directly solve the linear systems corresponding to discretized forms of these elliptic problems, but direct solvers generally scale as $O(n^3)$, where n is the number of unknowns. In most problems of interest, these linear systems are too large to solve directly, and iterative solvers are used instead. Iterative solvers can typically achieve computational complexities far better than $O(n^3)$ by taking advantage of the sparsity of the problem matrix. Although the matrix representation of a discretized PDE has $O(n^2)$ entries, it is often the case that only $O(n)$ of those entries are populated [13]. This sparsity is present in the discrete diffusion system used in CMFD.

The simplest iterative approaches are fixed-point smoothers such as Jacobi, Gauss-Seidel (GS), or successive over-relaxation (SOR). As the term "smoother" suggests, these methods are effective at "smoothing" the solution, rapidly eliminating high-frequency error modes. Although they are simple, can be easily implemented (especially in parallel), and eliminate high-frequency (local) error modes quickly, they are prohibitively slow at converging the low-frequency (global) error modes that dominate most elliptic problems [13].

For several decades, mathematicians have primarily overcome this problem by using a multilevel method known as *multigrid* [9,14,15]. Because smoothers are well-suited for eliminating the highest frequency error modes of a spatial grid, one can maximize their utility by applying them to coarser versions of the original spatial grid. When an error mode from the original fine grid is mapped to a coarser grid, its frequency is artificially increased. As a result, the error mode can be more readily eliminated by the application of smoothers. A typical iteration in the multigrid method is referred to as a “V-cycle”. In a “V-cycle,” one alternates between performing a smoothing step and restricting the resulting error onto a coarser spatial grid. When a sufficiently coarse spatial grid is reached, the problem is solved exactly with a direct approach such as LU (or with many iterations of an iterative approach such as GMRES), and the error corrections are interpolated successively back up to the finest grid. More complicated cycles such as “W-cycles” or “full V-cycles” also exist; in these, the order in which the grids are traversed is different [14].

Multigrid methods, when paired with a smoother that efficiently removes high-order error modes, have nearly optimal convergence rates for elliptic problems because they are able to eliminate both global and local error modes rapidly [16]. One of the most beneficial properties of multigrid methods is that the spectral radius of multigrid does not typically grow with the problem size. Because of this and the fact that each iteration has computational complexity $O(n)$, the multigrid method offers $O(n)$ convergence in ideal settings. This is in stark contrast to the “black-box” Krylov methods that many numerical simulation tools rely on today. Unlike multigrid methods, the convergence rates of Krylov methods typically degrade as the problem size increases. This is because the condition number of linear systems representing discretized PDEs typically increases with the problem size, and the convergence rate of Krylov methods degrades as the condition number of the matrix increases [13].

Today, the primary difficulties of implementing classic multigrid schemes are due to unstructured grids and parallelism. Unstructured grids are not directly related to the work in the thesis, as the CMFD problem in MPACT is defined in a nearly-regular Cartesian grid. However, unstructured grids have motivated the development of a class of multigrid methods known as *algebraic* multigrid [17,18,19,20], which is relevant to the multigrid method used in MSED. Thus far, the multigrid concepts we have described have been *geometric* in nature. Algebraic multigrid methods are attempts to generalize geometric multigrid methods to problems for which it is not easy to redefine the discretized operator of interest on a coarsened grid. Often, algebraic multigrid relies on heuristic and empirical algorithms to determine appropriate interpolation and restriction operators. Rather than attempt to rediscretize the PDE on a coarsened grid, algebraic multigrid uses a triple matrix product of the restriction operator, the original fine-grid operator, and the interpolation operator as its coarse-grid operator. The primary advantages of algebraic multigrid are its robustness and wide-reaching applicability, but the disadvantages include a possible degradation of the convergence rate and the memory and computational cost of computing the triple matrix product. Like Krylov solvers, algebraic multigrid methods can often be used as “black-box” solvers in which the user does not need to provide the solver anything more than the linear system in the form of a matrix. (However, the performance of these algebraic multigrid solvers can generally be improved with some optimization of its parameters to the problem at hand [19].) In general, multigrid methods are not explicitly geometric or algebraic; they are often a combination of both approaches. For example, one can define an “algebraic” multigrid method in which all of the parameters (such as the interpolation and restriction operators) are defined based on knowledge of the geometry. Elements of both geometric and algebraic multigrid are used in MSED, and more details regarding this matter are provided later in Chapter 7.

The second difficulty of implementing classical multigrid is parallelism. One of the primary reasons that multigrid is not more widely adopted, especially in the nuclear engineering community, is that it is not trivial to implement multigrid in a parallel setting. The communication-to-computation ratio of multigrid may not be optimal for parallel domains, as multigrid inherently requires communication between distant portions of the grid. Unless the coarsening process is terminated early, one inevitably reaches a point in which the number of unknowns is comparable to (if not smaller than) the number of processors used. When this happens, communication between processors becomes a much larger portion of the execution time; in order to obtain a global solution on the coarsest grid, one must consider all points of the grid [21, 22, 23]. Various attempts have been made to optimize multigrid (geometric and algebraic) for highly parallel modern computing architectures, such as [24, 25, 26, 27]. In MSED, we minimize the communication required by defining the coarsening such that the restriction and interpolation can be done locally on each processor; this process is described in detail in [28].

In some sense, these communication costs are unavoidable due to the nature of elliptic problems, regardless of the solver used, and it is not entirely correct to consider parallelism a weakness of multigrid. As noted earlier, elliptic problems are problems in which each spatial point is strongly coupled to nearly every other spatial point. Any solution to the problem must resolve this nontrivial dependence, and it is impossible to do this without some form of communication between all portions of the spatial domain. This communication requirement manifests itself in other, less apparent forms when using non-multigrid solvers. For Krylov methods, we have noted that their convergence worsens as the problem size grows, due to the growth of the condition number of the problem matrix. One way to overcome this deficiency in Krylov methods is to use preconditioners that are “global” in nature, such as Incomplete LU (ILU) (or even multigrid) rather than simpler, localized preconditioners such as Jacobi or GS. Unfortunately, such preconditioners do not scale well to highly parallelized problems on distributed memory architectures [29], as they are global in nature and require significant communication. Thus, we encounter the same parallel limitation with Krylov methods as with multigrid methods.

The use of multigrid methods in the field of reactor physics has been limited, as most reactor simulation tools in the 1990s or later have opted to use Krylov methods [4, 30, 31, 32]. The most notable effort to apply multigrid directly to neutron diffusion problems was that of Alcouffe and Dendy in the 1980s [33, 34, 35]. From a first glance, the neutron diffusion problem appears similar to Laplace operator problems that served as the primary motivation for the development of multigrid methods in the applied mathematics community. However, the neutron diffusion problem adds several unique challenges compared to the standard Poisson equation. First, there is the presence of heterogeneous diffusion coefficients embedded within the spatial derivatives. In [34], Alcouffe notes that these diffusion coefficients can be viewed as scalings of the spatial grid. He uses nonuniform, weighted interpolation operators – a technique normally used for nonuniform grid sizes – to solve the issues caused by strongly discontinuous diffusion coefficients. Second, unlike the Laplace problem which is typically described by a single PDE, the neutron diffusion problem is a weakly-coupled system of G PDEs, where G is the number of energy groups used to discretize the energy-dependence of the neutrons. (A detailed definition of "weakly-coupled" can be found in [36]; in short, any multigroup diffusion problem with physical cross sections is "weakly-coupled.") In [33], Alcouffe suggests solving these G equations one at a time so that the multigrid method can be applied in its usual (one-group) form, and using a GS-like iteration scheme to iterate through the equations many times until convergence. In MSED, a multigroup approach is used in the restriction, smoothing, and interpolation steps of multigrid so that all of the groups are solved simultaneously rather than one-at-a-time. More details are provided later in Chapter 7.

1.3 History of Multilevel Techniques in Reactor Physics

Although the use of (spatial) multigrid methods has been limited in reactor physics, the same cannot be said for general multilevel methods. In this section, we present a summary of such methods.

The earliest multilevel techniques in reactor physics were the Coarse Mesh Rebalance (CMR) methods, first developed in the 1960s for solving both neutron transport and nodal diffusion problems [37, 38, 39]. In CMR, the original fine-grid transport (or diffusion) equations are integrated in angle and over each coarse spatial cell to obtain a new set of equations. The solutions to these equations are shape factors used to coarsely scale the original fine-grid solution. Unlike more modern techniques such as DSA or CMFD, the coarsened equations of CMR do not limit to the diffusion equation as the grid size goes to zero. CMR can be implemented for general spatial grids relatively easily, but it has narrow regions of stability [40, 41]. Classical CMR methods are generally unstable when the coarse grid is too fine, and too slow when the coarse grid is sufficiently coarse. In recent years, significant work has been performed by Van Geemert et al. to develop multilevel versions of CMR: Multilevel CMR (MLCMR) and Multilevel Surface Rebalance (MLSR) [42, 43, 44]. Yamamoto has also developed a Generalized CMR method that combines CMR and CMFD [45]. For problems with the appropriate grid sizes, these multilevel CMR methods can provide significant acceleration for transport or nodal diffusion problems.

In the 1970s, development began on a new class of multilevel methods known as Diffusion Synthetic Acceleration (DSA) [2]. In DSA, a diffusion-like error equation (obtained from integrating the transport error equation over angle) is solved between iterations on the transport problem, and its solution is used to provide an additive (linear) update to the current estimate of the transport solution. Although variations of DSA exist, standard DSA typically operates on a single (fine) grid, and it is more naturally applied to fixed-source problems than eigenvalue problems. Significant work was performed to apply DSA to various spatial discretizations of the transport problem [46], to analyze its stability [47, 48], and to develop robust iteration schemes using DSA [49, 50, 51, 52]. Although the standard DSA method is not unconditionally stable, it has a broader stability region than CMR and can be applied to a larger class of problems.

DSA led to the development of CMFD, which is arguably the most prominent multilevel method in reactor physics today. The notion of CMFD was first introduced by Smith in 1986 [1].[1] CMFD was developed originally for accelerating the convergence of nodal diffusion methods, but its use was eventually extended to both deterministic and Monte Carlo transport [3, 55, 56]. Unlike DSA, CMFD has a multiplicative (nonlinear) update of the transport equation, which makes it sensitive to the presence of negative fluxes. Moreover, it typically uses a coarsened spatial grid for the low-order diffusion problem, and it is naturally applied to both fixed-source and eigenvalue problems. However, like DSA, the standard CMFD method is unstable for sufficiently large spatial cell sizes. DSA and CMFD are closely related methods, and a detailed comparison of DSA and CMFD can be found in [57]. There, it is shown that linearizing the CMFD method yields a coarse-mesh form of DSA.

In recent years, various attempts have been made to stabilize CMFD [58, 59, 60]. The most fruitful of these was the development of the optimally diffusive CMFD (odCMFD) method, which provided a simple formulation for an unconditionally stable CMFD scheme with a nearly optimal spectral radius [5]. This is the CMFD method used by MPACT; more details regarding odCMFD and its optimization can be found in [5, 61].

Although the standard approach in CMFD is to use two-layered approach in which one accelerates transport on a fine spatial grid with diffusion on a coarsened spatial grid, many attempts were

[1] [1] does not formulate the CMFD problem in its current standard formulation. As noted by [53], the first "modern" formulation of CMFD can be found in [54].

also made to extend CMFD to a multilevel scheme. In [3], the multigroup diffusion problem in CMFD is further coarsened into a two-group diffusion problem to reduce the cost of performing pinwise multigroup CMFD. Similar approaches are taken in [62, 63].

Aside from the two-group systems used in multilevel CMFD techniques, most of the discussion thus far has focused on spatial coarsening. We now turn our attention to multilevel methods in which the energy grid is coarsened. One of the most notable early attempts to use multiple grids in energy is the two-grid method developed by Adams and Morel [64]. In this method, a one-group diffusion problem is used in a multigrid-like manner to correct errors from the transport problem, and infinite-medium problems are solved for each material composition to obtain interpolation operators that rapidly resolve the energy-dependence due to scattering. In recent years, this method has been applied (with modifications) to the Denovo code [65], and it has inspired the multilevel-in-energy low order nonlinear diffusion acceleration (LONDA) method developed by Cornejo and Anistratov [66].

In the radiative transfer community, "grey" (one-group) problems are frequently used to accelerate the convergence of multigroup problems [67, 68, 69, 70, 71]. The primary purpose of the grey system is to provide a lower-order system in which the volatile material temperatures, opacities, and thermal emissions can be resolved without the high cost of simultaneously accounting for all of the multigroup and transport effects. In the reactor physics community, the common view has been that two groups are necessary to perform meaningful work on light water reactor (LWR) problems. The reasoning for this is that there are two energy ranges with distinct phenomena – neutrons are born from fission in the fast energy range, but (in LWRs at least) they can only induce fission in the thermal energy range. As a result, many people believe that a separation of these two ranges is necessary to obtain efficient solutions. However, from results in some of the aforementioned works (CMR, two-grid in energy, LONDA) and in results for MSED, we see that, even in thermal reactor applications, the use of a grey system is beneficial, and even optimal.

Finally, we make two notes regarding terminology. First, the terms "CMFD problem" and "diffusion problem" are used almost interchangeably in this thesis. This is because, from our experience, we have not had any stability issues with MSED resulting from the introduction of CMFD correction factors, and MSED can be used for either CMFD or diffusion. In this thesis, a "CMFD problem" is just a diffusion problem whose coefficients (material properties) have been adjusted by correction factors.

Second, the terms *multilevel* and *multigrid* can be somewhat vague or inconsistent in scientific literature. In this thesis, we define a multilevel method to be any method that solves a "high-order" problem by leveraging smaller, coarser, or "low-order" versions of that problem. Although many mathematicians and scientists contributed to the development of the family of multilevel methods, our view of multilevel methods is most consistent with those of Brandt [72]. In contrast, we use the term “multigrid” only to refer to the specific established method in the mathematics community in which *error* equation(s) are solved on coarsened version(s) of the original *spatial* grid. Such methods are well defined by texts such as [9]. In particular, we consider multigrid methods to be a subset of multilevel methods, and we consider CMFD and CMR to be multilevel methods that are not multigrid methods. Although MSED has a multigrid solver embedded in the inner loops of its iteration scheme, we consider MSED itself to be a general multilevel method and not a pure multigrid method.

1.4 Comparison of MSED to Historical Methods

The purpose of this section is to place MSED in the context of the methods described in the previous two sections. Here, we describe the influence of the previous methods on the development of MSED and the important features that set MSED apart.

A natural view of MSED is that it is an extension of classic geometric multigrid methods to the energy domain. That is, MSED collapses in both space and energy. A unique challenge posed by our application is that the diffusion system we wish to solve is an eigenvalue problem rather than a standard linear system. This necessitates an extra layer of iteration to converge the eigenvalue and fission source of the system, and an energy-integrated, grey (1-group) diffusion problem is used to accelerate this iteration. This two-level in energy method was inspired by methods from thermal radiative transfer, including the well-established grey acceleration scheme in [67] and the High-Order Low-Order (HOLO) method in [70, 71]. We refer to MSED’s 1-group acceleration scheme as “grey” acceleration because of these influences, despite the fact that the term grey, which stems from the field of astrophysics, is not typically used by reactor physicists.

The spatial collapse in MSED differs from standard multigrid linear solvers due to the multigroup nature of our problem of interest. In MSED, the multigrid linear solver is used on both the grey and multigroup diffusion systems. On the grey diffusion system, the spatial multigrid used by MSED is similar to standard multigrid approaches such as [33]. On the multigroup diffusion system, however, the spatial multigrid used by MSED coarsens, smooths, and interpolates all of the groups simultaneously. To our knowledge, this approach to spatially coarsening multigroup problems has not been tried before.

We noted earlier that another way to view MSED is that it is an extension of CMFD, and one might find it natural to consider MSED a multilevel CMFD technique. It is also possible to view MSED as an extension or variant of the aforementioned MLCMR and MLSR methods. However, MSED differs from these methods in several important aspects [10, 11]. First, the CMFD problem motivating the development of MSED is orders of magnitude larger (in terms of the number of unknowns and the number of processors used) than any of the applications in the CMFD and CMR references from the previous sections. Because of this, the historical methods described in the previous section do not scale well to our problem of interest. Second, unlike the MLCMR and MLSR techniques, MSED operates on a diffusion system in which the unknowns representing the neutron net current have already been eliminated. This simplifies the process of collapsing onto coarser spatial/energy grids. Third, whereas the multilevel CMFD techniques use flux-weighted cross sections and low-order "consistency factors" to generate its group-collapsed equations, MSED uses both flux-weighted cross sections and flux-weighted diffusion coefficients, and does not need additional "consistency factors." That is, the collapse in energy in MSED is similar to that of the MLCMR/MLSR methods, but not multilevel CMFD methods. Finally, the coarsening in space in MSED differs from both CMFD and CMR. In MSED, the spatial variable is collapsed using a standard spatial multigrid approach in which coarse-grid equations are error equations rather than approximations to the original system.

Recent work by Hao et al. has produced a multilevel CMFD method that is more similar to MSED than previous multilevel CMFD methods [73]. Hao's work is similar to the work in this thesis in that a one-group CMFD system instead of a two-group system is used to accelerate the convergence of a multigroup CMFD problem. However, Hao's method uses Generalized Minimal Residual (GMRES) with a novel preconditioner as its linear solver, whereas MSED uses a multigrid linear solver.

Finally, just as we can view MSED as an extension of classical multigrid (in space) methods to the energy domain, we can also view MSED as an extension of multilevel-in-energy schemes to the spatial domain. This is a particularly useful description for comparing the MSED method to the multilevel LONDA method developed by Cornejo et al. In [66], LONDA is described as a "nonlinear multigrid" scheme for the energy domain, used to accelerate the convergence of the transport eigenvalue problem. Although the MSED method presented in this thesis is presented as a diffusion/CMFD solver rather than a transport acceleration scheme, MSED and Cornejo's method traverse between energy grids in exactly the same manner. From the multigroup diffusion equation, both methods use a straightforward flux-weighted collapse to generate quantities on coarsened energy grids. This simple collapse, described in Equations 6.3 and 6.8 of Chapter 6 of this thesis, ensures algebraic consistency between the equations on each energy grid without the need for additional correction term(s).

Unlike Cornejo's method, however, the MSED method presented in this thesis only has two levels in energy and uses a multigrid-in-space linear solver on each level. Regarding the first difference, we note that the MSED method can easily be extended to more than two energy grids (full multigroup and grey). This is discussed briefly in Chapter 6, and we expect that, for problems with a large number of energy groups (e.g., > 100), the use of additional intermediate energy will provide benefits similar to those seen in Cornejo's work. Regarding the second difference, [66] describes the use of a Gauss-Seidel-like method with a BiCGSTAB linear solver as the “smoother” on each energy grid. That is, on each of the energy levels except the coarsest (grey) energy grid, Cornejo’s method sweeps through all of the groups, solving the one-group problem corresponding to each group using a BiCGSTAB linear solver. Using the same terminology as Cornejo, we would say that MSED uses multigrid-in-space as its smoother. On the coarsest grid, Cornejo’s method solves the eigenvalue problem using a Newton scheme described in [74], while MSED uses Wielandt-shifted power iteration in conjunction with a multigrid linear solver. In short, the differences between MSED and Cornejo’s method are the number of energy levels and the manner in which the equations are smoothed/solved on each level.

1.5 Outline

The remaining chapters of this thesis are organized as follows.

In Chapter 2, the theory and background are provided for CMFD and the linear solvers used in this thesis. Starting from the continuous neutron transport eigenvalue problem, various approximations are applied to obtain a formulation for our problem of interest: the multigroup diffusion/CMFD eigenvalue problem. The pertinent numerical methods used in solving multigroup diffusion eigenvalue problems (including smoothers, linear solvers, and eigenvalue iteration schemes) are described to the extent required to understand the work in this thesis. No new theory is developed in this chapter.

Chapter 3 describes the details of the two codes in which MSED is implemented. The first is a 1-D multigroup diffusion code developed for rapid prototyping and testing of ideas for MSED. The second is the well-established 3-D MPACT code. The available solver options in each code are summarized. Descriptions of several problems of interest, which will be reused throughout this thesis work to test MSED, are provided in Appendix C.

Power iteration (PI) is a standard method for solving eigenvalue problems, and it is MPACT’s default approach for solving multigroup diffusion eigenvalue problems. Chapter 4 describes initial efforts to reduce the cost of PI via novel space-dependent Wielandt shift (SDWS) techniques. The Fourier analysis that motivated the development of these WS techniques is shown, and results are presented for both a 1-D diffusion code and for MPACT. Our work with WS techniques ultimately fell short of our goals, as our SDWS techniques had limited benefits for the CMFD solver in MPACT. Nonetheless, this work provided valuable insight to the tradeoff between the condition number of the problem matrix and the spectral radius of PI when a WS is applied.

The limitations of the work described in Chapter 4 necessitated the development of the MSED method. MSED is the focus of this thesis, and a topical overview of the method is provided in Chapter 5. MSED can be viewed as a combination of multilevel-in-energy with multigrid-in-space, and we describe these two components separately in consecutive chapters. In Chapter 6, we describe MSED's multilevel-in-energy approach, which centers on the use of a grey diffusion equation. In Chapter 7, the addition of the multigrid-in-space component to MSED is described. The relative improvement provided by each component is verified via Fourier analyses, results from a 1D diffusion code, and selected 2D and 3D results from the MPACT code. For *3-D, full-core* problems, the introduction of the grey diffusion equation provides a $\sim$4-6x speedup over the default approach in MPACT, while the introduction of the multigrid-in-space solver provides an additional $\sim$2x speedup, leading to an overall speedup of $\sim$8-12x. (These speedups refer to the CMFD solver, not the overall MPACT code.)

In Chapter 8, additional results from MPACT are presented to verify that MSED remains efficient for a variety of different problem types. Chapter 8 includes a discussion of these results as well as a summary of the overall performance of MSED. Finally, potential avenues of future work are described in Chapter 9, and a final summary of the entire thesis is provided in Chapter 10.

CHAPTER 2

Background

In this chapter, the theoretical and practical background needed to understand the work in this thesis is presented. No new theory is presented in this chapter. Section 2.1 begins with the continuous neutron transport eigenvalue problem and describes the approximations and discretizations used to arrive at the multigroup neutron diffusion and CMFD eigenvalue problems that MSED is designed to solve. Section 2.2 introduces three categories of iterative linear solvers – smoothers, Krylov methods, and multigrid – and discusses the importance of condition numbers and preconditioning. Finally, section 2.3 introduces the generalized eigenvalue problem and describes the methods used in this thesis for solving it (namely, power iteration and Generalized Davidson).

2.1 Neutron Transport and Diffusion

2.1.1 Neutron Transport Equation

In its exact, continuous form, the neutron transport eigenvalue equation is given by

$$\begin{aligned}\left[\hat{\boldsymbol{\Omega}} \cdot \nabla + \Sigma_t(\boldsymbol{r}, E)\right] \psi(\boldsymbol{r}, E, \hat{\boldsymbol{\Omega}}) = \frac{1}{k}\frac{\chi(\boldsymbol{r}, E)}{4\pi} \int_0^\infty \nu\Sigma_f(\boldsymbol{r}, E')\phi(\boldsymbol{r}, E')dE' \\ + \int_{4\pi}\int_0^\infty \Sigma_s(\boldsymbol{r}, E' \to E, \hat{\boldsymbol{\Omega}}' \cdot \hat{\boldsymbol{\Omega}})\psi(\boldsymbol{r}, E', \hat{\boldsymbol{\Omega}}')dE'd\hat{\Omega}' \,. \end{aligned} \tag{2.1}$$

Here, the independent variables are $\boldsymbol{r}$, $\hat{\boldsymbol{\Omega}}$, and E; $\boldsymbol{r}$ is the spatial variable, $\hat{\boldsymbol{\Omega}}$ is a unit vector corresponding to a direction of travel, and E is the neutron energy. ψ is the space-, direction-, and energy-dependent angular neutron flux, and ϕ is the space- and energy-dependent scalar flux, defined by

$$\phi(\boldsymbol{r}, E) \equiv \int_{4\pi} \psi(\boldsymbol{r}, E, \hat{\boldsymbol{\Omega}})d\hat{\Omega} \,. \tag{2.2}$$

$\chi(\boldsymbol{r}, E)$ describes the energy-spectrum of neutrons generated by fission, $\Sigma_t(\boldsymbol{r}, E)$ is the total cross section, $\nu\Sigma_f(\boldsymbol{r}, E)$ is the neutron multiplicity multiplied by the fission cross section, and

$\Sigma_s(\boldsymbol{r}, E' \to E, \hat{\boldsymbol{\Omega}}' \cdot \hat{\boldsymbol{\Omega}})$ is the double-differential scattering cross section, which represents the rate at which neutrons scatter from $dE'd\hat{\Omega}'$ to $dEd\hat{\Omega}$ [75,76,77].

In this thesis, we focus on solving the eigenvalue problem rather than the transient problem. Thus, in lieu of a time variable, we have the effective neutron multiplication factor, k, which is used to scale the fission source and artificially create a "steady-state" problem. Moreover, we define

$$\lambda \equiv \frac{1}{k} \,. \tag{2.3}$$

Both λ and k can be considered "eigenvalues" in this eigenvalue problem – it simply depends on what one considers to be the operator in this problem. The operator for which λ is the eigenvalue is the inverse of the operator for which k is the eigenvalue. In this document, we generally use λ instead of k for mathematical convenience, and, unless otherwise stated, the term "eigenvalue" refers to λ, not k. As with all eigenvalue problems, there are multiple solutions; each solution has its own eigenvalue (though there may be repeated eigenvalues) and a corresponding set of eigenvectors. In our research, the only solution of interest is the solution with the smallest value of λ (largest value of k). It is the only solution for which the eigenvector (scalar flux) can be strictly nonnegative, and, consequently, it is the only physical solution. We note that this eigenvector of interest is only unique up to a multiplicative constant (i.e., a normalization factor).

Equation (2.1) is a frequently-used equation in reactor analysis, and its solution has many practical applications. Even without time-dependence, Equation (2.1) is a 6-dimensional equation that requires extensive computational resources to solve, and the development of techniques to accelerate the computation of its solution constitutes a vast field of research. The following subsection describes approximations and discretizations that enable deterministic codes to compute a numerical solution to Equation (2.1).

2.1.2 Approximations and Discretizations

2.1.2.1 Multigroup

The multigroup approximation is a common technique for discretizing the energy variable [77,78]. In the multigroup approximation, the interval representing the range of physically allowable energies, $0 \leq E \leq E_{max}$, is separated into G groups, or subintervals. In this document, we use the convention that energy group g is defined by the interval $E_g < E < E_{g-1}$. Thus, $E_0 = E_{max}$ and $E_G = 0$.

The scalar and angular fluxes are divided into G components, each of which represents the continuous version of the flux integrated over a particular energy group:

$$\psi_g(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}) = \int_{E_g}^{E_{g-1}} \psi(\boldsymbol{r}, E, \hat{\boldsymbol{\Omega}}) dE \,, \tag{2.4}$$

$$\phi_g(\boldsymbol{r}) = \int_{E_g}^{E_{g-1}} \phi(\boldsymbol{r}, E)dE\,. \tag{2.5}$$

Continuous-in-energy cross sections and material properties are replaced with average or flux-averaged quantities in each group. For example, the true total cross section in group g is defined as

$$\Sigma_{t,g}(\boldsymbol{r}) \equiv \frac{\int\limits_{E_g}^{E_{g-1}} \Sigma_t(\boldsymbol{r}, E)\phi(\boldsymbol{r}, E)dE}{\int\limits_{E_g}^{E_{g-1}} \phi(\boldsymbol{r}, E)dE}\,. \tag{2.6}$$

The multigroup form of Equation (2.1), obtained by integration over each group's energy bounds, is a coupled set of G equations given by

$$\begin{aligned}\left[\hat{\boldsymbol{\Omega}} \cdot \nabla + \Sigma_{t,g}(\boldsymbol{r})\right] \psi_g(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}) = \\ \sum_{g'=1}^{G} \int_{4\pi} \Sigma_{s,g'\to g}(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}' \cdot \hat{\boldsymbol{\Omega}})\psi_{g'}(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}')d\hat{\Omega}' + \lambda\frac{\chi_g(\boldsymbol{r})}{4\pi} \sum_{g'=1}^{G} \nu\Sigma_{f,g'}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r})\,.\end{aligned} \tag{2.7}$$

In practice, the multigroup cross sections are approximate; they depend on the continuous energy spectrum of the scalar flux, which is not known unless one has already solved Equation (2.1). Thus, Equation (2.7) produces approximate solutions whose accuracy depends on the group structure and the accuracy of the multigroup cross sections used.

2.1.2.2 Scattering Moments

The neutron scattering cross section is often written as an expansion in the Legendre polynomials [75, 77]:

$$\Sigma_{s,g'\to g}(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}' \cdot \hat{\boldsymbol{\Omega}}) = \sum_{n=0}^{\infty} \frac{2n+1}{4\pi} P_n(\hat{\boldsymbol{\Omega}}' \cdot \hat{\boldsymbol{\Omega}})\Sigma_{sn,g'\to g}(\boldsymbol{r})\,. \tag{2.8}$$

Here, P_n is the n-th Legendre polynomial, and

$$\Sigma_{sn,g'\to g}(\boldsymbol{r}) \equiv \int_{4\pi} P_n(\hat{\boldsymbol{\Omega}}' \cdot \hat{\boldsymbol{\Omega}})\Sigma_{s,g'\to g}(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}' \cdot \hat{\boldsymbol{\Omega}})d\hat{\Omega}'\,. \tag{2.9}$$

Making this substitution in Equation (2.7) yields:

$$\left[\hat{\boldsymbol{\Omega}} \cdot \nabla + \Sigma_{t,g}(\boldsymbol{r})\right] \psi_g(\boldsymbol{r}, \hat{\boldsymbol{\Omega}}) = \lambda\frac{\chi_g(\boldsymbol{r})}{4\pi} \sum_{g'=1}^{G} \nu\Sigma_{f,g'}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r})$$

$$+\sum_{g'=1}^{G}\sum_{n=0}^{\infty}\frac{2n+1}{4\pi}\Sigma_{sn,g'\to g}(\boldsymbol{r})\left[\int_{4\pi}P_n(\hat{\boldsymbol{\Omega}}'\cdot\hat{\boldsymbol{\Omega}})\psi_{g'}(\boldsymbol{r},\hat{\boldsymbol{\Omega}}')d\hat{\Omega}'\right]. \quad (2.10)$$

In MPACT, it is common to assume an isotropic scattering cross section, which means that all but the zeroth scattering moment are neglected. The resulting transport equation is much easier to solve computationally and is given by

$$\left[\hat{\boldsymbol{\Omega}}\cdot\nabla+\Sigma_{t,g}(\boldsymbol{r})\right]\psi_g(\boldsymbol{r},\hat{\boldsymbol{\Omega}}) = \frac{1}{4\pi}\sum_{g'=1}^{G}\Sigma_{s0,g'\to g}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r})+\lambda\frac{\chi_g(\boldsymbol{r})}{4\pi}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r}). \quad (2.11)$$

To offset losses in accuracy, a technique known as Transport-Corrected P_0 (TCP$_0$) is used. With TCP$_0$, the scattering cross section matrix and the total cross section are artificially altered to improve the accuracy of the solution. TCP$_0$ improves the accuracy of reactor physics simulations without the extra cost of using higher Legendre scattering moments. However, it may result in negative cross sections and cause convergence problems for the transport solver. More details about TCP$_0$ and its use in MPACT can be found in [79, 80]. In this thesis, all problems in MPACT are run with TCP$_0$ unless otherwise stated.

2.1.2.3 Discrete Ordinates

To discretize the angular variable in MPACT, the discrete ordinates method, or S_N, is used [76, 77]. In S_N, numerical solutions are computed for the a set of N representative directions of travel, $\{\hat{\boldsymbol{\Omega}}_n\}$, determined by a quadrature approximation. Each direction $\hat{\boldsymbol{\Omega}}_n$ has a corresponding weight w_n that can be used for an approximate integration over the angular variable as follows:

$$\int_{4\pi}f(\hat{\boldsymbol{\Omega}})d\hat{\Omega}=\sum_{n=1}^{N}w_nf(\hat{\boldsymbol{\Omega}}_n). \quad (2.12)$$

For example, in S_N, the scalar flux is given by

$$\phi_g(\boldsymbol{r})=\sum_{n=1}^{N}w_n\psi_{g,n}(\boldsymbol{r}). \quad (2.13)$$

The S_N form of Equation (2.11) can now be written as

$$\left[\hat{\boldsymbol{\Omega}}_n\cdot\nabla+\Sigma_{t,g}(\boldsymbol{r})\right]\psi_{g,n}(\boldsymbol{r})=\frac{1}{4\pi}q_g(\boldsymbol{r}), \quad (2.14)$$

where, for simplicity, we have defined:

$$q_g(\boldsymbol{r}) \equiv \sum_{g'=1}^{G} \Sigma_{s0,g'\to g}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r}) + \lambda \frac{\chi_g(\boldsymbol{r})}{4\pi} \sum_{g'=1}^{G} \nu\Sigma_{f,g'}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r})\,. \tag{2.15}$$

2.1.2.4 Method of Characteristics

The method of characteristics (MOC) is a discretization technique for hyperbolic PDEs in which one solves the PDE along characteristic paths. For the transport equation, these characteristics are straight lines that coincide with the directions given by the quadrature set. In a 2-D problem, each direction in the quadrature set has a corresponding set of characteristics covering the entire domain. These characteristics are separated uniformly by a distance known as the ray spacing. For a characteristic parameterized as $\boldsymbol{r}^{(0)} + s\hat{\boldsymbol{\Omega}}_n$ using an arbitrary reference point $\boldsymbol{r}^{(0)}$ and a distance variable s, the transport PDE can be formulated as an ordinary differential equation:

$$\left[\frac{d}{ds} + \Sigma_{t,g}(\boldsymbol{r}^{(0)} + s\hat{\boldsymbol{\Omega}}_n)\right]\psi_{g,n}(\boldsymbol{r}^{(0)} + s\hat{\boldsymbol{\Omega}}_n) = \frac{1}{4\pi} q_g(\boldsymbol{r}^{(0)} + s\hat{\boldsymbol{\Omega}}_n)\,. \tag{2.16}$$

For a fixed q_g, this equation can be solved exactly by using an integrating factor. In MPACT, the domain is divided up into spatial cells, and Σ_t and q_g are assumed to be constant in each spatial cell. The characteristics are defined such that each spatial cell has a sufficient number of characteristics intersecting it in each direction of the quadrature set. Transport sweeps are performed by moving along each characteristic, with q_g defined from the previous estimate of $\psi_{g,n}$. Once all of the characteristics have been traversed, q_g is updated for each spatial cell, and the process is repeated. This process is known as *source iteration*, and it is the most common method for solving transport problems. More details can be found in [4, 53].

2.1.3 Neutron Diffusion Equation

The multigroup neutron diffusion eigenvalue problem is given by

$$\left[-\nabla \cdot D_g(\boldsymbol{r})\nabla + \Sigma_{t,g}(\boldsymbol{r})\right]\phi_g(\boldsymbol{r}) - \sum_{g'=1}^{G} \Sigma_{s0,g'\to g}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r}) = \lambda\chi_g(\boldsymbol{r}) \sum_{g'=1}^{G} \nu\Sigma_{f,g'}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r})\,. \tag{2.17}$$

This equation is obtained by integrating Equation (2.10) over all directions of travel and then applying Fick's Law:

$$\boldsymbol{J}_g(\boldsymbol{r}) \equiv \int_{4\pi} \hat{\boldsymbol{\Omega}}\psi_g(\boldsymbol{r}, \hat{\boldsymbol{\Omega}})d\hat{\Omega} \approx -D_g(\boldsymbol{r})\nabla\phi_g(\boldsymbol{r})\,. \tag{2.18}$$

Here, the $D_g(\boldsymbol{r})$ are diffusion coefficients and are often approximated as either $1/\Sigma_t$ or $1/\Sigma_{tr}$, where Σ_{tr} is the "transport cross-section." In problems with TCP$_0$, the transport cross-section is typically the transport-corrected total cross section.

Fick's Law is a reasonable approximation for many reactor problems, but may be inaccurate near problem boundaries, near material boundaries, and in regions with strongly absorbing material and/or highly anisotropic scattering [75, 77]. In particular, the diffusion equation often adequately captures the overall flux shape and energy spectrum, but it provides no information regarding the angular distribution of the neutrons and may not capture the finer details of the spatial distribution (e.g., the spatial distribution of ϕ within a pin cell). Nonetheless, acceleration methods such as CMFD (described in the next subsection) often leverage diffusion-like equations to converge the eigenvalue, the energy-dependence, and the low spatial frequency components of the flux, leaving only the angular-dependence and finer spatial dependencies as work for the transport sweeper. Such methods work well because the slowest converging error mode in source iteration has a weak angular dependence and varies slowly in space [52].

A centered, second-order, finite-difference spatial discretization of Equation (2.17) on a 3-D Cartesian grid is given by

$$\begin{aligned}
&- D_{i+\frac{1}{2},j,k,g}\frac{\phi_{i+1,j,k,g}-\phi_{i,j,k,g}}{\Delta x_{i+\frac{1}{2}}\Delta x_i} + D_{i-\frac{1}{2},j,k,g}\frac{\phi_{i,j,k,g}-\phi_{i-1,j,k,g}}{\Delta x_{i-\frac{1}{2}}\Delta x_i} \\
&\qquad - D_{i,j+\frac{1}{2},k,g}\frac{\phi_{i,j+1,k,g}-\phi_{i,j,k,g}}{\Delta y_{j+\frac{1}{2}}\Delta y_j} + D_{i,j-\frac{1}{2},k,g}\frac{\phi_{i,j,k,g}-\phi_{i,j-1,k,g}}{\Delta y_{j-\frac{1}{2}}\Delta y_j} \\
&\qquad - D_{i,j,k+\frac{1}{2},g}\frac{\phi_{i,j,k+1,g}-\phi_{i,j,k,g}}{\Delta z_{k+\frac{1}{2}}\Delta z_k} + D_{i,j,k-\frac{1}{2},g}\frac{\phi_{i,j,k,g}-\phi_{i,j,k-1,g}}{\Delta z_{k-\frac{1}{2}}\Delta z_k} \\
&\qquad + \Sigma_{t,i,j,k,g}\phi_{i,j,k,g} - \sum_{g'=1}^{G}\Sigma_{s0,i,j,k,g'\to g}\phi_{i,j,k,g'} = \lambda\chi_{i,j,k,g}\sum_{g'=1}^{G}\nu\Sigma_{f,i,j,k,g'}\phi_{i,j,k,g'}\,.
\end{aligned} \tag{2.19}$$

Here, the diffusion coefficients D are face-averaged quantities, Δx_i is the width of spatial cells in the i-th grid width in the x direction, and

$$\Delta x_{i+\frac{1}{2}} \equiv \frac{1}{2}\left[\Delta x_{i+1} + \Delta x_i\right] . \tag{2.20}$$

Δy and Δz are defined in a similar manner.

In matrix notation, Equation (2.19) can written as

$$M\boldsymbol{\phi} = \lambda F\boldsymbol{\phi}\,. \tag{2.21}$$

Here, $\boldsymbol{\phi}$ is the multigroup scalar flux vector[1], and M and F are the matrices representing the left and right sides of Equation (2.19), respectively.

2.1.4 Coarse Mesh Finite Difference (CMFD)

Coarse Mesh Finite Difference (CMFD) is a widely-used acceleration technique for transport problems. In CMFD, an iterative solution technique for the transport problem is accelerated by solving an approximate diffusion system between iterations. The CMFD equation is a modified version of Equation (2.19), obtained by adding a correction factor to each surface of the spatial cell as follows [54]:

$$\begin{aligned} D_{i+\frac{1}{2},j,k,g} \frac{\phi_{i+1,j,k,g} - \phi_{i,j,k,g}}{\Delta x_{i+\frac{1}{2}} \Delta x_i} + \hat{D}_{i+\frac{1}{2},j,k,g} \left(\phi_{i+1,j,k,g} + \phi_{i,j,k,g} \right) + \dots \\ + \Sigma_{t,i,j,k,g} \phi_{i,j,k,g} - \sum_{g'=1}^{G} \Sigma_{s0,i,j,k,g' \to g} \phi_{i,j,k,g'} = \lambda \chi_{i,j,k,g} \sum_{g'=1}^{G} \nu \Sigma_{f,i,j,k,g'} \phi_{i,j,k,g'} \, . \end{aligned} \quad (2.22)$$

For brevity, we have omitted the terms corresponding to five of the six surfaces. Here, the $\hat{D}$ are *CMFD correction factors*, which are redefined after each transport sweep so that the CMFD eigenvalue problem is consistent with the transport eigenvalue problem. Between transport sweeps (source iterations), $\hat{D}$ is updated from the current estimate of the angular neutron flux from the transport problem, the CMFD eigenvalue problem is solved to generate new estimates of the scalar flux and eigenvalue, and the spatial shape of the angular flux is updated to be consistent with that of the CMFD scalar flux. Algorithm 1 briefly describes the CMFD acceleration scheme.

Algorithm 1: CMFD Acceleration

```
1. Compute $\hat{D}$ using the current estimate of the angular flux in the transport system.
2. Solve the CMFD eigenvalue problem to obtain the scalar flux on the CMFD spatial grid
   and the eigenvalue.
3. Scale the angular flux in the transport problem using the new scalar flux from the CMFD
   problem. Also, use the CMFD scalar flux to generate a new fine-grid fission source.
4. Perform an iteration on the transport problem (i.e., a transport sweep). During this sweep,
   the fission source and eigenvalue are fixed.
5. Repeat steps 1-4 until convergence.
```

[1]The scalar form of the scalar flux (ϕ_g) is typeset without bolding, while its vector form is bolded ($\boldsymbol{\phi}$).

It is often the case that the CMFD problem is defined on a coarser (and more structured) grid than the original fine grid problem. In MPACT, each spatial cell in CMFD corresponds to a single pin cell, while the fine (transport) grid divides each pin cell into $O(100)$ smaller cells [4].

Equation 2.22, with or without the presence of CMFD correction factors, is the problem for which MSED has been developed. As mentioned earlier, the two problems are treated in the same manner by MSED. This is possible because, in each of the G multigroup equations in Equation (2.19), there are enough degrees of freedom between the D_g and $\Sigma_{t,g}$ to "absorb" the application of the CMFD correction factors $\hat{D}_g$. That is, it is possible to redefine the D_g and $\Sigma_{t,g}$ such that it is equivalent to Equation (2.22). Thus, both Equation (2.19) and Equation (2.22) have the same coefficient structure, despite the presence of the correction factors in Equation (2.22).

If $\hat{D}_g$ is sufficiently large, it is possible that negative fluxes may appear in the solution to the CMFD system. MSED may not converge in this case, but negative scalar fluxes would cause the CMFD acceleration of transport to fail, regardless of the solver used for the CMFD system. Nonetheless, in our work, we have encountered no issues stemming from large $\hat{D}_g$.

The work in this thesis focuses a specific type of spatial mesh and a specific spatial discretization (given by Equation (2.19)). However, these are common choices for the mesh and spatial discretization of diffusion and CMFD problems due to the regular lattice structure of LWRs. Moreover, much of the theory presented can be generalized to other spatial discretizations and geometries. For example, the multilevel-in-energy component of MSED, described in Chapter 7 can be applied without modification to any multigroup diffusion or CMFD eigenvalue problem, regardless of the geometry or spatial treatment.

2.2 Iterative Linear Solvers

The goal of a linear solver is to solve for $\boldsymbol{x}$ in the linear system

$$A\boldsymbol{x} = \boldsymbol{b}\,. \tag{2.23}$$

Here, $\boldsymbol{x}, \boldsymbol{b} \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$, and $\boldsymbol{b}$ and A are defined by the problem of interest. For the problems we are primarily interested in, solving linear systems directly is prohibitively costly in terms of computational time and memory. One reason for this is that, although A may be sparsely populated (number of nonzeros is $O(n)$, where n is again the number of unknowns), its inverse A^{-1} and"lower-upper" (LU) factorizations are generally dense, and solving the system directly is still an $O(n^3)$ process for all but the simplest problems [81]. Because of this, iterative linear solvers are needed. For sparse linear systems in particular, these iterative solvers take advantage of the fact that multiplication of a sparse matrix with a vector is an operation with complexity $O(n)$. In this section, we provide background on three types of iterative linear solvers: fixed-point iteration schemes (smoothers), multigrid, and Krylov methods.

2.2.1 Fixed-point Iteration Schemes (Smoothers)

For a linear system $A\boldsymbol{x} = \boldsymbol{b}$, we consider fixed-point iteration schemes of the form

$$A_L \boldsymbol{x}^{(l+1)} = \boldsymbol{b} - A_R \boldsymbol{x}^{(l)} \,. \tag{2.24}$$

Here, A_L and A_R are matrices chosen such that $A = A_L + A_R$. The convergence properties of such a scheme can be determined from the eigenvalues of $A_L^{-1} A_R$. The magnitude of the largest eigenvalue determines its asymptotic convergence rate (also known as the spectral radius). If any of the eigenvalues of $A_L^{-1} A_R$ has a magnitude greater than or equal to 1, the method does not generally converge. Many fixed-point schemes have the property that the magnitudes of the eigenvalues corresponding to low-frequency error modes are close to 1, while those corresponding to high-frequency error modes are much less than 1. That is, these fixed-point schemes typically converge high-frequency error modes rapidly, but they are ineffective at removing low-frequency error modes. Such schemes are referred to as *smoothers*.

The *Jacobi* method is one of the simplest and most well-known smoothers. It defines A_L as the diagonal of the matrix A, and it is unconditionally convergent for diagonally dominant systems (systems in which the magnitude of the diagonal entry is greater than the sum of the magnitudes of the off-diagonal entries for each row). Jacobi converges in 1 iteration for diagonal matrices.

Gauss-Seidel (GS) smoother is an extension of Jacobi that defines A_L as the lower triangular portion of the matrix A (including the diagonal). Because A_L is lower triangular for GS, inverting it is only an $O(n^2)$ process (whereas inversion of an arbitrarily filled matrix is generally $O(n^3)$). The stability criteria for the GS is more complex than that of the Jacobi method. GS converges in 1 iteration for cases in which A is lower-triangular (i.e., when $A = A_L$). For diagonally dominant systems, GS is also unconditionally convergent, and its spectral radius is bounded above by the spectral radius for Jacobi. More details regarding the convergence of GS can be found in [82]. In most of the systems of interest in this thesis, GS has a smaller spectral radius than Jacobi. However, each entry of $\boldsymbol{x}^{(l+1)}$ in Equation (2.24) can be computed independently of the other rows for the Jacobi method, making Jacobi more readily parallelizable than GS.

A third approach to smoothing is SOR. For a given smoother in the form of Equation (2.24), SOR is a relaxation mechanism defined by

$$\begin{aligned} \boldsymbol{x}^{(l+1)} &= (1-\omega)\boldsymbol{x}^{(l)} + \omega \boldsymbol{x}^{(l+\frac{1}{2})} \\ &= (1-\omega)\boldsymbol{x}^{(l)} + \omega A_L^{-1} \left[\boldsymbol{b} - A_R \boldsymbol{x}^{(l)} \right] \,. \end{aligned} \tag{2.25}$$

Typically, SOR is applied to either GS or Jacobi. When $\omega = 1$, Equation (2.25) is the same as Equation (2.24). Choosing $\omega < 1$ results in under-relaxation, which can help stabilize a divergent smoother. Choosing $\omega > 1$ results in over-relaxation, which can accelerate slowly-converging smoothers in certain situations. However, finding a choice of ω with tangible improvements to the convergence properties is challenging and only practical for select model problems [83, 84]. We do not use under-relaxation or over-relaxation in the work in this thesis.

There are two commonly-used variants of the aforementioned smoothers that are of particular interest to the work in this thesis. First, one can choose to perform GS or SOR with a "*red-black ordering*," resulting in a smoother referred to as either red-black GS, red-black Jacobi, or red-black SOR. The motivation for red-black ordering stems from the structure of Equation (2.19). In Equation (2.19), the solution estimate at iteration $(l+1)$ at cell i, j, k only depends on the solution estimates of its six immediate neighbors at iteration (l). If we color all of the cells in the domain in a checkerboard-like manner, then, in a GS smoothing step, the red cells only need data from the black cells, and vice versa. Since there is no dependence between any of the red cells, all of the red cells can be updated in parallel at once. Then, all of the black cells can be updated in parallel from the information in the updated red cells. Thus, red-black GS is a highly parallelizable smoothing algorithm; at any given step, half of the cells can be updated simultaneously. Generally, one obtains a convergence rate similar to that of GS while retaining most of the parallelizability of Jacobi. The primary downside to red-black ordering is that its performance may be hindered by cache misses in larger problems [85].

A second way to modify the standard smoothers is to apply them in a block-like manner. For example, if a matrix A can be separated into blocks B_{ij},

$$A = \begin{bmatrix} B_{11} & B_{12} & \dots & B_{1m} \\ B_{21} & B_{22} & \dots & B_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ B_{m1} & B_{m2} & \dots & B_{mm} \end{bmatrix}, \tag{2.26}$$

then we can define a block Jacobi method in which A_L is the block diagonal of A:

$$A_{L,\text{block Jacobi}} \equiv \begin{bmatrix} B_{11} & & & \\ & B_{22} & & \\ & & \ddots & \\ & & & B_{mm} \end{bmatrix}. \tag{2.27}$$

Computing the inverse of A_L in this case would require one to invert each block matrix. Often, these block inverses are computed approximately. For multigroup diffusion or CMFD equations, one can organize the blocks such that each block corresponds to the set of G equations for a single spatial cell; such an approach is useful if these blocks can be inverted or solved in an efficient manner. The same approach is used to obtain block-versions of GS and SOR [13].

More details regarding smoothers and their convergence properties can be found in any numerical linear algebra textbook such as [13].

2.2.2 Krylov Methods

For a matrix A and a seed vector $\boldsymbol{r}^{(0)}$, its m-dimensional Krylov subspace is defined as

$$\mathcal{K}_m(A, \boldsymbol{r}^{(0)}) \equiv \mathrm{span}\{\boldsymbol{r}^{(0)}, A\boldsymbol{r}^{(0)}, A^2\boldsymbol{r}^{(0)}, \ldots, A^{m-1}\boldsymbol{r}^{(0)}\}\,. \tag{2.28}$$

Krylov methods are a class of iterative solvers that make use of these Krylov subspaces. In particular, these subspaces are often generated by defining $\boldsymbol{r}^{(0)}$ as the initial residual of the system (i.e., $\boldsymbol{r}^{(0)} = \boldsymbol{b} - A\boldsymbol{x}^{(0)}$) [13].

The most discussed Krylov method in this thesis is GMRES. At each iteration l, GMRES seeks a solution estimate whose corresponding residual is orthogonal to $A\mathcal{K}_l(A, \boldsymbol{r}^{(0)})$. By satisfying this condition, the solution estimate produced at iteration l is the one with the minimal residual norm over all vectors from the affine subspace $x_0 + \mathcal{K}_l(A, \boldsymbol{r}^{(0)})$. One particularly desirable property of GMRES is that it is guaranteed to produce the exact solution in n or fewer iterations, where n is the size of the problem [13]. (This is because, after n iterations, we have $A\mathcal{K}_n = \mathbb{R}^n$.)

Because GMRES requires the explicit construction of the Krylov subspace $\mathcal{K}_m$, its memory cost can be high. Moreover, because GMRES must orthogonalize against a growing Krylov subspace, the cost of a GMRES iteration grows as $O(l^2 n)$ where l is the iteration number and n is the number of nonzeros in A. To mitigate these costs, GMRES is often "restarted" after a predefined number of iterations, referred to as the restart length. At each restart, the Krylov subspace is cleared and a new Krylov subspace is seeded with the current residual vector. Restarted GMRES is not guaranteed to converge since the full Krylov subspace is no longer explicitly constructed. A tradeoff occurs in the restart length – smaller restart lengths result in a lower memory and computational burdens, but they reduce the probability of convergence [13].

A popular alternative to GMRES is the Biconjugate Gradient Stabilized (BiCGSTAB) method [13]. As its name suggests, BiCGSTAB is a stabilized version of BiCG, which is an extension of the conjugate gradient (CG) method to non-symmetric problems. Unlike GMRES, BiCGSTAB does not construct the full Krylov subspace. Estimates are computed via a three-term recurrence relation ($\boldsymbol{x}^{(l+1)}$ only depends on information from iterations (l) and $(l-1)$), and storage of earlier residuals or solution vectors is not needed. However, as with restarted GMRES, BiCGSTAB does not always converge. Moreover, BiCGSTAB requires two matrix multiplications per iteration, whereas GMRES only needs one.

More details regarding GMRES and BiCGSTAB, their variants, and their convergence properties can be found in [13].

2.2.3 Multigrid

We have already discussed the history of multigrid methods in Chapter 1. In this subsection, we provide a mathematical formulation of the core multigrid concepts.

First, we remind the reader of the standard multigrid procedure. The motivation for multigrid is that smoothers such as GS or Jacobi rapidly converge high-frequency error modes, but not low-frequency error modes. In multigrid, residual errors from the original fine grid are "restricted" to coarser grids to artificially increase its component frequencies. Smoothing on this coarser grid allows for the rapid convergence of the error modes that appear as high-frequency modes on the coarse grid but not the fine grid. This iteration between smoothing and restriction occurs until a sufficiently coarse grid is reached. At that grid, the problem is solved as exactly as possible (either with a direct solver or many iterations of an iterative solver), eliminating all remaining error modes. The results are then interpolated back up to the fine grid to correct the estimate of the solution [9, 15].

We now introduce some notation for a multigrid method used to solve the linear system $A\boldsymbol{x} = \boldsymbol{b}$:

- q is the V-cycle index (number of V-cycles that have been run).
- $\boldsymbol{x}^{(q)}$ is the estimate of $\boldsymbol{x}$ after q V-cycles.
- $\boldsymbol{x}^{(q+\frac{1}{2})}$ is the estimate of $\boldsymbol{x}$ after q V-cycles and smoothing in the $(q+1)$-th V-cycle.
- p is the variable used for the grid index. $p = 0$ is the coarsest grid and higher values of p correspond to finer grids.[2]
- P is the number of grids (multigrid levels). The original, finest grid is indexed by $p = P - 1$.
- $A_{(p)}$ is the grid-p formulation of the fine-grid matrix operator A. If P is the total number of grids, then $A = A_{(P-1)}$.
- $I_{(p_1)}^{(p_2)}$ is the interpolation/restriction matrix that maps quantities on grid p_1 to quantities on grid p_2 (interpolation if $p_2 > p_1$, restriction if $p_1 > p_2$).
- $\boldsymbol{r}_{(p)}^{(q+\frac{1}{2})}$ is the residual after the smoothing step on grid p in the q-th V-cycle.

[2]This is the opposite of the convention used in [10], but is consistent with the notation in the PETSc library [7].

- $\boldsymbol{r}_{(p)}^{(q)} = I_{(p+1)}^{(p)} \boldsymbol{r}_{(p+1)}^{(q+\frac{1}{2})}$ is the source vector in the q-th V-cycle for the error equation on grid p, obtained by coarsening the residual $\boldsymbol{r}_{(p+1)}^{(q+\frac{1}{2})}$ from the previous, finer grid.

- $\boldsymbol{\varepsilon}_{(p)}^{(q+\frac{1}{2})}$ is the estimate to the solution of the error equation on grid p after smoothing. On grid $p = 0$ (the coarsest grid), this should be the exact solution or close to the exact solution of the error equation.

- $\boldsymbol{\varepsilon}_{(p)}^{(q+1)}$ is the estimate to the solution of the error equation on grid p after correcting with an interpolated error from grid $p-1$. When $p = P-1$, this error estimate is used to correct $\boldsymbol{x}^{(q+\frac{1}{2})}$.

Using the above notation, a typical V-cycle in multigrid is described in Algorithm 2. A visual of the algorithm is provided in Figure 2.1, which has been reproduced from [86]. In this diagram, smoothing is performed during both the restriction steps (left half of the V) and interpolation steps (right half of the V). However, in practice, smoothing during interpolation is often skipped. For simplicity, Algorithm 2 and the above notation assume no smoothing in the interpolation steps.

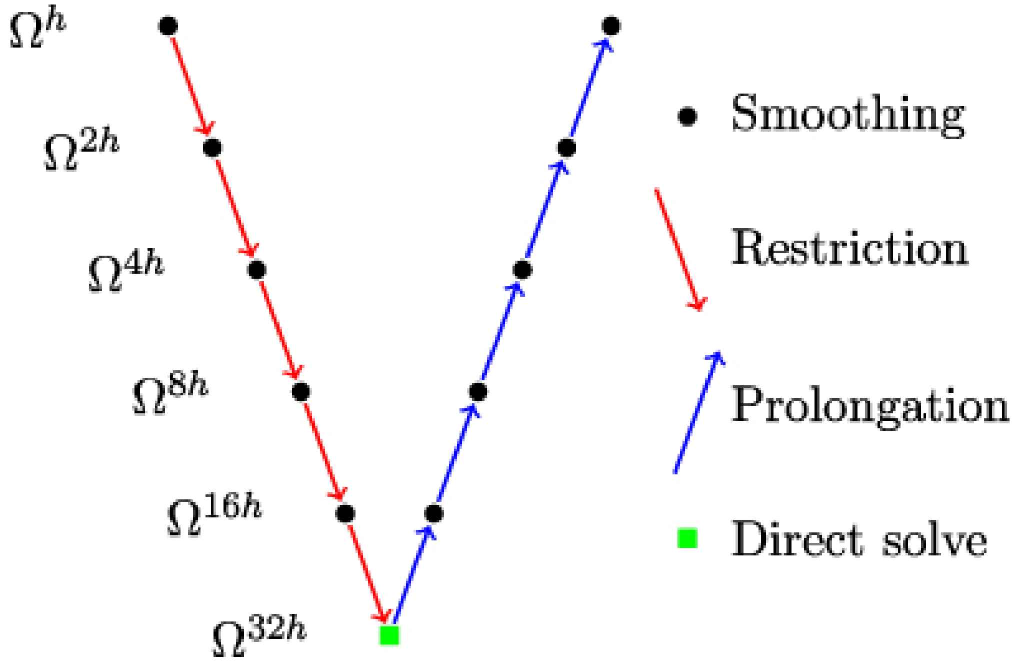


Figure 2.1: An illustration of a typical multigrid V-cycle [86]. We note that prolongation and interpolation are synonymous in multigrid terminology.

Though Algorithm 2 describes the typical process of a multigrid V-cycle, it has not defined several important aspects: how to choose a coarse grid, how to define the interpolation/restriction operators $I_{(p\pm1)}^{(p)}$, and how to define the coarse grid operators $A_{(p)}$. The optimal choices for these parameters depends on the geometry and physics of the problem of interest.

Algorithm 2: V-cycle for solving the fixed-source diffusion problem $A\boldsymbol{x} = \boldsymbol{b}$.

Input: $\boldsymbol{x}^{(q)}$
Result: $\boldsymbol{x}^{(q+1)}$

1. Perform one smoothing step on the original fine grid ($p = P-1$) to obtain $\boldsymbol{x}^{(q+\frac{1}{2})}$.

2. Compute the residual on the original grid:

$$\boldsymbol{r}_{(P-1)}^{(q+\frac{1}{2})} \equiv \boldsymbol{b} - A\boldsymbol{x}^{(q+\frac{1}{2})} . \tag{Alg2.1}$$

3. Traverse down the V-cycle. That is, for $p = P-2, \ldots, 1$:

   (a) Restrict the residual from grid $p+1$ to grid p:

$$\boldsymbol{r}_{(p)}^{(q)} = I_{(p+1)}^{(p)} \boldsymbol{r}_{(p+1)}^{(q+\frac{1}{2})} . \tag{Alg2.2}$$

   (b) Perform a smoothing step on the grid p error equation,

$$A_{(p)}\boldsymbol{\varepsilon}_{(p)} = \boldsymbol{r}_{(p)}^{(q)} , \tag{Alg2.3}$$

   to obtain $\boldsymbol{\varepsilon}_{(p)}^{(q+\frac{1}{2})}$, an uncorrected estimate of $\boldsymbol{\varepsilon}_{(p)}$.

   (c) Compute the residual of the grid p error equation:

$$\boldsymbol{r}_{(p)}^{(q+\frac{1}{2})} \equiv \boldsymbol{r}_{(p)}^{(q)} - A\boldsymbol{\varepsilon}_{(p)}^{(q+\frac{1}{2})} . \tag{Alg2.4}$$

4. Restrict the residual from grid $p = 1$ to the final grid, $p = 0$. Solve the error equation exactly on this grid to obtain $\boldsymbol{\varepsilon}_{(0)}^{(q+1)}$.

5. Traverse up the V-cycle. That is, for $p = 1, \ldots, P-2$, interpolate the estimate of the error from grid $p-1$ to grid p. Use this result to update the error estimate on grid p and obtain $\boldsymbol{\varepsilon}_{(p)}^{(q+1)}$:

$$\boldsymbol{\varepsilon}_{(p)}^{(q+1)} = \boldsymbol{\varepsilon}_{(p)}^{(q+\frac{1}{2})} + I_{(p-1)}^{(p)} \boldsymbol{\varepsilon}_{(p-1)}^{(q+1)} . \tag{Alg2.5}$$

6. Interpolate $\boldsymbol{\varepsilon}_{(P-2)}^{(q+1)}$ to the original grid, and update $\boldsymbol{x}^{(q+\frac{1}{2})}$:

$$\boldsymbol{x}^{(q+1)} = \boldsymbol{x}^{(q+\frac{1}{2})} + I_{(P-2)}^{(P-1)} \boldsymbol{\varepsilon}_{(P-2)}^{(q+1)} . \tag{Alg2.6}$$

Typical choices for the aforementioned parameters are described as follows [9]. The "textbook" geometric multigrid approach for Laplace-like problems (i.e., problems whose form is similar to $\triangle f = u$) on uniform grids is to choose a coarse grid by removing every other spatial index in each direction. An example of this for a uniform array of grid points in 2-D is shown in Figure 2.2. For points on the fine-grid that correspond to points on the coarse-grid, interpolation is typically performed by injection. (With injection, one simply uses the value of the corresponding coarse-grid point.) For points on the fine-grid that do not correspond to points on the coarse-grid, linear interpolation is performed using the nearest coarse-grid points. The restriction operator is typically chosen as the transpose of the interpolation operator:

$$I_{(p+1)}^{(p)} \equiv \left[I_{(p)}^{(p+1)} \right]^T . \tag{2.29}$$

This relation between the interpolation and restriction operators leads to several desirable properties for the multigrid operator, which are discussed in [9] and [15]. Finally, on problems for which spatial discretization is straightforward, $A_{(p)}$ is defined by simply rediscretizing the problem on the coarse grid p.

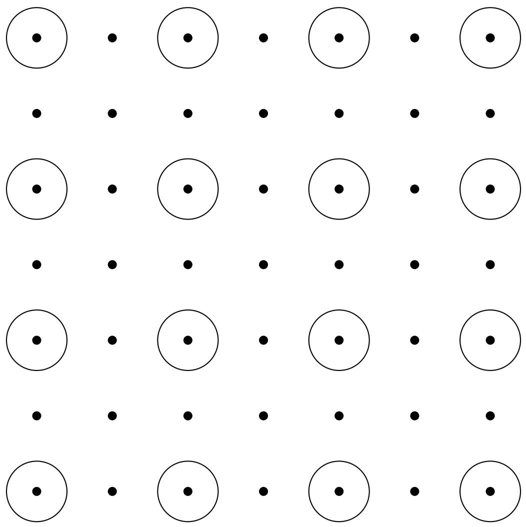


Figure 2.2: A sample coarsening for a uniform 2-D Cartesian grid. Dots correspond to points on the fine grid while circles correspond to coarse-grid points. Here, the set of coarse-grid points is a subset of the original fine-grid.

In the case of nonuniform grids, linear interpolation can still be performed using the nearest coarse-grid points, resulting in nonuniform interpolation weights in $I_{(p)}^{(p+1)}$. Restriction can still be defined as the transpose of interpolation. As noted earlier in Chapter 1, Alcouffe showed that it was possible to deal with heterogeneous diffusion coefficients by treating them as effective scalings of the spatial cell widths [34]. We have not found this to be necessary for the problems in MPACT, but we recommend that anybody who tries to use MSED on problems with strongly discontinuous diffusion coefficients consider the work by Alcouffe in [34].

Lastly, we discuss the notion of algebraic multigrid in the context of the formulations in this section. One of the defining features of algebraic multigrid is the definition of the coarse-grid operator via a triple matrix product:

$$A_{(p)} = I_{(p+1)}^{(p)} A_{(p+1)} I_{(p)}^{(p+1)} . \tag{2.30}$$

This is often referred to as the *Galerkin process* or the *Galerkin triple product*, and multigrid methods that use Equation (2.30) are said to be of the Galerkin type [17]. When the coarse-grid operators are defined using Equation (2.30), knowledge of a physical grid is not necessary. With Equations 2.29 and 2.30, the only component needed for Algorithm 2 to work is the definition of the interpolation operators $I_{(p)}^{(p+1)}$. In algebraic multigrid libraries, it is possible to construct these operators solely from the entries of the matrix A, but some geometric input is often used to improve the convergence rate.

In general, multigrid methods should be as geometric in nature as possible, as these methods have reduced costs and better convergence rates. However, algebraic multigrid concepts provide robustness and allow for the extension of multigrid methods beyond the "textbook" cases. In MSED, a combination of both geometric and algebraic multigrid is used. The Galerkin triple product is used to define coarse-grid operators, allowing us to easily handle cases in which pin cells are misaligned, but the interpolation and restriction operators used are determined entirely from the problem geometry. More details regarding the implementation of spatial multigrid in MSED is provided in Chapter 7.

2.2.4 Condition Number and Preconditioning

For a given matrix A and matrix norm $\|\cdot\|_p$, its *condition number* is defined as

$$\kappa_p(A) \equiv \|A\|_p \|A^{-1}\|_p \,. \tag{2.31}$$

The condition number is an indication of the sensitivity of a solution $\boldsymbol{x}$ in the linear system $A\boldsymbol{x} = \boldsymbol{b}$ to perturbations of the source $\boldsymbol{b}$. It is an important quantity in numerical linear algebra because most iterative solvers suffer a breakdown in their convergence rate when $\kappa_p(A) \to \infty$. Even when the convergence rate does not suffer or a direct method is used, the precision limits of computers can prevent the calculation of accurate solutions for problems with ill-conditioned matrices (i.e., matrices with large condition numbers) [13].

For systems with a finite-difference spatial discretization of the Laplace operator, the condition number is of particular concern. In these problems, $\kappa \to \infty$ as the number of mesh elements goes to ∞ [87]. Geometric multigrid methods do not usually suffer a degradation in convergence rate as the problem size grows [15], despite the growth of the condition number. For this reason, multigrid is particularly appealing for systems with finite-difference spatial discretizations.

We note that the definition of a condition number depends on the norm used. In many cases, the induced 2-norm $\|\cdot\|_2$ is used. This norm is defined by [13]

$$\|A\|_2 \equiv \sup_{\|\boldsymbol{x}\|_2} \|A\boldsymbol{x}\|_2 \,, \tag{2.32a}$$

$$\|\boldsymbol{x}\|_2 \equiv \left(\sum_{i=1}^{n} x_i^2 \right)^{-1/2} \,. \tag{2.32b}$$

In this thesis, $\|\cdot\|$ is the 2-norm $\|\cdot\|_2$ defined by Equations (2.32). Another frequently used norm is the 1-norm, defined for vectors by

$$\|\boldsymbol{x}\|_1 \equiv \sum_{i=1}^{n} |x_i| \,. \tag{2.33}$$

The convergence rate of Krylov methods is determined by the condition number of the problem matrix. Typically, the decay rate of the residual norm approaches 1 as $\kappa \to \infty$. For example, in the case where A is symmetric and positive definite, the residual norm satisfies the following bound [88]:

$$\|\boldsymbol{r}^{(l)}\| \leq \left[\frac{\sqrt{[\kappa_2(A)]^2 - 1}}{\kappa_2(A)} \right]^l \|\boldsymbol{r}^{(0)}\| \,. \tag{2.34}$$

Because of this, Krylov methods are typically ineffective unless a good *preconditioner* is applied. A preconditioner is any matrix P^{-1} that serves as an approximation to A^{-1}. With a preconditioned method, one solves a preconditioned linear system

$$P^{-1}A\boldsymbol{x} = P^{-1}\boldsymbol{b} \tag{2.35}$$

in which the new problem matrix $P^{-1}A$ has a lower condition number than A. (The above formulation describes left-preconditioning. A discussion of right-preconditioning, and its advantages and disadvantages over left-preconditioning, can be found in [89].)

The best preconditioners are those that draw from the physics of the problem. However, as noted in Chapter 1, it is difficult to construct such preconditioners for elliptic problems on highly parallel computing architectures. Solutions are "global" (each point depends strongly on every other point), and computing an effective preconditioner necessitates a significant amount of inter-processor communication. One commonly used "global" preconditioner is ILU, which is an approximation of the exact LU factorization technique in which A is decomposed as a product of a lower-triangular matrix (L) and an upper triangular matrix (U). In ILU, L and U are chosen such that they preserve the sparsity structure of the original matrix A. Because "global" preconditioners such as ILU require significant inter-processor communication, "local" preconditioners such as block Jacobi are often used for parallel applications [7]. These simpler preconditioners do not reduce the condition number as much as ILU, but they are easier to implement and require minimal inter-processor communication.

Lastly, we note the parallels between preconditioning and multilevel schemes. It is often the case that the best preconditioners are drawn from simpler versions of the original problem. In fact, one can view multilevel methods such as CMFD or multigrid as preconditioning schemes. In CMFD, the diffusion operator is used as a preconditioner to the transport operator. In multigrid, coarse-grid versions of the original operator are used as preconditioners to the original fine-grid problem.

In fact, a popular technique is to use algebraic multigrid as a preconditioner in conjunction with Krylov methods such as GMRES. This approach is often used in problems with complex geometries or features, for which multigrid (geometric or algebraic) does not have optimal convergence. In an ideal multigrid setting, smoothing steps and solves on the coarse grid rapidly converge low-frequency error components, while smoothing on the fine grid converges the high-frequency error components. As the complexity of a problem increases, however, certain error modes may remain large throughout the multigrid cycle, thus hindering the performance of multigrid. The use of a Krylov method as an "outer solver" can provide significant improvements in such cases by converging these error modes [15].

2.3 Eigenvalue Solvers

2.3.1 What is an eigenvalue problem?

The problem of interest in this thesis is the multigroup diffusion *eigenvalue* problem. Unlike standard linear systems $A\boldsymbol{x} = \boldsymbol{b}$, general eigenvalue problems in reactor physics are of the form:

$$M\boldsymbol{\phi} = \lambda F\boldsymbol{\phi}\,. \tag{2.36}$$

In this general eigenvalue problem, it is common for only M to be invertible. That is, F may not have full rank. This is the case with the multigroup diffusion or CMFD eigenvalue problems, in which F represents the fission matrix. The goal of an eigenvalue problem is to compute the eigenvalue(s) λ and obtain the corresponding eigenvectors. Even for a single eigenvalue λ, these eigenvectors are only unique up to a constant. If $\boldsymbol{\phi}$ is an eigenvector for some eigenvalue λ (i.e., λ and $\boldsymbol{\phi}$ satisfy Equation (2.36)), then, for any constant scalar c, $c\boldsymbol{\phi}$ is also an eigenvector with eigenvalue λ.

Two important distinctions make eigenvalue problems more difficult to solve than linear systems. First, there are generally multiple eigenvalues λ. In reactor physics, the λ of interest is typically the smallest λ. It is the only λ whose eigenspace (span of the corresponding eigenvectors) consists of solutions that do not change signs (i.e., solutions that are entirely nonnegative or entirely nonpositive).

Second, an iterative method is needed for all but the most trivial eigenvalue problems. The most "direct" approach for computing the eigenvalues consists of computing the determinant of $M - \lambda F$ and finding the values of λ that make the determinant zero. However, this approach is unfeasible for most problems because (1) computing the determinant is computationally expensive, (2) the determinant will be an n-th degree polynomial of λ (where n is the number of unknowns in the system), and (3) the roots of an n-th degree polynomial cannot be determined analytically. The third reason implies that this "direct approach" is actually iterative, since an iterative root-finder is needed for all but the most trivial eigenvalue problems. More importantly, the "direct approach" is an unstable algorithm and should be avoided, even if use of computational resources is not a concern. Its instability can result in inaccurate eigenvalues due to the precision limits of computers [81].

In the following subsections, we describe the power iteration (PI) and Generalized Davidson (GD) methods for solving eigenvalue problems.

2.3.2 Power Iteration

2.3.2.1 Standard Power Iteration (Unshifted Inverse Iteration)

Power iteration (PI) is a standard technique for obtaining the smallest eigenvalue and its corresponding normalized eigenvector in systems of the form of Equation (2.36). It is defined as follows:

$$M\boldsymbol{\phi}^{(l+\frac{1}{2})} = \lambda^{(l)} F \boldsymbol{\phi}^{(l)} \,, \tag{2.37a}$$

$$\lambda^{(l+1)} = \lambda^{(l)} \frac{\|F\boldsymbol{\phi}^{(l)}\|}{\left\|F\boldsymbol{\phi}^{(l+\frac{1}{2})}\right\|} \,, \tag{2.37b}$$

$$\boldsymbol{\phi}^{(l+1)} = \left\|\boldsymbol{\phi}^{(l+\frac{1}{2})}\right\|^{-1} \boldsymbol{\phi}^{(l+\frac{1}{2})} \,. \tag{2.37c}$$

Here, and throughout this paper, l is used as the iteration index for PI. In Equation (2.37c), a 1-norm or 2-norm is typically used.

Algorithm 3: A standard PI step.

```
    Input: $\boldsymbol{\phi}^{(l)}$
    Result: $\boldsymbol{\phi}^{(l+1)}$
1. Solve for $\boldsymbol{\phi}^{(l+\frac{1}{2})}$ using Equation (2.37a).
2. Update $\lambda^{(l+1)}$ using Equation (2.37b).
3. Normalize $\boldsymbol{\phi}$ using Equation (2.37c).
```

PI is guaranteed to converge to the smallest eigenvalue of the system. It also converges to a unique eigenvector if the corresponding eigenspace for the smallest eigenvalue is spanned by a single eigenvector. (This is true for the multigroup diffusion eigenvalue problem.)

Algorithm 3 converges at a rate equal to the dominance ratio of the system:

$$\text{DR} = \frac{|\lambda_2|}{|\lambda_1|} \tag{2.38}$$

Here, λ_i is the i-th smallest eigenvalue of the system in magnitude. (λ_1 is the smallest eigenvalue.) Unfortunately, for most problems in reactor physics, the dominance ratio of the system is close to 1, and PI converges too slowly for practical use.

2.3.2.2 Wielandt Shift (Shifted Inverse Iteration)

To accelerate the slowly converging PI scheme, a *Wielandt shift* (WS) is typically applied. The PI scheme with WS applied is described as follows:

$$\left[M - \lambda'^{(l)} F\right] \boldsymbol{\phi}^{(l+\frac{1}{2})} = \left[\lambda^{(l)} - \lambda'^{(l)}\right] F\boldsymbol{\phi}^{(l)} \,, \tag{2.39a}$$

$$\lambda^{(l+1)} = \lambda'^{(l)} + \left[\lambda^{(l)} - \lambda'^{(l)}\right] \frac{\|F\boldsymbol{\phi}^{(l)}\|}{\left\|F\boldsymbol{\phi}^{(l+\frac{1}{2})}\right\|} \,, \tag{2.39b}$$

$$\boldsymbol{\phi}^{(l+1)} = \left\|\boldsymbol{\phi}^{(l+\frac{1}{2})}\right\|^{-1} \boldsymbol{\phi}^{(l+\frac{1}{2})} \,. \tag{2.39c}$$

The application of a WS can significantly reduce the dominance ratio of a system when the provided shift parameter λ' is close to the eigenvalue of interest (in our case, this is the smallest eigenvalue). This presents a “chicken-and-egg” problem. In order to achieve a minimum spectral radius in Algorithm 4, one must choose λ' as close to λ as possible. However, this λ is not known unless the problem has already been solved. Moreover, one must be careful not to over-shift. Whereas Algorithm 3 converges to the smallest eigenvalue (eigenvalue closest to zero), Algorithm 4 converges to the eigenvalue closest to λ'. If λ' is too large, then Algorithm 4 no longer converges to the smallest eigenvalue [37]. Because the λ of interest in multigroup diffusion problems is the smallest eigenvalue of the system, most implementations of WS try to ensure that $\lambda' < \lambda$ to guarantee the convergence of Algorithm 4 to the correct solution.

Algorithm 4: A WS-accelerated PI step.

```
Input: $\boldsymbol{\phi}^{(l)}$
Result: $\boldsymbol{\phi}^{(l+1)}$
1. Determine an appropriate WS parameter $\lambda'^{(l)}$.
2. Solve for $\boldsymbol{\phi}^{(l+\frac{1}{2})}$ using Equation (2.39a).
3. Update $\lambda^{(l+1)}$ using Equation (2.39b).
4. Normalize $\boldsymbol{\phi}$ using Equation (2.39c).
```

In practice, two approaches are taken: (1) choosing some reasonable fixed value for λ', and (2) defining $\lambda'^{(l)}$ as a function of the current eigenvalue estimate $\lambda^{(l)}$. In the Purdue Advanced Reactor Core Simulator (PARCS) code, the latter approach is taken and the shift is determined by the following empirical formula:

$$\lambda_P'^{(l)} \equiv \max\left\{\lambda^{(l)} - c_1 \left|\lambda^{(l)} - \lambda^{(l-1)}\right| - c_0, \lambda_{\min}\right\} \,. \tag{2.40}$$

Here, c_1 and c_0 are user-defined constants (with typical values of 10 and either 0.01 or 0.02, respectively) while $\lambda_{\min}$ is chosen such that it is physically impossible for λ to be less than $\lambda_{\min}$ [90]. The purpose of c_1 is to gauge how well converged the current estimate of λ is, while the purpose of c_0 is to prevent $\lambda = \lambda'$. For reactor problems, we typically choose $\lambda_{min} = 1/3$. The PARCS shift is designed to produce a shift as close to the true eigenvalue as possible, while minimizing the risk of λ' exceeding the true λ.

In both Algorithms 3 and 4, a linear system must be solved for every power iteration step. Because most problem matrices M are large and sparse, iterative linear solvers such as Krylov methods, smoothers, or multigrid are used. With the application of WS, a tradeoff arises due to the use of iterative linear solvers. Although choosing λ' closer to λ results in a reduced spectral radius for PI, this also pushes $M - \lambda' F$ closer to being a singular matrix (i.e., a rank-deficient matrix, or a matrix with an infinite condition number), and the number of linear solver iterations required per power iteration increases. This is because $M - \lambda F$ is singular when λ is an eigenvalue of the problem. Even with a direct solver, the numerical precision limits of computers prevent one from choosing λ' arbitrarily close to λ. This tradeoff is explored in detail in Chapter 4.

2.3.2.3 A Note on Terminology for Power Iteration

We note that there is an inconsistency between the nomenclatures for PI-like methods in the fields of reactor physics and applied mathematics. The naming conventions adopted thus far have been those of the reactor physics community. In the general applied mathematics community, the power iteration scheme described in Algorithm 3 would be described as unshifted inverse iteration, while the Wielandt shift accelerated PI scheme in Algorithm 4 would be described as inverse iteration. (In the mathematics community, the use of a shift is usually implied in inverse iteration, so the adjective "shifted" is typically omitted.) Rayleigh quotient iteration [81] is another commonly used term in the mathematics community, and it is used to describe a Wielandt-shifted PI scheme with $\lambda'^{(l)} = \lambda^{(l)}$.

2.3.3 Generalized Davidson

Lastly, we briefly describe the *Generalized Davidson* (GD) Krylov method for solving eigenvalue problems. We note that the Krylov solvers we have discussed so far (GMRES and BiCGSTAB) have been linear solvers. Unlike other Krylov methods, the subspace that GD works with is generated using a shifted and preconditioned matrix $P^{-1}\left[M - \theta F\right]$ rather than $P^{-1}M$ or some form of $M^{-1}F$. Here, θ is the most recent estimate of the eigenvalue and has a similar purpose to that of the Wielandt shift. At each GD iteration, the Rayleigh-Ritz procedure [81] is used to extract estimates of the first few eigenvalues and eigenvectors of the system.

The performance of GD is heavily dependent on the effectiveness of the chosen preconditioner. When chosen well, GD can be significantly faster than standard Wielandt-shifted PI methods [32,91]. The standard Davidson method uses $(D - \theta F)$ as the preconditioner, where D is the diagonal of M, but this is not sufficient for most problems of interest [92]. In MPACT, GD is implemented with an algebraic multigrid preconditioner. One significant drawback to GD is its memory usage. The explicit formation of the pseudo-Krylov subspace and the computation of an effective preconditioner can be highly memory intensive processes.

More information regarding the Rayleigh-Ritz procedure can be found in Lecture 33 of [81]. More information about GD can be found in [92, 93, 94]. GD is described in algorithmic form in [94] for symmetric problems, and the details for extending it to nonsymmetric problems are provided in [92].

CHAPTER 3

Implementation Details

In this chapter, we describe the two codes used to generate the results in this thesis: a 1-D multigroup diffusion code, and MPACT. For each code, we list the solvers against which MSED is compared in Chapters 6, 7, and 8.

3.1 1-D Multigroup Diffusion Test Code

Because MPACT is a large code with many components outside of CMFD, it is not suitable for verifying untested numerical procedures on toy problems. To overcome this issue, we have developed a 1-D test code in Python for solving multigroup diffusion eigenvalue problems. Although this test code only runs in serial and has limited capabilities, it provides a convenient and simple environment in which we can quickly test ideas and verify the efficiency of MSED.

The 1-D test code uses the NumPy and SciPy packages for nearly all of its linear algebra functionality [95, 96]. The only exception is the multigrid linear solver, which is written in Fortran 90 to ensure comparable performance with solvers from SciPy packages. NumPy is used for basic linear algebra operations such as dot products or norms, while SciPy is used for sparse matrix data structures and Krylov solvers.

The test code allows users to choose between various Wielandt shifts, including the novel SDWS techniques introduced in Chapter 4. The code also allows for the use of a grey acceleration scheme (multilevel-in-energy) described in Chapter 6. Finally, the code allows for the comparison of the multigrid linear solver to linear solvers from SciPy, including GMRES and BiCGSTAB. Different preconditioner options are also provided by the SciPy library. Invoking the MSED method in this code is equivalent to turning on the grey acceleration and selecting the multigrid-in-space linear solver for both the grey and multigroup diffusion systems. We note that GD is not available as an eigenvalue solver in this test code, so all eigenvalue iterations are performed with either PI or MSED.

In all of the results using this code, the solution is deemed converged if the change in the eigenvalue and flux is 10^{-6} or smaller between consecutive iterations (either MSED iterations or PIs):

$$\left|\lambda^{(l)} - \lambda^{(l-1)}\right| \leq \epsilon = 10^{-6}\,, \tag{3.1a}$$

$$\left\|\boldsymbol{\phi}^{(l)} - \boldsymbol{\phi}^{(l-1)}\right\|_2 \leq \epsilon = 10^{-6}\,. \tag{3.1b}$$

Moreover, in all cases, the code is initialized with $\lambda^{(0)} = 1$ and a scalar flux $\boldsymbol{\phi}^{(0)}$ that is constant in both space and energy. Results from the test code comparing different Wielandt shifts are shown in Chapter 4. Performance improvements from leveraging two components of MSED – the grey diffusion equation and the multigrid linear solver – are shown in Chapters 6 and 7, respectively. In those chapters, they are compared to different combinations of solver, preconditioner, and Wielandt shift options. Further results from MPACT on a variety of problem types are shown in Chapter 8. A description of the problems solved in this 1-D test code can be found in Appendix B.

3.2 MPACT

MPACT is a code designed to simulate realistic reactor models in 3-D with pin-resolved accuracy. It is the deterministic neutronics component of the larger Virtual Environment for Reactor Applications (VERA) software suite, which provides multiphysics simulation capability for reactor applications [97]. MPACT is written in Fortran 2003 and developed using modern software quality assurance practices [98]. In this section, we describe some of the features of MPACT and summarize the solver options for its CMFD system.

One of the key components that sets MPACT apart from other codes is its cross-section processing capability. MPACT has its own set of cross-section libraries (which are updated and improved periodically) and performs various self-shielding calculations in its cross section processing to improve the accuracy of its solutions. For the results in this thesis, we primarily used MPACT's 47-, 51-, and 252-group libraries [79, 99].

Another component that makes MPACT ideal for our work is its parallel scalability and domain partitioning flexibility. Problems in MPACT can be decomposed into assemblies radially (each assembly can be on a different processor), and they can be fully decomposed axially (each plane can reside on its own processor). This allows users to run full-core or quarter-core reactor problems on thousands of cores, significantly reducing the time required for a solution. For the work in this thesis, MPACT is an excellent platform for verifying the performance of MSED on highly parallel architectures. Communication across processors in MPACT is handled using the Message Passing Interface (MPI) [100].

MPACT is capable of performing both criticality calculations (i.e., computing λ) and critical boron search problems (problems in which the boron concentration is updated between transport sweeps and the code proceeds until λ converges to 1). It also has various transient capabilities [101], but these are not considered in this work. Moreover, MPACT can perform depletion calculations in which a series of criticality or critical boron problems are solved sequentially, with each problem having unique control rod positions and fuel compositions.

Thermohydraulic feedback can be considered by linking to COBRA-TF or using MPACT's internal simplified thermohydraulics model. Xenon feedback can also be considered in its simulations. Both of these feedback mechanisms are handled in MPACT via a basic Picard iteration [102]. Between each transport solve, thermohydraulic quantities are updated using the thermohydraulics code and xenon concentrations are updated (step 5 of Algorithm 5). These quantities are then used to compute updated cross sections. Unfortunately, this simple Picard scheme can be slow or unstable. In the results of Chapter 4 and Section 8.4, we see that improving the CMFD solver can result in a larger overall runtime. Because of the instability and poor convergence of the feedback scheme, the poor convergence of MPACT's current CMFD solver acts as a pseudo-relaxation mechanism in many of the problems of interest. Paradoxically, we see in Section 8.4 that loosening the convergence of the CMFD system can actually lead to an improvement in the spectral radius of the outer feedback solver. Nonetheless, the unusual convergence behavior is a problem with the feedback iteration mechanism, not the CMFD solver. The preferred approach for the future of MPACT is to improve the feedback solver so that it can leverage the converged CMFD solution properly. Improvement of the feedback solver is beyond the scope of the work in this thesis, but others at the University of Michigan are currently working on this issue [103].

Instead of solving a 3-D transport problem, MPACT models the reactor with a 2-D/1-D method in which the axial direction is modeled with approximate diffusion physics and the radial direction is modeled with full transport physics [104, 105]. MOC is used for the spatial discretization in the radial direction, and a nodal discretization is used in the axial direction. Between each transport sweep, CMFD is used for acceleration and a nodal solver is used to update the axial leakage. Algorithm 5 summarizes this process; it is the same as Algorithm 1 except for the presence of the nodal solver.

As noted earlier, MPACT uses the odCMFD method developed by Zhu et al. rather than standard CMFD. Compared to standard CMFD, odCMFD is unconditionally stable and has an improved convergence rate [5]. In odCMFD, the standard diffusion coefficients D are replaced by the modified diffusion coefficients

$$\tilde{D} = D + \Delta\theta \,. \tag{3.2}$$

Here, θ is determined by an empirical optimization of the spectral radius in a Fourier analysis, and Δ is the grid-size in the direction of the face on which D resides. (The definition of $\tilde{D}$ in Equation (3.2)

Algorithm 5: Iteration scheme for solving the neutron transport eigenvalue problem in MPACT.

```
1. Using the most recent estimates of the axial current from the nodal solver and the most
   recent estimates of the angular flux, compute $\hat{D}$ as described in the odCMFD procedure [5].
2. Solve the CMFD eigenvalue problem to obtain the scalar flux on the CMFD spatial grid
   and the eigenvalue.
3. Using the radial currents from CMFD, perform one or more iterations in the nodal diffusion
   solver to resolve the axial dependence of the solution. This produces new coarse-cell scalar
   fluxes and axial currents.
4. Using the most recent scalar fluxes, update the fine-grid angular flux and fission source,
   and perform a transport sweep (source iteration).
5. If feedback is being accounted for, perform the necessary calculations to obtain updated
   temperatures, densities, and xenon and boron concentrations. Use these quantities to
   update the cross section values in each spatial cell.
6. Repeat steps 1-5 until convergence.
```

is taken from [5]. We note that $\tilde{D}$ has a different meaning in the documentation for MPACT; there, it is used to represent the average of two neighboring cell-based diffusion coefficients.)

The purpose of MSED in MPACT is to efficiently solve the (od)CMFD eigenvalue problem in step 2 of Algorithm 5. In the results in Chapters 6 and 7, we primarily compare the performance of MSED in MPACT to the performance of the following four alternatives:

- MPACT's default method, which is standard PI with a fixed WS of $\lambda' = 2/3$ and block-Jacobi preconditioned GMRES as the linear solver,
- standard PI with the PARCS WS (Equation (2.40)) and the same linear solver as the default method,
- standard PI with a fixed WS of $\lambda' = 2/3$ and "red-black" block Jacobi (RBBJ) as the linear solver[1], and
- Generalized Davidson

All of the aforementioned options except GD and RBBJ leverage data structures and routines from the PETSc library [7]. The implementation of GD from the Trilinos library is used [8], and it is preconditioned by an algebraic multigrid preconditioner from the MueLu package [106]. The RBBJ

[1]In MPACT, this solver is referred to as the "MGRBSOR" solver. However, we refer to it in this thesis as RBBJ because we did not use any relaxation in our results.

solver is internal to MPACT, and the maximum number of iterations permitted per linear solve using RBBJ is 150.

In addition to the PARCS and fixed Wielandt shifts, several of the SDWS options described in Chapter 4 have also been implemented in the CMFD solver in MPACT. The results in Chapter 4 include a comparison of the performance of these shifts for the MPACT code.

For the PI methods, the maximum permitted number of Krylov iterations for GMRES is 100 per PI for the "default" method, and it is 200 per PI for any PI scheme that uses an iteration-dependent WS such as the PARCS WS or SDWS. GMRES is preconditioned with a block Jacobi preconditioner; the blocks of this preconditioner are partially solved with an ILU iteration. Appendix A provides more information regarding the different PETSc solvers, preconditioners, and smoothers used for the work in this thesis.

Because MPACT is a transport code with CMFD, there are several layers of convergence criteria. The stopping criterion for the code overall is determined by the transport solution. MPACT considers its transport solution converged if the change in eigenvalue and fission source between transport sweeps is sufficiently small. Mathematically, this criterion is described by:

$$\left|\lambda^{(n)} - \lambda^{(n-1)}\right| \leq 10^{-6}\,, \tag{3.3a}$$

$$\sqrt{\frac{1}{N_{fi}}\sum_i \left(f_i^{(n)} - f_i^{(n-1)}\right)^2} \leq 5\cdot 10^{-5}\,. \tag{3.3b}$$

Here, m is the transport sweep index, N_{fi} is the number of source regions with fission (i.e., the number of spatial cells on the transport grid with fission), and f_i is the normalized fission source in spatial cell i of the transport grid. Between each transport sweep, a CMFD problem is solved and an inner convergence criterion is used to determine when MPACT leaves the CMFD solver and returns to the transport problem. Unless otherwise stated, MPACT leaves the CMFD solver when the eigenvalue between consecutive eigenvalue iterations (i.e., PIs or MSED iterations) is less than 10^{-6} and the residual of the CMFD eigenvalue system has been reduced by a factor of 10^{-2}. A maximum of 20 eigenvalue iterations are permitted per CMFD solve. In some cases, the eigenvalue convergence in MSED is determined by the grey diffusion system rather than the full multigroup CMFD problem; this is discussed later in the results. MPACT is initialized with $\lambda^{(0)} = 1$ and a scalar flux $\boldsymbol{\phi}^{(0)}$ that is constant in both space and energy. A CMFD solve is performed to generate an initial guess for the first transport sweep.

Brief descriptions of the problems run in MPACT for the work in this thesis are provided in Appendix C. More details regarding the capabilities and techniques of MPACT can be found in [4, 98].

CHAPTER 4

Investigation of Wielandt Shifts

In this chapter, we present theory, Fourier analyses, and results for a new concept: the space-dependent Wielandt shift (SDWS). The content in this chapter describes novel work developed over the course of the author's dissertation and is a summary of [107]. Although the development of SDWS did not provide the speedup needed to meet the goals of this thesis, it led to insights on the conditioning of shifted diffusion operators, and this further motivated the development of the MSED method.

4.1 Theory

4.1.1 Terminology: Criticality of a Diffusion Operator

First, we define the terms *subcritical*, *critical*, and *supercritical*, which are used frequently in this chapter to describe shifted (and unshifted) diffusion operators. Physically, a subcritical operator has larger loss terms than gain terms, a critical operator has equal loss and gain terms, and a supercritical operator has larger gain terms than loss terms. The standard diffusion operator (left side of Equation (2.17)),

$$M_g(\boldsymbol{r})\phi_g(\boldsymbol{r}) \equiv \left[-\nabla \cdot D_g(\boldsymbol{r})\nabla + \Sigma_{t,g}(\boldsymbol{r})\right]\phi_g(\boldsymbol{r}) - \sum_{g'=1}^{G} \Sigma_{s0,g'\to g}(\boldsymbol{r})\phi_{g'}(\boldsymbol{r})\,, \tag{4.1}$$

is subcritical for any physical system with nonzero leakage or at least one region with a nonzero absorption cross section. The application of a Wielandt shift (WS) introduces an extra gain term that renders the diffusion operator less subcritical (i.e., closer to critical). Subcritical operators are guaranteed to have nonnegative solutions when the source is positive.[1] However, the same cannot be said for critical or supercritical operators, and one should avoid using a WS that renders the

[1]This fact is a consequence of a generalization of Hopf's Lemma to weakly coupled systems of partial differential equations [108, 109].

diffusion operator critical or supercritical. In Section 2.3.2.2, we noted that the standard approach to choosing λ' for diffusion problems is to choose the shift parameter λ' as close to λ as possible without exceeding λ. This is equivalent to shifting a diffusion operator so that it is as close to critical as possible while remaining subcritical.

More information, including mathematical definitions for these three terms, can be found in [107].

4.1.2 Motivation

As we have discussed, the goal of the work in this thesis is to efficiently solve multigroup diffusion eigenvalue problems. The underlying equation of interest, with no spatial discretization, is described by Equation (2.17):

$$\left[-\nabla \cdot D_g \nabla + \Sigma_{t,g}\right] \phi_g - \sum_{g'=1}^{G} \Sigma_{s0,g' \to g} \phi_{g'} = \lambda \chi_g \sum_{g'=1}^{G} \nu \Sigma_{f,g'} \phi_{g'} \, . \tag{2.17 revisited}$$

All of the coefficients and unknowns except λ are dependent on the spatial variable $\boldsymbol{r}$, but the explicit dependence has been dropped from the notation for brevity. We note that the theory developed in this chapter is independent of the spatial discretization, as Wielandt shifts do not alter the leakage term in Equation (2.17).

The power iteration (PI) scheme described in Section 2.3.2.1 is a standard technique for solving eigenvalue problems, but this technique is insufficient for multigroup diffusion problems in reactor physics due to their high dominance ratios (≥ 0.96). WSs, which were introduced in Section 2.3.2.2, are needed to reduce the spectral radius of PI so that it can converge in a reasonable number of iterations. As noted in Section 2.3.2.2, however, the determination of the shift parameter λ' in WS-accelerated PI is not a trivial task. One must choose a λ' that is reasonably close to the true λ without knowing what λ is. In MPACT, the default approach is to simply fix $\lambda' = 2/3$, but this does not provide an optimal reduction of the spectral radius.

Even if a reasonably accurate estimate of λ is available at the beginning of the iteration scheme, this estimate could not be used as the shift parameter because we do not know if the estimate overestimates λ or underestimates λ. In the former case, using such a shift would result in a supercritical diffusion operator. In principle, one could still improve the PI scheme by merely making $\lambda^{(0)}$ more accurate (e.g., one could volume-average the reactor into a single spatial cell, solve a $G \times G$ eigenvalue problem, and use this result as the initial guess). In most reactor problems, however, the true λ is usually close to 1, and trying to improve upon using an initial guess of 1 is not a fruitful effort.

To generate a good WS parameter without overshifting, many codes use empirical formulas based on the most recent estimates of λ. The example we focus on in this thesis is the following shift from the PARCS code:

$$\lambda_P'^{(l)} \equiv \max\left\{\lambda^{(l)} - c_1\left|\lambda^{(l)} - \lambda^{(l-1)}\right| - c_0, \lambda_{\min}\right\} . \tag{2.40 revisited}$$

The parameters in this equation are defined in the text immediately following Equation (2.40) in Chapter 2.3.2.2. However, such empirical shifts are only effective when $\lambda^{(l)}$ is reasonably well-converged and closely approximates the true λ (i.e., Equation (2.40) may not be effective in the first few iterates).

The question is then: can we improve upon existing WS methods using the physics of the problem rather than empirical formulas? Our attempts to answer this led to the development of the space-dependent Wielandt shift (SDWS). As the name suggests, SDWS differs from conventional WS because the shift parameter in SDWS is allowed to vary in space. More importantly, unlike standard WS, the shift values in SDWS depend on the material properties of the problem of interest.

We note that it is not technically correct to describe SDWS as a "Wielandt shift." Eigenvalues are scalar quantities, and it is unclear what the shifted quantity $\lambda - \lambda'(\boldsymbol{r})$ would represent. One could view SDWS as a matrix shift rather than a Wielandt shift; that is, SDWS tries to solve an eigenvalue problem of the form

$$[M - S]\,\boldsymbol{\phi} = [\lambda F - S]\,\boldsymbol{\phi} \tag{4.2}$$

in lieu of the original problem $M\boldsymbol{\phi} = \lambda\boldsymbol{\phi}$. (With a standard shift, we would have $S = \lambda' F$.) Nonetheless, the primary objective of SDWS is the same as that of standard WS; they are both methods for reducing the dominance ratio of the eigenvalue problem.

In the following two subsections, we describe six variants of SDWS, grouped into two categories: three shifts based on the local eigenvalue (LE) and three shifts based on the local absorption (LA).

4.1.3 Local Eigenvalue Shifts

In this subsection, we describe three SDWS methods that make use of local eigenvalues obtained by solving infinite-medium problems at each spatial point. These three shifts are: the LE shift, the LE Positive Source (LEPS) shift, and the PARCS LEPS (PLEPS) shift. The PLEPS shift is the best of the three shifts in terms of stability and efficiency, but all three shifts are described in this section because the LE and LEPS shifts are needed to understand the derivation of the PLEPS shift.

Since the shifts described in this section are space-dependent, the corresponding accelerated PI procedure is slightly different than that of the standard Wielandt shifted PI scheme in Algorithm 4.

Algorithm 6 describes the adjusted PI scheme for any of the LE-based SDWSs in this subsection, and it reduces to Algorithm 4 if the space-dependence of λ' is removed.

Algorithm 6: A PI step accelerated by an LE-based SDWS for multigroup diffusion eigenvalue problems.

```
Input: $\boldsymbol{\phi}^{(l)}$
Result: $\boldsymbol{\phi}^{(l+1)}$
1. Compute the WS parameter $\lambda'^{(l)}(\boldsymbol{r})$ for all points in space using either Equations (4.3),
   (4.7), or (4.8). (For the LE shift, this only needs to be done once since $\lambda'_{LE}(\boldsymbol{r})$ does not
   change from iteration to iteration.)
2. Solve a fixed-source diffusion problem with the current fission source to obtain $\phi_g^{(l+\frac{1}{2})}(\boldsymbol{r})$:
   $$\left[-\nabla\cdot D_g(\boldsymbol{r})\nabla+\Sigma_{t,g}(\boldsymbol{r})\right]\phi_g^{(l+\frac{1}{2})}(\boldsymbol{r}) - \sum_{g'=1}^{G}\left[\Sigma_{s0,g'\to g}(\boldsymbol{r})-\lambda'^{(l)}(\boldsymbol{r})\chi_g(\boldsymbol{r})\nu\Sigma_{f,g'}(\boldsymbol{r})\right]\phi_{g'}^{(l+\frac{1}{2})}(\boldsymbol{r}) = \left(\lambda^{(l)}-\lambda'^{(l)}(\boldsymbol{r})\right)\chi_g(\boldsymbol{r})\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\boldsymbol{r})\phi_{g'}^{(l)}(\boldsymbol{r})\,, \quad \text{(Alg6.1)}$$
3. Update the eigenvalue:
   $$\lambda^{(l+1)}=\frac{\sum_{g'=1}^{G}\int\nu\Sigma_{f,g'}\left\{\lambda'\phi_{g'}^{(l+\frac{1}{2})}(\boldsymbol{r})+\left[\lambda^{(l)}-\lambda'^{(l)}(\boldsymbol{r})\right]\phi_{g'}^{(l)}(\boldsymbol{r})\right\}d^3r}{\sum_{g'=1}^{G}\int\nu\Sigma_{f,g'}\phi_{g'}^{(l+\frac{1}{2})}(\boldsymbol{r})d^3r}\,. \quad \text{(Alg6.2)}$$
4. Re-normalize the flux using Equation (2.39c).
```

First, we present the *LE shift*, defined by:

$$\lambda'_{LE}(\boldsymbol{r})\equiv\begin{cases}\lambda_\infty(\boldsymbol{r}), & \text{if } \boldsymbol{r} \text{ is in a fissile region,}\\ 0, & \text{if } \boldsymbol{r} \text{ is in a nonfissile region.}\end{cases} \tag{4.3}$$

At points $\boldsymbol{r}$ that are in fissile regions, $\lambda_\infty(\boldsymbol{r})$ is the solution to the following infinite-medium eigenvalue problem:

$$\Sigma_{t,g}(\boldsymbol{r})\phi_{\infty,g}(\boldsymbol{r})-\sum_{g'=1}^{G}\Sigma_{s0,g'\to g}(\boldsymbol{r})\phi_{\infty,g'}(\boldsymbol{r})=\lambda_\infty(\boldsymbol{r})\chi_g(\boldsymbol{r})\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\boldsymbol{r})\phi_{\infty,g'}(\boldsymbol{r})\,. \tag{4.4}$$

Using $G \times G$ matrices and length-G column vectors, we can also write Equation (4.4) as

$$[\Sigma_t(\boldsymbol{r}) - \Sigma_{s0}(\boldsymbol{r})]\,\boldsymbol{\phi}_\infty = \lambda_\infty(\boldsymbol{r})\boldsymbol{\chi}(\boldsymbol{r})\,[\boldsymbol{\nu\Sigma}_f(\boldsymbol{r})]^T\,\boldsymbol{\phi}_\infty\,. \tag{4.5}$$

Here, bolded quantities are vectors while the remaining quantities are matrices or scalars. $\Sigma_t(\boldsymbol{r})$ is a diagonal matrix, and the T denotes the transpose operation. Because $\boldsymbol{\chi}(\boldsymbol{r})\,(\boldsymbol{\nu\Sigma}_f(\boldsymbol{r}))^T$ is a rank-1 matrix, there is only one nonzero eigenvalue, which can be obtained without iteration by the following formula:

$$\lambda_\infty(\boldsymbol{r}) = \left\{[\boldsymbol{\nu\Sigma}_f(\boldsymbol{r})]^T\,[\Sigma_t(\boldsymbol{r}) - \Sigma_{s0}(\boldsymbol{r})]^{-1}\,\boldsymbol{\chi}(\boldsymbol{r})\right\}^{-1}\,. \tag{4.6}$$

The motivation for this shift is that, for homogeneous systems with reflecting or periodic boundary conditions, Algorithm 6 has a spectral radius of zero when Equation (4.3) is used as the shift. This fact is verified by a Fourier analysis in Section 4.3. From an intuitive standpoint, this is true because $\lambda_\infty = \lambda$ in homogeneous systems with reflective or periodic boundary conditions. In those problems, we effectively compute the true eigenvalue and use that as a shift. In realistic problems, heterogeneities and leakage would be present, and we would not have $\lambda = \lambda_\infty$. Nonetheless, the idea behind the LE shift is that $\lambda_\infty(\boldsymbol{r})$ is a reasonable, physically-motivated guess for λ in realistic problems.

From numerical tests on 1-D problems, we have seen that the LE shift can reduce the number of PIs needed for convergence by one or two orders of magnitude (compared to unshifted PI). However, when the LE shift is used, the diffusion operator can be supercritical (overshifted), and the shifted fission source may have negative components. These two factors can result in convergence to the wrong solution, slow convergence, or even divergence.

The convergence problems of the LE shift motivated the development of the *LEPS shift*. The LEPS shift significantly reduces the possibility of overshifting by restricting the shift values from the LE shift:

$$\lambda'^{(l)}_{LEPS}(\boldsymbol{r}) \equiv \min\left\{\lambda^{(l)} - \varepsilon, \lambda'_{LE}(\boldsymbol{r})\right\}\,. \tag{4.7}$$

Here, $\lambda'^{(l)}_{LEPS}(\boldsymbol{r})$ is bounded above by $\lambda^{(l)} - \varepsilon$ to ensure that the fission source in Equation (Alg6.1) is never nonnegative (or close to nonnegative). ε is a small positive quantity used to ensure that the fission source is not identically zero (e.g., $\varepsilon = 0.001$); in most realistic problems, ε can be safely set to 0.

To further improve the performance of LE-based SDWS techniques, we developed the *PLEPS shift*, which combines the LEPS shift with the PARCS shift:

$$\lambda'^{(l)}_{PLEPS}(\boldsymbol{r}) \equiv \max\left\{\lambda'^{(l)}_{PARCS}, \lambda'^{(l)}_{LEPS}(\boldsymbol{r})\right\}\,. \tag{4.8}$$

Here, $\lambda'^{(l)}_{PARCS}$ is defined by Equation (2.40). At each power iteration l, the PLEPS shift looks at both the LEPS and PARCS shifts and uses the more aggressive of the two shifts at each spatial point. At the beginning of the problem, when the initial guess of the eigenvalue may not be optimal, the PLEPS shift primarily leverages the LEPS shift, which is physically-motivated and not solely dependent on the accuracy of the current iterate of the eigenvalue. Towards the end of the problem, the PLEPS shift primarily leverages the PARCS shift, which is effective when the current iterate of the eigenvalue is accurate and well-converged. In the results shown later in this chapter, we see that the number of power iterations required for the PLEPS shift is fewer than that of the LEPS shift or the PARCS shift alone, but the magnitude of this reduction is highly problem-dependent.

There are two primary drawbacks to the LE-based shifts described in this section. First, all LE-based shifts require the computation of the local infinite-medium eigenvalue in every spatial cell, and each of these computations requires solving a $G \times G$ linear system. In realistic reactor problems, each spatial region will have unique cross sections due to feedback, even if material compositions are repeated, and the number of operations required to compute the local eigenvalues will scale as $O(NG^3)$, where N is the number of spatial cells. This overhead cost is particularly large for feedback problems with a large number of groups (e.g., $G = 252$). For problems in which G is relatively low (e.g., $G = 51$), the cost of computing these local eigenvalues is significantly less than the cost of performing an extra power iteration, and the application of an LE-based shift may be beneficial if it reduces the number of PIs by one or more. One could reduce the number of LEs that need to be computed by ignoring the effects of feedback in the computation of the shift, but this degrades the efficiency of the shift and increases the possibility of an overshift.

The second drawback is that LE-based shifts can overshift and result in a supercritical diffusion operator if $\lambda^{(0)}$ is significantly larger than λ. Although the LEPS and PLEPS shifts are bounded by $\lambda^{(l)}$, this bound is only effective if $\lambda^{(l)}$ is not significantly larger than λ. We note that this is a problem for both LE-based SDWSs and the PARCS shift. If $\lambda^{(0)} > \lambda$ and there is an iteration early in the iteration scheme in which the eigenvalue does not change significantly, then the PARCS shift in Equation (2.40) can also result in an overshift. An initial guess of 1 is typically acceptable, but a smaller estimate should be used if one expects that the problem has a λ much smaller than 1.

4.1.4 Local Absorption Shifts

In this subsection, we present three shifts based on the local absorption (LA) cross sections: the LA shift, the LA Positive Source (LAPS) shift, and the PARCS LAPS (PLAPS) shift. Unlike the LE-based shifts, a more general group-dependent quantity $s^{(l)}_{g,g'}(\boldsymbol{r})$ is needed to describe the LA-based shifts. Algorithm 7 describes how these three LA-based SDWSs can be used to accelerate

PI, and it reduces to Algorithm 6 when $s_{g,g'}^{(l)}(\boldsymbol{r})$ takes on the following form:

$$s_{g,g'}^{(l)}(\boldsymbol{r}) = \lambda'^{(l)}(\boldsymbol{r})\chi_g(\boldsymbol{r})\nu\Sigma_{f,g'}(\boldsymbol{r})\,. \tag{4.9}$$

Algorithm 7: A PI step accelerated by an LA-based SDWS for multigroup diffusion eigenvalue problems.

Input: $\phi_g^{(l)}(\boldsymbol{r})$, $\lambda^{(l)}$
Result: $\phi_g^{(l+1)}(\boldsymbol{r})$, $\lambda^{(l+1)}$

1. Compute the WS parameter $s_{g,g'}^{(l)}(\boldsymbol{r})$ for all points in space using either Equations (4.10), (4.13), or (4.14). (For the LA shift, this only needs to be done once since $s_{LA,g,g'}(\boldsymbol{r})$ does not change from iteration to iteration.)
2. Solve a fixed-source diffusion problem with the current fission source:

$$\begin{aligned}\left[-\nabla\cdot D_g(\boldsymbol{r})\nabla + \Sigma_{t,g}(\boldsymbol{r})\right]\phi_g^{(l+\frac{1}{2})}(\boldsymbol{r}) - \sum_{g'=1}^{G}\left[\Sigma_{s0,g'\to g}(\boldsymbol{r}) + s_{g,g'}(\boldsymbol{r})\right]\phi_{g'}^{(l+\frac{1}{2})}(\boldsymbol{r}) \\ = \lambda^{(l)}\chi_g(\boldsymbol{r})\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\boldsymbol{r})\phi_{g'}^{(l)}(\boldsymbol{r}) - \sum_{g'=1}^{G}s_{g,g'}(\boldsymbol{r})\phi_{g'}^{(l)}(\boldsymbol{r})\,,\end{aligned} \tag{Alg7.1}$$

3. Update the eigenvalue:

$$\lambda^{(l+1)} = \frac{\lambda^{(l)}\sum\limits_{g'=1}^{G}\int\nu\Sigma_{f,g'}\phi_{g'}^{(l)}(\boldsymbol{r})d^3r + \sum\limits_{g=1}^{G}\sum\limits_{g'=1}^{G}\int s_{g,g'}(\boldsymbol{r})\left[\phi_{g'}^{(l+\frac{1}{2})}(\boldsymbol{r}) - \phi_{g'}^{(l)}(\boldsymbol{r})\right]d^3r}{\sum\limits_{g'=1}^{G}\int\nu\Sigma_{f,g'}\phi_{g'}^{(l+\frac{1}{2})}(\boldsymbol{r})d^3r}\,. \tag{Alg7.2}$$

4. Re-normalize the flux using Equation (2.39c).

First, we present the *LA shift*, which is defined as follows:

$$s_{LA,g,g'}(\boldsymbol{r}) \equiv \chi_g(\boldsymbol{r})\Sigma_{a,g'}(\boldsymbol{r})\,. \tag{4.10}$$

Here, Σ_a is the absorption cross section defined by

$$\Sigma_{a,g}(\boldsymbol{r}) \equiv \Sigma_{t,g}(\boldsymbol{r}) - \sum_{g'=1}^{G}\Sigma_{s0,g\to g'}(\boldsymbol{r})\,, \tag{4.11}$$

χ is the fission spectrum, and we adopt the convention that $\chi_g(\boldsymbol{r}) \equiv 0$ in all nonfissile regions. Physically, the LA shift removes all losses due to absorption from the diffusion operator. The LA shift is motivated by the fact that the shifted diffusion operator is critical in problems with no leakage and no nonfissile regions. For realistic diffusion problems with nonzero leakage and/or nonfissile regions, the shifted diffusion operator is subcritical, which guarantees a nonnegative solution as long as the fission source is nonnegative (see Section 4.1.1 for more details on this). We note that the LA shift is equivalent to the LE shift in grey (1-group) problems since the local infinite-medium eigenvalue in a 1-group problem is given by:

$$\lambda_\infty(\boldsymbol{r}) = \frac{\Sigma_a(\boldsymbol{r})}{\nu\Sigma_f(\boldsymbol{r})} . \tag{4.12}$$

Although an LA-shifted diffusion operator is subcritical in realistic problems, the shifted fission source may still have negative components, and this is problematic for CMFD. In a manner analogous to the development of the LEPS and PLEPS shifts, the LAPS and PLAPS shifts were developed to eliminate negative sources from the LA shift and to improve its efficiency. These shifts are defined as follows:

$$s^{(l)}_{LAPS,g,g'}(\boldsymbol{r}) \equiv \min\left\{\left(\lambda^{(l)} - \epsilon\right)\chi_g(\boldsymbol{r})\nu\Sigma_{f,g'}(\boldsymbol{r}), \chi_g(\boldsymbol{r})\Sigma_{a,g'}(\boldsymbol{r})\right\} , \tag{4.13}$$

$$s^{(l)}_{PLAPS,g,g'}(\boldsymbol{r}) \equiv \max\left\{\lambda^{(l)}_{PARCS}\chi_g(\boldsymbol{r})\nu\Sigma_{f,g'}(\boldsymbol{r}),\ s^{(l)}_{LAPS,g,g'}(\boldsymbol{r})\right\} . \tag{4.14}$$

The LAPS shift is designed to eliminate the possibility of negative sources, while the PLAPS shift is designed to improve the efficiency of LAPS in the later iterations of PI by leveraging the PARCS shift.

In general, we have seen that LE-based shifts have a smaller spectral radius than LA-based shifts. (That is, LE-based shifts typically converge in fewer PIs.) However, LA based shifts have two primary advantages over LE-based shifts: (1) they are less likely to produce critical or supercritical shifted diffusion operators, and (2) they do not require the computation of infinite-medium eigenvalues in each spatial region.

4.2 Fourier Analysis

In this section, we perform a multigroup Fourier analysis of the following shifted PI scheme for 1-D multigroup diffusion eigenvalue problems:

$$\left[-\frac{d}{dx}D_g\frac{d}{dx} + \Sigma_{t,g}\right]\phi_g^{(l+\frac{1}{2})}(x) - \sum_{g'=1}^{G}\left[\Sigma_{s0,g'\to g} + s_{g,g'}\right]\phi_{g'}^{(l+\frac{1}{2})}(x)$$

$$= \lambda^{(l)}\chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'}\phi_{g'}^{(l)}(x) - \sum_{g'=1}^{G} s_{g,g'}\phi_{g'}^{(l)}(x)\,, \tag{4.15a}$$

$$\lambda^{(l+1)} = \frac{\lambda^{(l)} \sum\limits_{g'=1}^{G} \int \nu\Sigma_{f,g'}\phi_{g'}^{(l)}(x)d^3r + \sum\limits_{g=1}^{G}\sum\limits_{g'=1}^{G} \int s_{g,g'}\left[\phi_{g'}^{(l+\frac{1}{2})}(x) - \phi_{g'}^{(l)}(x)\right]d^3r}{\sum\limits_{g'=1}^{G} \int \nu\Sigma_{f,g'}\phi_{g'}^{(l+\frac{1}{2})}(x)d^3r}\,, \tag{4.15b}$$

$$\phi_g^{(l+1)} = \left[\frac{1}{X}\sum_{g=1}^{G}\int_0^X \phi_g^{(l+\frac{1}{2})}(x)dx\right]^{-1}\phi_g^{(l+\frac{1}{2})}(x)\,. \tag{4.15c}$$

In this Fourier analysis, the domain is a homogeneous medium of length X with periodic boundary conditions, and $s_{g,g'}$ is a general shift parameter (either an LE-based shift, an LA-based shift, or a standard WS).

We begin with the following ansatz:

$$\phi_g^{(l)}(x) = \varphi_g + \epsilon a_g \omega^l e^{i\xi x}\,, \tag{4.16a}$$

$$\phi_g^{(l+\frac{1}{2})}(x) = \varphi_g + \epsilon b_g \omega^{l+1} e^{i\xi x}\,, \tag{4.16b}$$

$$\lambda^{(l)} = \lambda_0 + \epsilon\omega^l\lambda_1\,. \tag{4.16c}$$

Here, φ_g is the normalized and spatially constant eigenvector of Equations (4.15), and λ_0 is its corresponding eigenvalue. ϵ is a small parameter tending to zero, ξ is the spatial frequency of the error mode, and ω is the decay factor that we wish to calculate. To satisfy the periodicity of the domain, we impose the following constraint on ξ:

$$\xi \in \left\{ \frac{2\pi z}{X} \,\middle|\, z \in \mathbb{Z}, z > 0 \right\}\,. \tag{4.17}$$

Substituting the Fourier ansatz into Equations 4.15b and 4.15c yields, after dropping the $O(\epsilon^2)$ terms and performing some straightforward algebra,

$$a_g = b_g\,, \tag{4.18}$$

$$\phi_g^{(l+1)} = \phi_g^{(l+\frac{1}{2})}\,, \tag{4.19}$$

$$\lambda_1 = 0\,. \tag{4.20}$$

These results stem primarily from the fact that the $O(\epsilon)$ terms integrate to 0 in Equations 4.15b and 4.15c due to the periodicity of the error modes. Substituting these three results into Equations (4.16) yields an updated Fourier ansatz:

$$\phi_g^{(l)}(x) = \varphi_g + \epsilon a_g \omega^l e^{i\xi x} \,, \tag{4.21a}$$

$$\phi_g^{(l+\frac{1}{2})}(x) = \varphi_g + \epsilon a_g \omega^{l+1} e^{i\xi x} \,, \tag{4.21b}$$

$$\lambda^{(l)} = \lambda_0 \,. \tag{4.21c}$$

Next, when we substitute Equations (4.21) into Equation (4.15a), all the terms of $O(1)$ cancel out, and, after dividing both sides by $\epsilon e^{i\xi x}$, we are left with the following equation:

$$\omega \left[\xi^2 D_g + \Sigma_{t,g}\right] a_g - \omega \sum_{g'=1}^{G} \left[\Sigma_{s0,g'\to g} + s_{g,g'}\right] a_{g'} = \lambda_0 \chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'} a_{g'} - \sum_{g'=1}^{G} s_{g,g'} a_{g'} \,. \tag{4.22}$$

Equation (4.22) is a $G \times G$ linear eigenvalue problem that can be solved to obtain ω for given ξ, cross sections, and D_g. The spectral radius is the maximum magnitude of ω over all possible values of ξ. (Usually, the spectral radius is the $|\omega|$ corresponding to $\xi_{\text{min}} = \frac{2\pi}{X}$.)

For the LE shift, we substitute

$$s_{g,g'} = \lambda_0 \chi_g \nu\Sigma_{f,g'} \tag{4.23}$$

into Equation (4.22). ω would then be 0 since the right side of Equation (4.22) becomes identically zero. This proves our earlier claim that the LE shift has a spectral radius of zero for homogeneous problems with periodic boundary conditions. (We note that performing a shifted PI on a realistic problem with $\lambda' = \lambda$ would require solving a singular system, and this is difficult to do numerically.)

The spectral radius for the LA shift and standard iteration-independent WS is shown in Figure 4.1. To generate the results in this figure, we set $X = 756$ cm (600 pin pitches) and we considered the smallest possible ξ. There, we see that, although the LA shift does not yield a predicted spectral radius of zero like the LE shift, it is still very small ($\sim$0.01). Moreover, we see that, for a standard WS, the predicted spectral radius is much larger than the spectral radius of the LA shift for this toy problem unless λ' is *very* close to $\lambda = \lambda_0$.

The applicability of the results of the Fourier analysis in this section is limited for two reasons. First, the convergence rate of WS-accelerated PI in realistic problems depends strongly on heterogeneities and boundary conditions, and neither of these aspects are considered in the Fourier analysis. Second, although a WS reduces the number of PIs needed to converge, the linear systems that must be solved in each PI (Equations (2.39a), (Alg6.1), and (Alg7.1)) are more difficult to solve. There is a tradeoff that exists – the spectral radius of WS-accelerated PI is inversely related to the condition number of the shifted diffusion operator. That is, a WS reduces the number of PIs needed to converge, but the number of linear solver iterations needed per PI may increase significantly. This tradeoff will be examined in detail in the next section. Despite these limitations, the Fourier

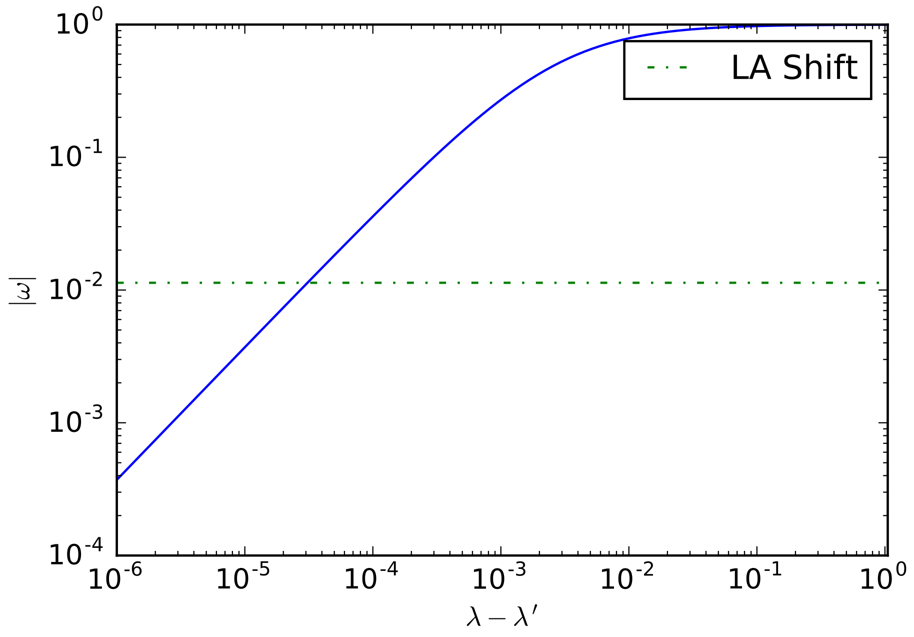


Figure 4.1: Spectral radii predicted by Fourier analysis for a PI scheme with a space-independent Wielandt shift, λ'. Values on the x-axis represent the difference between the true eigenvalue λ and the shift λ'. The predicted spectral radius for the LA shift, which does not depend on λ', is also shown using a dotted line. Here, $X = 756$ cm, and the diffusion coefficients and cross sections were obtained by volume averaging the leftmost five pins of the WB-1D-1 problem described in Appendix B.

analysis in this section still provides some insight in the general convergence properties of WS, and it helped motivate our development of the LE-based SDWSs.

4.3 Impact of Wielandt Shift on Conditioning

In this section, we study the tradeoff between reducing the number of PIs needed for convergence and the conditioning of the shifted diffusion system in each PI. The content in this section is generally known by the reactor physics community, but it has not been studied in detail.

When the "aggressiveness" of an applied WS (space-dependent or standard) is increased, we see all of the following effects:

1. the diffusion system becomes less subcritical or closer to critical,
2. the diffusion system is closer to a singular system,
3. the diffusion system has a higher condition number,
4. the dominance ratio decreases (i.e., it is farther from 1),

5. the spectral radius of the WS-accelerated PI scheme decreases (i.e., it takes fewer PIs to converge).

These five effects are very closely related, and one could argue that they are actually different manifestations of the same phenomenon. Items #2 and #3 are related to the difficulty of solving the linear systems in step 2 of Algorithms 4, 6, or 7, while items #4 and #5 are related to the convergence rate of WS-accelerated PI. Figure 4.2 shows a plot of the condition number of the shifted diffusion operator as a function of $\lambda - \lambda'$ for the standard WS. (The condition number is defined in Section 2.2.4.) The condition numbers of the shifted diffusion matrix for the LE and LA SDWSs are also shown for reference. We see from this plot that the condition number grows rapidly as $\lambda' \to \lambda$, and we see that the shifted diffusion matrices for the LE and LA SDWS techniques has a condition number that is 1000 times larger than the original condition number of the matrix.

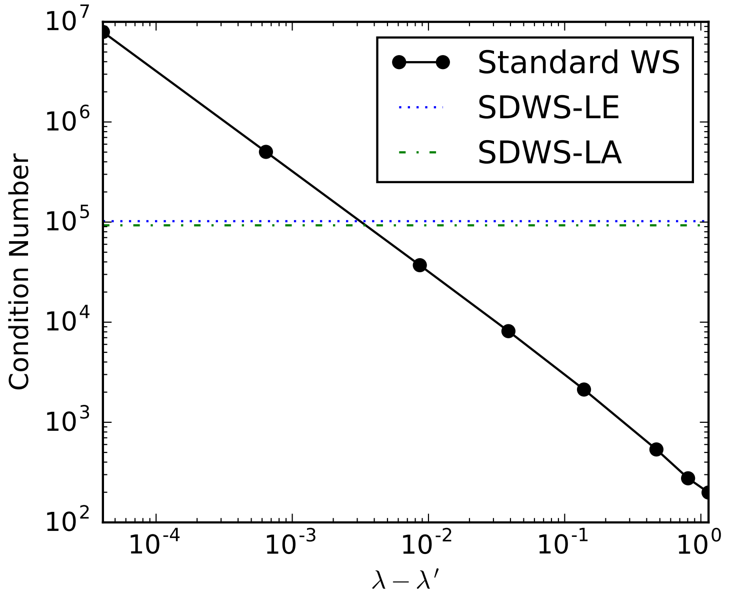


Figure 4.2: Plot of the conditioner number as a function of the standard (space-independent) WS parameter λ' for WB-1D-1 with $\Delta x = 6.3$ cm.

Figure 4.3 shows the condition number of all the iteration-dependent WSs described in this paper (PARCS, LEPS, PLEPS, LAPS, PLAPS) over the course of a WS-accelerated PI scheme for the WB-1D-1 problem described in Appendix B with $\Delta x = 6.3$ cm. From this plot, we can directly see the tradeoff between more aggressive and less aggressive WS techniques. The techniques that converge in fewer PIs also result in shifted diffusion matrices with higher condition numbers. Moreover, we see the "benefit" of using an SDWS over a traditional WS such as the PARCS shift. The SDWSs have higher condition numbers than the PARCS shift at the beginning of the problem because they use the material properties of the problem to generate an aggressive shift. On the other hand, the PARCS shift, whose effectiveness in reducing the dominance ratio is entirely dependent on the convergence of $\lambda^{(l)}$, does not produce an aggressive shift until the end of the problem.

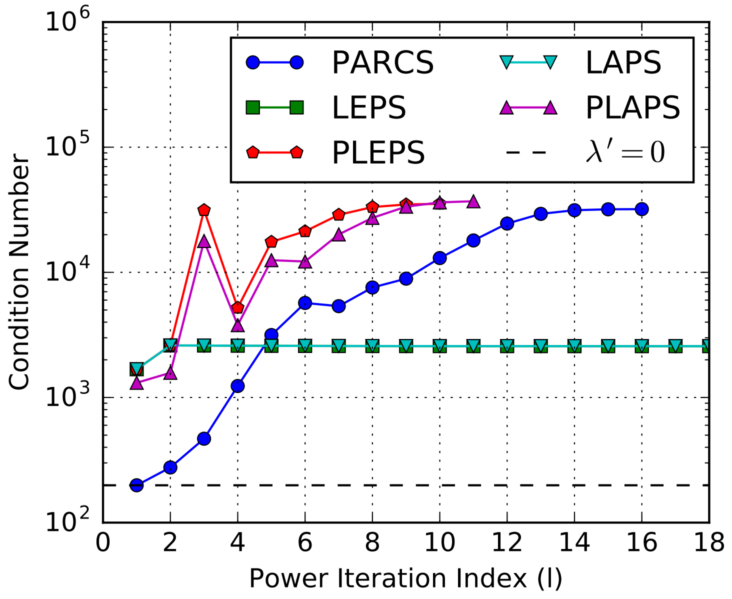


Figure 4.3: Plot highlighting the growth of the conditioner number for several iteration-dependent WSs over the course of the PI scheme (i.e., Algorithms 4, 6, and 7) for the WB-1D-1 problem described in Appendix B with $\Delta x = 6.3$ cm. Here, the x-axis is the power iteration index l. The curves terminate upon convergence to a tolerance of 10^{-6}.

Finally, in Figure 4.4, the condition numbers in Figure 4.2 are translated into linear solver iterations for 6 different linear solver methods. Figure 4.4 provides a more concrete sense of the cost of solving the ill-conditioned systems in WS-accelerated PI. From this plot, we see that all of the linear solvers, except for those preconditioned by ILU, degrade as $\lambda - \lambda' \to 0$. Unfortunately, as we we noted in Chapter 2, the ILU preconditioner does not scale well to larger problems on parallel architectures. Both block Jacobi preconditioned Krylov methods suffer significant degradation as $\lambda - \lambda' \to 0$, but BiCGSTAB appears to perform better than GMRES for this particular problem. (The log-scale makes this difficult to see, but the iterations required block Jacobi preconditioned BiCGSTAB method also increases by a factor of ~4 from the rightmost point to the leftmost point.) "Red-black" block Jacobi (RBBJ) degrades rapidly and immediately as $\lambda - \lambda' \to 0$, while a multigrid method using RBBJ as a smoother remains efficient for a wider range of $\lambda - \lambda'$ values. The asymptotic slopes of the multigrid and RBBJ data appears to be the same as $\lambda - \lambda' \to 0$, indicating that the performance of multigrid in the limit as $\lambda - \lambda' \to 0$ may be limited by the performance of its smoother. We note that the primary benefit of using a multigrid method – the insensitivity of its spectral radius to the problem size – is not captured by this plot.

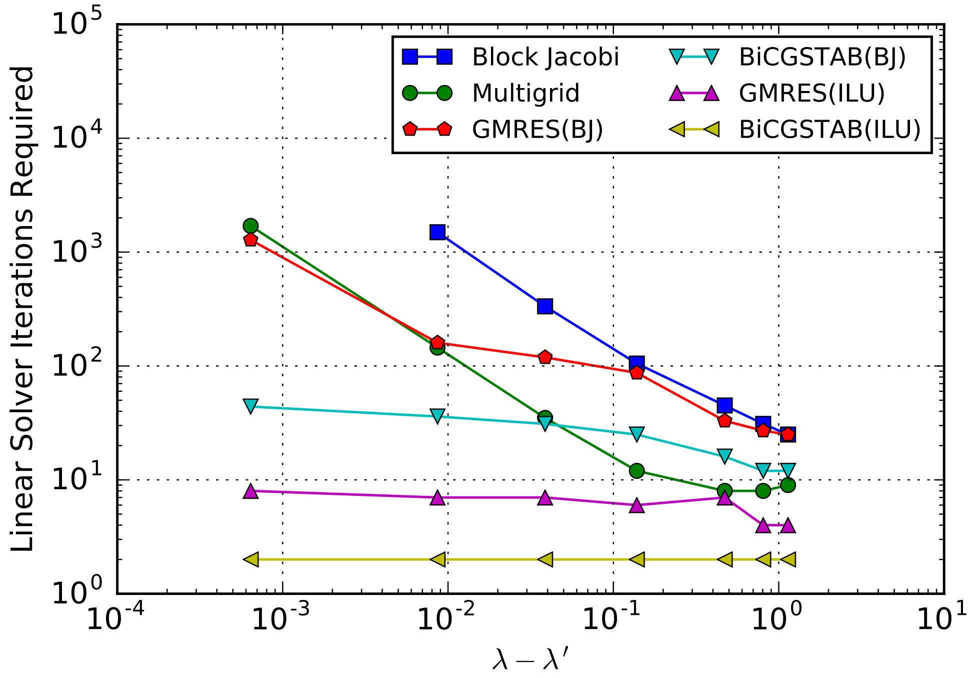


Figure 4.4: Plot of the number of linear solver iterations required for step 2 of Algorithm 4 at $l = 1$ versus the standard WS parameter for the WB-1D-1 problem with $\Delta x = 6.3$ cm. Here, "Block Jacobi" refers to "red-black" block Jacobi (RBBJ). "(BJ)" refers to the use a block Jacobi preconditioner (blocks of size G) and "(ILU)" refers to the use of an ILU preconditioner. The multigrid method uses block Jacobi as its smoother.

In practice, λ' is chosen reasonably far from the true λ in fixed (iteration-independent) WS schemes, since the true λ is not known at the beginning of the problem. (For example, $\lambda' = 2/3$ in MPACT's default CMFD solver.) Moreover, for the PARCS, PLEPS, and PLAPS shifts, c_0 in Equation (2.40) limits how much the diffusion system can be shifted. The original purpose of c_0 was to ensure $\lambda' < \lambda$, but one can also adjust this parameter to control the trade off between the "aggressiveness" of the shift and the condition number of the shifted diffusion operator.

The results in this section (1) underscore the importance of choosing the appropriate linear solver and preconditioner/smoother when solving shifted diffusion systems, and (2) highlight the practical limits on the aggressiveness of WSs. If an aggressive WS is used, the linear solver must be able to handle the high condition numbers of the problem matrix. For large problems, this is problematic, as the ill-conditioning caused by the WS is magnified by the size of the matrix. If an effective linear solver is not available for solving such a system, one should consider reducing the aggressiveness of the WS (e.g., by increasing c_0 for the PARCS shift).

4.4 1-D Diffusion Results

In this section, the 1-D code described in Section 3.1 is used to generate results for the three 1-D reactor models in Appendix B: the Joo-Lee problem, WB-1D-1, and WB-1D-2. From the previous section, we found that the ILU-preconditioned BiCGSTAB method worked best for these 1-D problems in serial, and this will be the linear solver used in all of the results in this section. We compare the performance of the six SDWS techniques described in this chapter to the PARCS shift, a fixed WS of 2/3, and unshifted PI. The convergence criteria and initial guesses are described in Section 3.1.

Table 4.1 provides the runtimes and iterations required for each of the different WS options in the Joo-Lee problem. A plot of the decay of the eigenvector error in the Joo-Lee problem over the course of the simulation is shown in Figure 4.5. Tables 4.2 and 4.3 provide runtime and iteration totals for the WB-1D-1 problem with $\Delta x = 6.3$ cm (5 pin cells) and $\Delta x = 1.26$ cm (1 pin cell), respectively. Tables 4.4 and 4.5 provide the runtime and iteration totals for the WB-1D-2 problems on the same two spatial grids. The corresponding plots of the eigenvector convergence are shown in Figure 4.6.

Table 4.1: Comparison of WS methods for the 1D Joo-Lee problem.

Shift	Power Iterations	Total LSIs	Runtime [s]
$\lambda' = 0$	402	402	7.0
$\lambda' = 2/3$	106	106	2.1
PARCS	17	17	0.4
LE	48	48	1.3
LEPS	70	70	2.2
PLEPS	16	16	0.4

We see in Table 4.1 that only one ILU-preconditioned BiCGSTAB linear solver iteration (LSI) is needed for any of the linear solves. (In Table 4.1, the number of PIs is equal to the number of LSIs for all of the cases.) This is because the Joo-Lee problem, which has no energy-dependence, is a relatively small problem. Results for the LA-based shifts are not shown for this problem because LA-based and LE-based shifts are the same in 1-group problems. (This fact was noted in Section 4.1.4.)

Although the relative performance of all of the methods varies from problem to problem, we see that two of the SDWS methods – PLEPS and PLAPS – had smaller PI, LSI, and runtime totals than the PARCS shift for all of the Joo-Lee and WB-1D problems. (In the Joo-Lee problem, the PLEPS and PLAPS shifts are the same shift since it is a one-group problem.) For the WB-1D-1 and WB-1D-2 problems (i.e., the multigroup problems), the speedups provided by PLEPS/PLAPS over

Table 4.2: Comparison of WS methods for the WB-1D-1 problem, $\Delta x = 6.3$ cm. Here, DNC stands for did not converge.

Shift	PIs	Total LSIs	Runtime [s]
$\lambda' = 0$	199	266	13.4
$\lambda' = 2/3$	94	164	7.4
PARCS	16	39	1.4
LE	–	–	DNC
LEPS	26	51	2.4
PLEPS	10	25	0.9
LA	–	–	DNC
LAPS	44	84	3.9
PLAPS	11	29	1.0

Table 4.3: Comparison of WS methods for the WB-1D-1 problem, $\Delta x = 1.26$ cm.

Shift	PIs	Total LSIs	Runtime [s]
$\lambda' = 0$	227	347	217.2
$\lambda' = 2/3$	111	211	116.1
PARCS	15	48	17.7
LE	68	139	73.5
LEPS	68	139	74.8
PLEPS	14	46	16.5
LA	75	150	78.9
LAPS	97	189	107.7
PLAPS	14	47	16.5

Table 4.4: Comparison of WS methods for the WB-1D-2 problem, $\Delta x = 6.3$ cm.

Shift	PIs	Total LSIs	Runtime [s]
$\lambda' = 0$	216	239	50.5
$\lambda' = 2/3$	70	130	20.1
PARCS	13	26	4.2
LE	7	19	2.5
LEPS	11	23	3.6
PLEPS	7	18	2.5
LA	8	21	2.7
LAPS	52	96	15.4
PLAPS	12	24	4.0

Table 4.5: Comparison of WS methods for the WB-1D-2 problem, $\Delta x = 1.26$ cm.

Shift	PIs	Total LSIs	Runtime [s]
$\lambda' = 0$	224	319	400.8
$\lambda' = 2/3$	74	150	150.8
PARCS	13	34	31.2
LE	13	36	29.1
LEPS	15	39	34.7
PLEPS	9	29	23.1
LA	19	48	41.2
LAPS	61	125	133.5
PLAPS	12	35	29.7

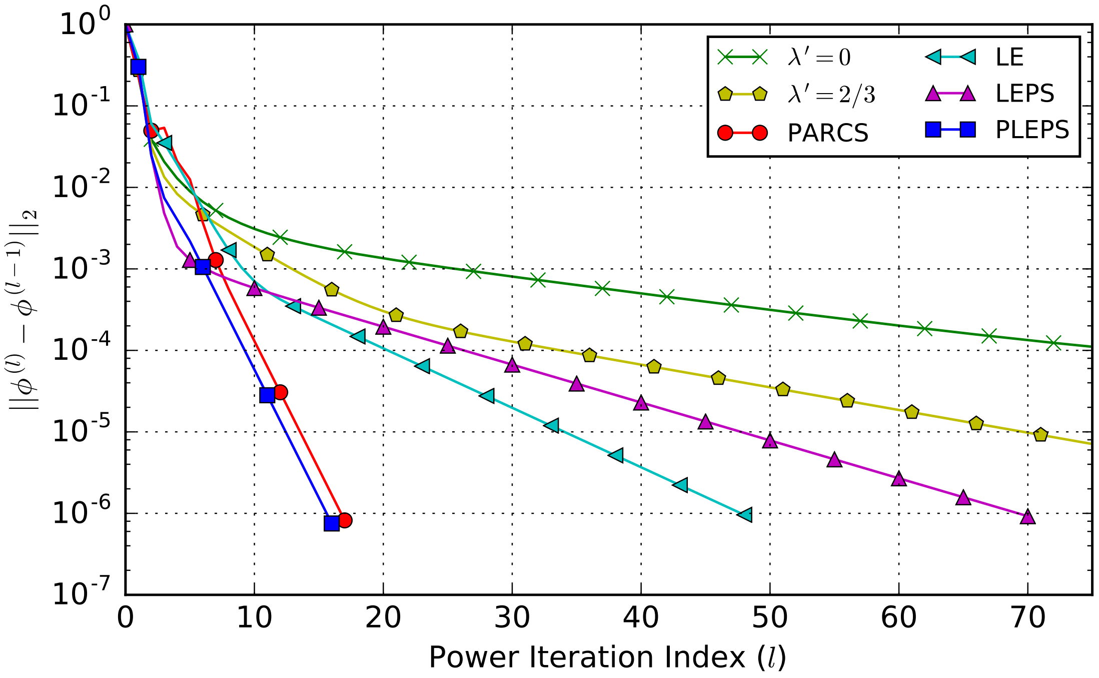


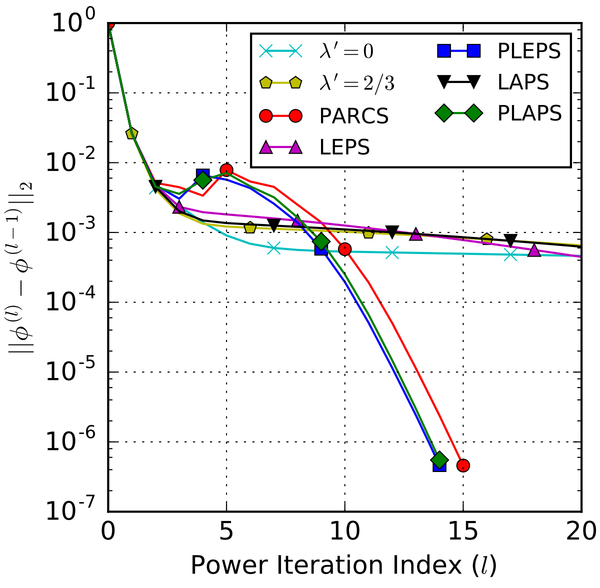
Figure 4.5: Convergence of eigenvector in the 1D 1-group Joo-Lee problem. A summary of the iterations and runtime required to converge is available in Table 4.1.

the PARCS shift ranged between ∼5% and ∼40%. No speedup is seen for the PLEPS shift over the PARCS shift in the 1-group Joo-Lee problem despite a reduction in the number of PIs required from 17 to 16. This is likely because of the difficulty of measuring runtimes accurately for such a small problem.

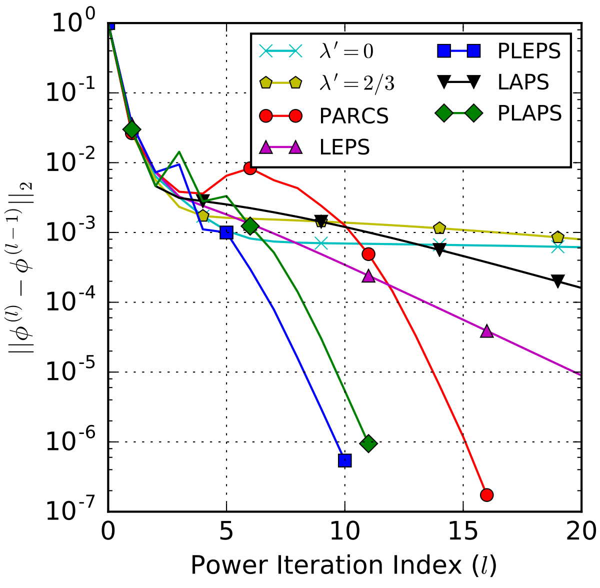
For the problems considered in this section, it appears that the difference between the spectral radii of the PLEPS and PLAPS methods is more significant than the overhead cost of computing infinite-medium eigenvalues in the PLEPS method. The PLEPS shift consistently performed better than the PLAPS shift in all of the results. However, this advantage should disappear for sufficiently large G since the cost of computing infinite-medium eigenvalues scales as G^3.

When they converge, the LE and LA shifts provide improved convergence rates over the LEPS and LAPS shifts. However, they can still converge slowly (e.g., the results in Table 4.3), and they can fail to converge (e.g., Table 4.2). Moreover, even when they do converge, the LE and LA shifts generally do not perform as well as the PLEPS and PLAPS shifts.

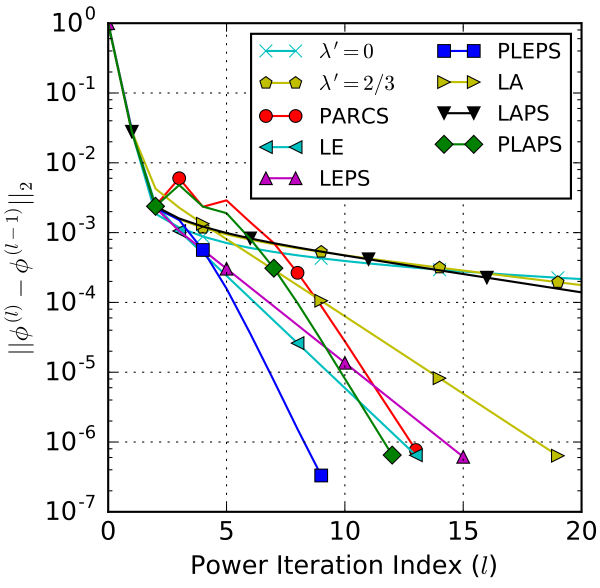
Finally, we note that the convergence patterns seen in Figure 4.6 reflect our design of the PLEPS/PLAPS shifts. At the beginning of the iteration scheme, when the current estimate of the eigenvalue may not be accurate or well-converged, the convergence of the PLEPS/PLAPS shifts primarily follows that of the LEPS/LAPS shifts. (That is, their slopes in Figure 4.6 are similar for small l.) However, after several iterations, the current estimate of the eigenvalue is reasonably



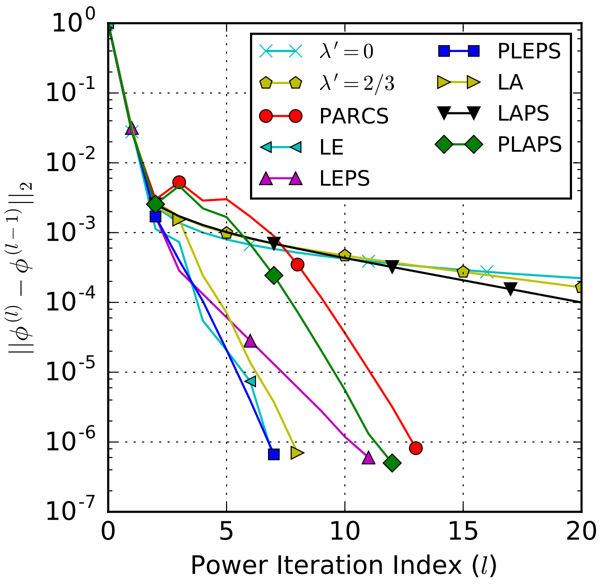
(a) WB-1D-1 problem, $\Delta x = 6.3$ cm.



(b) WB-1D-1 problem, $\Delta x = 1.26$ cm.



(c) WB-1D-2 problem, $\Delta x = 6.3$ cm.



(d) WB-1D-2 problem, $\Delta x = 1.26$ cm.

Figure 4.6: Convergence of the eigenvector for different WS methods in the WB-1D problems, described in Appendix B. A summary of the iterations and runtime required to converge is available in Tables 4.2-4.5.

converged, and the asymptotic convergence rates of the PLEPS and PLAPS shifts are similar to that of the PARCS shift. More analysis of these results can be found in [107]. *In short, the PLAPS and PLEPS shifts improve upon the PARCS shift by providing better shifts in the initial iterations.*

4.5 MPACT Results (3-D CMFD)

Based on the results of the previous section, we implemented the PARCS, PLEPS, and PLAPS shifts in the CMFD solver of the MPACT code for testing on full-sized 3-D reactor problems. In this section, we compare the performance of these three shifts and MPACT's default shift (an iteration-independent shift of $\lambda' = 2/3$) on two problems: problem 5a-ARO and problem 7. Descriptions of these problems are provided in Appendix C, while descriptions of the convergence criteria in MPACT are provided in Section 3.2. MPACT's 47-group library is used for the results in this section (version 3.1, dated November 11th, 2014 [4, 79]). In each PI, block Jacobi preconditioned GMRES from PETSc is used to solve the shifted linear system. The linear solver exits when the default convergence criterion for PETSc Krylov solvers (described in [7]) is met or when it has performed 100 iterations. The results in this section are generated using the version of MPACT corresponding to SHA-1 number `03ec0e117`, which is more recent than that used in [107]. All problems are run on 4234 cores using the Titan computing system at the Oak Ridge Leadership Computing Facility (OLCF). (In [107], we used the Eos computing system at the OLCF, on which MPACT generally runs about twice as fast.)

Results from our implementation are provided in Tables 4.6 and 4.7. The results in these two tables show that the PLEPS and PLAPS shifts can provide a slight improvement over the PARCS shift. However, these improvements are small compared to the improvements seen in the previous sections for 1-D diffusion problems. This is due to several reasons. First, in the MPACT problems, the initial guess of eigenvalue ($\lambda^{(0)} = 1$) is not far from the true solution. The true eigenvalue in problem 5a-ARO is $\lambda = 0.988010$, while the "true eigenvalue" in problem 7 is $\lambda = 1$ because it is a critical boron search problem. Second, unlike the diffusion problems of the previous section, the problems solved in this section are CMFD problems in the context of a transport sweeper. These problems are solved many times (once between each transport sweep), and their solutions are only needed to accelerate the transport solver. After a few transport sweeps and CMFD solves, each CMFD problem is not significantly different from the previous problem (the change in the CMFD correction factors converges to 0 as the problem converges). Thus, most of the time, the solution from the previous CMFD solve serves as a good initial guess for the present CMFD solve. We noted that the primary benefit of using SDWSs is that they provide a good "start" for the shifted PI scheme. In this problem, however, we already have a decent initial guess for the CMFD solves, and the benefit of using SDWSs over the space-independent and empirical PARCS shift is significantly reduced.

Table 4.6: Comparison of WS methods for problem 5a-ARO in MPACT.

Shift	Transport Sweeps	PIs	Total LSIs	CMFD Runtime [s]	Total Runtime [s]
$\lambda' = 2/3$	12	100	10000	490	950
PARCS	12	59	5708	293	765
PLEPS	12	55	5500	282	753
PLAPS	12	56	5600	286	757

Table 4.7: Comparison of WS methods for problem 7 in MPACT.

Shift	Transport Sweeps	PIs	Total LSIs	CMFD Runtime [s]	Total Runtime [s]
$\lambda' = 2/3$	12	101	10100	494	1128
PARCS	13	62	6039	312	1002
PLEPS	13	59	5900	302	991
PLAPS	13	59	5900	301	990

Moreover, in problem 7, the presence of thermohydraulic feedback also distorts the results. We see that the default MPACT shift of $\lambda' = 2/3$ actually converges in one transport sweep fewer than the three options, and the total runtime difference between the default shift and the three iteration-dependent shifts is not as great as in problem 5a-ARO. This is primarily because of the poor convergence properties of the outer feedback solver described in Chapter 3. This poor convergence (and instability) is further elucidated in Section 8.4. There, the MSED method is used to solve the CMFD systems for depletion problems with feedback, but the MSED method has to be tweaked to loosen the convergence of CMFD.

The results in this section show that there are limits to how much the CMFD solver can be improved by optimizing the WS. Table 4.8 further illuminates this limitation. There, the 100-iteration limit on GMRES is raised to 1000 iterations, and we see the full impact of the tradeoff described in Section 4.3. In Table 4.8, we see that, as the aggressiveness of the shifts increases, the total number of GMRES iterations required increases despite the reduction in the number of PIs. For this particular linear solver (block Jacobi preconditioned GMRES with a restart length of 100), much of the benefit of using more aggressive WSs can only be attained by capping the number of GMRES iterations per PI. However, if more aggressive shifts are used (e.g., by reducing c_0 in Equation (2.40)), the solution produced after 100 GMRES iterations may have negative values; these negative values cannot be handled by CMFD's nonlinear update of the transport system.

Table 4.8: Comparison of WS methods for problem 5a-ARO in MPACT, with 1000 GMRES iterations permitted per PI (restart length of 100). In Table 4.6, only 100 GMRES iterations (with no restarts) are permitted per PI.

Shift	Transport Sweeps	PIs	Total LSIs	CMFD Runtime [s]	Total Runtime [s]
$\lambda' = 2/3$	12	100	14441	691	1159
PARCS	12	54	20474	1002	1477
PLEPS	12	50	20921	1023	1492
PLAPS	12	52	20616	1006	1476

Although the results in Table 4.6 and 4.7 show that MPACT's default WS can be improved by using any of the iteration-dependent shifts, this improvement is not sufficient. As an accelerator for the transport solver with significantly fewer unknowns than the transport system (several orders of magnitude fewer), the CMFD system should only require an insignificant fraction of the overall runtime. However, Tables 4.6 and 4.7 indicate that the CMFD runtime still constitutes $\sim 30\%$ of the overall runtime in the best-case scenario. In the remainder of this thesis, we develop the MSED method in order to further reduce this computational cost.

CHAPTER 5

An Overview of MSED

This chapter serves as a high-level summary of the Multilevel-in-Space-and-Energy Diffusion (MSED) method, which is the focus of the remainder of this thesis. Algorithm 8 and Figures 5.1 and 5.2 provide an overview of the MSED method. MSED has two primary components: a grey diffusion eigenvalue problem used to accelerate the convergence of the fission source and eigenvalue (step 2 of Algorithm 8), and a multigrid-in-space linear solver for fixed source diffusion eigenvalue problems (steps 2b and 4 of Algorithm 8).

Figure 5.2 is a visualization of the steps in Algorithm 8; the axes of this plot describe the fidelity of the spatial and energy discretizations of the equations used in MSED. In step 1, we collapse the original multigroup diffusion (or CMFD) eigenvalue problem to a grey diffusion eigenvalue problem (Equation (6.2)). In step 2, we perform PIs using either the iteration-dependent PARCS WS defined by Equation (2.40) or the iteration- and space-dependent PLEPS shift defined by Equation (4.8). In each PI, the fixed-source diffusion problem is solved using the multigrid linear solver. Each of these linear solves consists of performing multiple V-cycles in which one traverses between spatial grids of varying sizes (each spatial grid is represented by a green circle in Figure 5.2). The repetition of the orange arrows in Figure 5.2 reflects the fact that multiple PIs are needed to converge the eigenvalue and fission source.

Step 3 brings us back to the multigroup system; here, we update estimates of the eigenvalue, eigenvector, and fission source using the results of the grey system. Then, in step 4, we perform a "partial" PI to converge the energy-dependent aspects of the solution (e.g., the scattering source, the differences in the spatial shape of each group's scalar flux, and the leakage spectrum). It is "partial" in the sense that only two multigrid V-cycles are used to "solve" the fixed-source problem in this PI. The choice to use 2 V-cycles stems from the Fourier analysis in Section 7.2, which tells us that using additional V-cycles would have a negligible impact on the spectral radius of the overall MSED scheme. (This is also verified experimentally in [10].) The multigrid linear solver used to solve the multigroup system in step 4 has the same structure as the one used to solve the grey system in step 2; the only difference is that the interpolation and restriction operators are account for the group structure in the multigroup case. Details regarding the multigrid solver are provided in Chapter 7.

Algorithm 8: An MSED iteration.

Input: $\phi_{j,g}^{(l)}$, $\lambda^{(l)}$

Result: $\phi_{j,g}^{(l+1)}$, $\lambda^{(l+1)}$

1. Using the multigroup scalar flux $\phi_{j,g}^{(l)}$, compute the grey quantities given by Equations (6.3) to construct the grey diffusion eigenvalue problem (Equation (6.2)).

2. Solve the grey diffusion eigenvalue problem by performing many Wielandt shifted PIs. Each shifted PI consists of the following steps:

   (a) Compute the WS using Equation (2.40) (or Equation (4.8)).

   (b) Apply the WS and solve the resulting fixed-source grey diffusion problem using the multigrid (in space) linear solver. Each iteration in the multigrid linear solver is a V-cycle described by Algorithm 2 using interpolation and restriction operators described in Equations (7.1).

   (c) Update the eigenvalue using Equation (2.39b) or Equation (Alg6.2), and re-normalize the grey scalar flux using Equation (2.39c).

3. Set the estimate of the eigenvalue in the multigroup system equal to the eigenvalue obtained in the grey system. Update the estimate of the multigroup scalar flux using Equation (6.4).

4. Perform a "partial" PI on the multigroup system (Equation (6.5)). That is, perform the steps described by Algorithm 3, but only perform two multigrid V-cycles when "solving" the fixed-source diffusion system. The V-cycles on the multigroup system have the same structure as the V-cycles on the grey system and are also described by Algorithm 2. However, the interpolation and restriction operators (described by Equations (7.1)) account for the group structure.

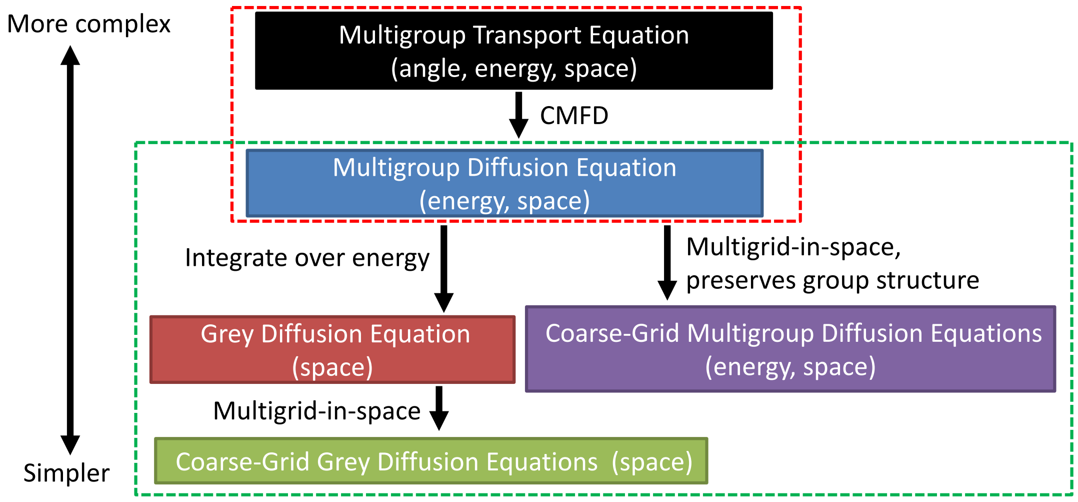

More complex
Simpler
Multigroup Transport Equation (angle, energy, space)
CMFD
Multigroup Diffusion Equation (energy, space)
Integrate over energy
Multigrid-in-space, preserves group structure
Grey Diffusion Equation (space)
Coarse-Grid Multigroup Diffusion Equations (energy, space)
Multigrid-in-space
Coarse-Grid Grey Diffusion Equations (space)

Figure 5.1: A reproduction of Figure 1.1, which visualizes the multilevel hierarchy of equations in CMFD and MSED. The upper (red) dashed box encloses the equations used in CMFD, while the lower (green) dashed box encloses the equations used in MSED.

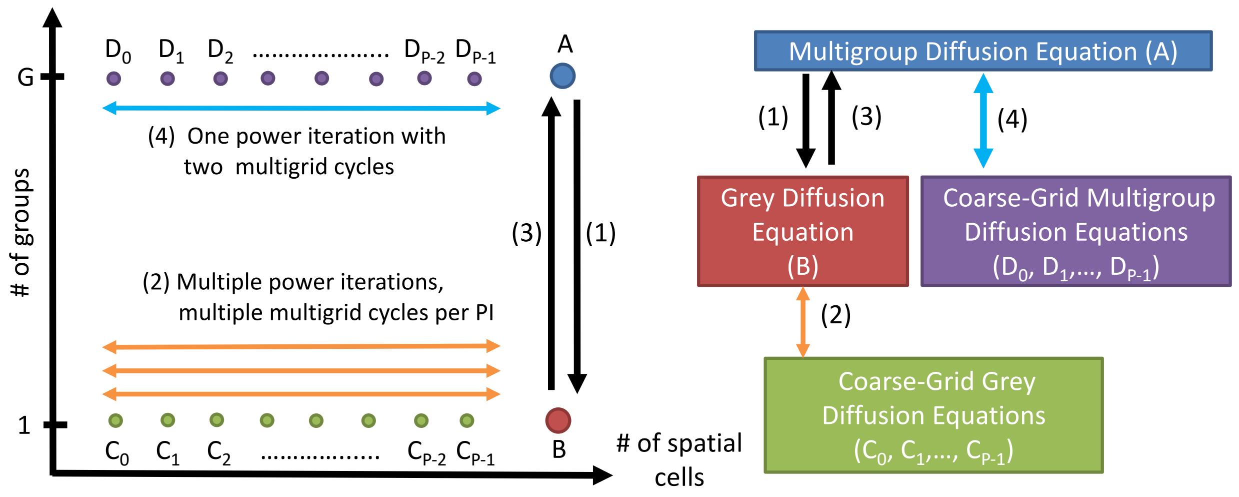

# of groups
G
1
D_0 D_1 D_2 .... D_{P-2} D_{P-1}
A
(4) One power iteration with two multigrid cycles
(3)
(1)
(2) Multiple power iterations, multiple multigrid cycles per PI
C_0 C_1 C_2 .... C_{P-2} C_{P-1}
B
# of spatial cells
Multigroup Diffusion Equation (A)
(1)
(3)
(4)
Grey Diffusion Equation (B)
Coarse-Grid Multigroup Diffusion Equations (D_0, D_1,..., D_{P-1})
(2)
Coarse-Grid Grey Diffusion Equations (C_0, C_1,..., C_{P-1})

Figure 5.2: A visualization of an MSED iteration in phase space. Details on each of the four visualized steps are provided in Algorithm 8.

The final chapters of this thesis are organized as follows. Chapters 6 and 7 discuss the two components of MSED separately. Chapter 6 develops the theory for the grey diffusion eigenvalue problem and the multilevel-in-energy method used to iterate between the multigroup and grey energy grids. It also provides a Fourier analysis of the multilevel-in-energy iteration scheme (denoted as MED), and selected results are presented to isolate and quantify the acceleration provided by the grey diffusion eigenvalue problem in the MSED Method. Chapter 7 serves the equivalent purpose for the multigrid-in-space component of MSED.

Chapter 8 provides further results for a larger selection of problems and an additional discussion regarding the overall performance of MSED. The purpose of Chapter 8 is to show that MSED is a robust method that can efficiently solve a wide array of problems. Chapter 9 discusses potential avenues for future work, and Chapter 10 provides a final summary of the important conclusions of the thesis.

CHAPTER 6

Multilevel in Energy

In this chapter, we develop the theory for the grey diffusion eigenvalue problem used in MSED (red box in Figure 1.1). This is the first of the two major components of MSED; the other component (multigrid-in-space) is discussed in Chapter 7. Fourier analyses and results from both the 1-D diffusion code and MPACT are presented to quantify the speedup provided by the use of the grey diffusion eigenvalue problem. For the seven representative problems considered in MPACT, we observe a speedup of $\sim$4-20x over the default CMFD solver when the grey diffusion eigenvalue problem is used. For the *3-D, full-core* problems specifically, the speedup range is $\sim$4-6x. This chapter primarily describes novel work performed over the course of the PhD; the background of grey diffusion acceleration (i.e., its history) is covered in Chapter 2. To our knowledge, our specific formulation of the grey diffusion system, the manner in which it is used to accelerate the convergence of the multigroup system, and the analysis we have performed are novel, even though grey diffusion acceleration itself is not a new idea. This chapter also discusses how MSED can be extended to arbitrary multilevel-in-energy structures (i.e., coarse energy grids with more than one group and methods with more than two energy grids).

6.1 Theory

6.1.1 Grey Diffusion

We begin with the 1-D CMFD eigenvalue problem:

$$
\begin{aligned}
-\frac{1}{\Delta x_j}\left[D_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}-\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}}-D_{j-\frac{1}{2},g}\frac{\phi_{j,g}-\phi_{j-1,g}}{\Delta x_{j-\frac{1}{2}}}\right]& \\
+\frac{1}{\Delta x_j}\left[\hat{D}_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}+\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}}+\hat{D}_{j-\frac{1}{2},g}\frac{\phi_{j,g}+\phi_{j-1,g}}{\Delta x_{j+\frac{1}{2}}}\right]& \\
+\Sigma_{t,j,g}\phi_{j,g}-\sum_{g'=1}^{G}\Sigma_{s0,j,g'\to g}\phi_{j,g'}=\lambda\chi_{j,g}\sum_{g'=1}^{G}\nu\Sigma_{f,j,g'}\phi_{j,g'}\,.& \quad (6.1)
\end{aligned}
$$

The full 3-D CMFD eigenvalue problem is provided in Equation (2.22), and descriptions of the variables are provided in Chapter 2. We present the theory in 1-D to simplify the notation and make the concepts easier to understand; all of the content in this chapter can be extended to 2-D and 3-D problems in a straightforward manner.

By adjusting $\Sigma_{t,j,g}$ and $D_{j\pm\frac{1}{2},g}$ to "absorb" the $\hat{D}$ terms, we can define an equivalent formulation of Equation (2.22) without the $\hat{D}$. Thus, the manner in which MSED is applied to CMFD problems and diffusion problems are the same [11]. For generality, we present the theory for CMFD problems (i.e., with the $\hat{D}$ explicitly included); the corresponding theory for multigroup diffusion problems can be obtained by simply setting $\hat{D}$ to zero. The theory in this chapter should also be extensible to multigroup diffusion eigenvalue problems with other forms of spatial discretization (i.e., not 2nd-order cell-centered finite-difference), but the grey diffusion coefficient in Equation (6.3d) would have to be redefined in a suitable manner.

The grey diffusion eigenvalue problem (red box in Figure 5.1) is obtained by a straightforward summation of Equation (6.2) over all groups g:

$$
\begin{aligned}
-\left[\frac{\left\langle D_{j+\frac{1}{2},j+1}\right\rangle}{\Delta x_j\Delta x_{j+\frac{1}{2}}}\right]\Phi_{j+1}-\left[\frac{\left\langle D_{j-\frac{1}{2},j-1}\right\rangle}{\Delta x_j\Delta x_{j-\frac{1}{2}}}\right]\Phi_{j-1}+& \\
\left[\frac{\left\langle D_{j+\frac{1}{2},j}\right\rangle}{\Delta x_j\Delta x_{j+\frac{1}{2}}}+\frac{\left\langle D_{j-\frac{1}{2},j}\right\rangle}{\Delta x_j\Delta x_{j-\frac{1}{2}}}+\left\langle\Sigma_{a,j}\right\rangle\right]\Phi_j=\lambda\left\langle\nu\Sigma_{f,j}\right\rangle\Phi_j\,.& \quad (6.2)
\end{aligned}
$$

In this equation, the grey scalar flux and bracketed quantities are given by

$$\Phi_j \equiv \sum_{g=1}^{G} \phi_{j,g} \,, \tag{6.3a}$$

$$\langle \Sigma_{a,j} \rangle \equiv \frac{1}{\Phi_j} \sum_{g=1}^{G} \Sigma_{a,j,g} \phi_{j,g} \,, \tag{6.3b}$$

$$\langle \nu\Sigma_{f,j} \rangle \equiv \frac{1}{\Phi_j} \sum_{g=1}^{G} \nu\Sigma_{f,j,g} \phi_{j,g} \,, \tag{6.3c}$$

$$\langle D_{j_1,j_2} \rangle \equiv \begin{cases} \frac{1}{\Phi_{j_2}} \sum\limits_{g=1}^{G} \left(D_{j_1,g} - \hat{D}_{j_1,g} \right) \phi_{j_2,g} \,, & \text{if } j_2 = j \text{ (if } j_2 \text{ is the current cell)} \,, \\ \frac{1}{\Phi_{j_2}} \sum\limits_{g=1}^{G} \left(D_{j_1,g} + \hat{D}_{j_1,g} \right) \phi_{j_2,g} \,, & \text{if } j_2 \neq j \text{ (if } j_2 \text{ is a neighbor)} \,. \end{cases} \tag{6.3d}$$

Here, Σ_a is the absorption cross section defined by Equation (4.11). The boundary conditions of the grey problem can be obtained by performing a similar flux-weighted collapse of the coefficients of $\phi_{j,g}$. After Equation (6.2) is solved, its solution can be used to update the multigroup scalar flux as follows:

$$\phi_{j,g}^{(\text{new})} = \frac{\Phi_j^{(\text{new})}}{\Phi_j^{(\text{old})}} \phi_{j,g}^{(\text{old})} = \frac{\Phi_j^{(\text{new})}}{\sum\limits_{g'=1}^{G} \phi_{j,g'}^{(\text{old})}} \phi_{j,g}^{(\text{old})} \,. \tag{6.4}$$

The careful collapse of the diffusion coefficients via Equation (6.3d) ensures that the grey diffusion coefficients are positive and well-defined when $(\Phi_{j+1} - \Phi_j) \to 0$. From Equation (6.3d), we see that it is possible for the grey diffusion coefficients to be negative if the CMFD correction factors are sufficiently large. However, we have not found this to be a concern in practice. In all of the problems considered in this thesis, the grey diffusion coefficients generated via Equation (6.3d) are positive. We note that Equations (6.3) describe a collapse-in-energy that is consistent with the collapse used in Cornejo's multilevel LONDA method [66]. A comparison of the Cornejo's method and MSED was provided in Section 1.4.

Algorithm 9 describes the manner in which the grey diffusion equation is used to accelerate the convergence of the full multigroup diffusion or CMFD eigenvalue problem. It is an incomplete version of Algorithm 8; effectively, Algorithm 9 is Algorithm 8 with no details regarding how to solve the fixed-source diffusion problems in steps 2b and 4. In the remainder of this thesis, we will refer to Algorithm 9 as *Multilevel-in-Energy Diffusion (MED)* whenever multigrid-in-space is not used as the linear solver in steps 2 and 4. In other words, MED is a generalization of MSED, and the MSED method is a specific type of MED method. For this thesis, the MED method refers to the use of a Krylov solver such as GMRES or BiCGSTAB in lieu of the multigrid solver.

Algorithm 9: Steps for an MED iteration. (This algorithm is the same as Algorithm 8, except that there are no details here describing how the fixed-source diffusion problems in steps 2b and 4 are solved.)

```
Input: $\phi_{j,g}^{(l)}$, $\lambda^{(l)}$
Result: $\phi_{j,g}^{(l+1)}$, $\lambda^{(l+1)}$
1. Using the multigroup scalar flux $\phi_{j,g}^{(l)}$, compute the grey quantities given by Equations (6.3)
   to construct the grey diffusion eigenvalue problem (Equation (6.2)).
2. Solve the grey diffusion eigenvalue problem by performing many Wielandt shifted PIs.
   Each shifted PI consists of the following steps:
    (a) Compute the WS using Equation (2.40) (or Equation (4.8)).
    (b) Apply the WS and solve the resulting fixed-source grey diffusion problem.
    (c) Update the eigenvalue using Equation (2.39b) or Equation (Alg6.2), and re-normalize
        the grey scalar flux using Equation (2.39c).
3. Set the estimate of the eigenvalue in the multigroup system equal to the eigenvalue obtained
   in the grey system. Update the estimate of the multigroup scalar flux using Equation (6.4).
4. Perform an unshifted PI (Algorithm 3) on the multigroup system:
    (a) Solve a fixed-source multigroup diffusion or CMFD problem (Equation (6.5)) in
        which the fission source is generated using the multigroup scalar flux and eigenvalue
        from step 3.
    (b) Update the eigenvalue using Equation (2.37b).
    (c) Re-normalize the flux using Equation (2.37c).
```

The grey diffusion eigenvalue problem allows us to efficiently converge the eigenvalue and the fission source without the use of a Wielandt shift on the multigroup problem. Both the fission source and eigenvalue are grey quantities regardless of the energy-dependence of the problem, and their impact on the solution is mostly captured by the grey diffusion problem. The energy-independence of the fission source results from the assumption that the fission spectrum χ_g is independent of the incident neutron energy. If this assumption did not hold, χ_g would depend on the incoming neutron energy group g', and we would not be able to factor it out of the summation on the right side of Equation (6.1).

In either Algorithm 4 or Algorithm 9, the application of a Wielandt shift is the driving mechanism for converging the "eigen"-aspects of the problem; it is needed for PI to converge the eigenvalue and eigenvector in an optimal number of iterations. However, by applying the WS to the grey diffusion problem instead of the multigroup problem, the ill-conditioning described in Section 4.3 is mitigated significantly. In both MED/MSED and standard Wielandt-shifted PI, the application of WS results in higher condition numbers. The difference is that the grey problem is G times smaller than the multigroup problem, and a small ill-conditioned linear system is much easier to solve than a large ill-conditioned linear system. Applying the WS to the grey diffusion system instead of the multigroup system allows us to bypass the limitations of WS described in Section 4.3. Work is still needed on the multigroup diffusion system to converge the energy-dependence of the problem in either MED or MSED, but the quantity of work required is significantly reduced by leveraging the much smaller grey diffusion system and removing ill-conditioning caused by WS from the multigroup system. This phenomenon is further elucidated in the numerical results of this chapter, and it is consistent with the findings of recent work by [73].

6.1.2 Treatment of Multigroup System in MSED

In this subsection, we discuss how one can take advantage of the structure of the scattering operator when solving multigroup fixed source problems in MED or MSED. For a fixed source $q_{j,g}$, the fixed-source multigroup diffusion problem is given by:

$$-\frac{1}{\Delta x_j}\left[D_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}-\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}} - D_{j-\frac{1}{2},g}\frac{\phi_{j,g}-\phi_{j-1,g}}{\Delta x_{j-\frac{1}{2}}}\right] + \Sigma_{t,j,g}\phi_{j,g} - \sum_{g'=1}^{G}\Sigma_{s0,j,g'\to g}\phi_{j,g'} = q_{j,g}\,. \tag{6.5}$$

The content in this subsection applies to fixed-source CMFD problems as well, but we have omitted the CMFD correction factors for simplicity. In MPACT, this system is typically solved by explicitly constructing the linear system in the form of a matrix and then applying an iterative linear solver (e.g., GMRES) to all of the groups at once. We refer to this approach in this thesis as the "standard" approach or the "full multigroup" approach.

In most reactor problems, however, there is no up-scattering to the first $\sim G/2$ groups, and we can take advantage of this scattering structure in MED and MSED. To aid this discussion, we define a few variables. First, we define g_{th} to be the first group for which there is an incoming upscattering source in any spatial cell j. That is, g_{th} is defined such that for any $g < g_{th}$, we have $\Sigma_{s0,j,g' \to g} = 0$ for all j and all $g' > g$. We define the *thermal* groups as groups g_{th}, …, G, and we define G_{th} to be the number of thermal groups:

$$G_{th} \equiv G - g_{th} + 1 \,. \tag{6.6}$$

All of the remaining groups (1, …, $g_{th} - 1$) are considered to be *non-thermal*. For these non-thermal groups, the diffusion operator (without Wielandt shift) only operates on groups 1 through g. In other words, when $g < g_{th}$, the scattering source into group g only depends on the scalar flux from groups $1, \ldots, g$. Thus, in MED and MSED, the scalar flux for groups 1 through $g_{th} - 1$ can be obtained by solving one-group problems in a Gauss-Seidel-like manner. If we solve for the scalar flux one group at a time starting from $g = 1$, the incoming scattering source (excluding the self-scattering component $g \to g$) is always known for these non-thermal groups.

Once the scalar flux is obtained for all of the non-thermal groups, there are two choices for obtaining the scalar fluxes of the remaining thermal groups. The first approach is to solve for the thermal groups simultaneously so that no upscatter iterations are needed. We refer to this approach as the "*reduced multigroup*" approach in this thesis. This approach is similar to the "standard" approach, except that we only form a multi-group linear system for the G_{th} thermal groups. The alternative approach is to continue solving the groups one at a time as we do with the non-thermal groups; this requires us to iterate through the thermal groups a few times to converge the upscatter source. We refer to this approach as the "*one-group sweep*" approach. Algorithms 10 and 11 describe these two approaches in more detail. We note that one-group sweeps are used in Cornejo's multilevel-in-energy method as smoothing steps on each energy grid [66].

In their current forms, Algorithms 10 and 11 cannot be used when a WS is applied. This is because the shift term $\lambda' \chi_g \nu\Sigma_{f,g'}$ introduces effective upscattering from the thermal groups to the fast groups. (In reactor problems, $\nu\Sigma_{f,g'}$ is generally nonzero for $g > g_{th}$ while χ_g is generally nonzero for $g < g_{th}$.) Thus, the efficient approaches described in these two algorithms are a unique advantage provided by the grey diffusion eigenvalue problem, which allows us to avoid the use of a WS on the multigroup problem. One can still use Algorithm 11 to solve the fixed-source problems in PI schemes with WS, but one would have to iterate over all of the groups rather than just the thermal groups.

Compared to the standard approach, the reduced multigroup approach is almost always more efficient (in terms of both memory and computational cost) because the cost of solving the diffusion linear system scales *super*linearly with the number of groups G. This superlinearity is is due to the fact that (1) the number of nonzeros in the diffusion operator matrix scales as G^2, and (2) the

Algorithm 10: "Reduced" approach for solving the fixed-source multigroup diffusion problem described in Equation (6.5). This algorithm does not describe the linear solver used in steps 1 and 2.

```
    Result: $\phi_{j,g}$
1. For $g = 1, \dots, g_{th} - 1$:
     Solve the following one-group problem to obtain $\phi_{j,g}$ for all $j$:
```

$$-\frac{1}{\Delta x_j}\left[D_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}-\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}} - D_{j-\frac{1}{2},g}\frac{\phi_{j,g}-\phi_{j-1,g}}{\Delta x_{j-\frac{1}{2}}}\right] + \left[\Sigma_{t,j,g} - \Sigma_{s0,g\to g}\right]\phi_{j,g} = q_{j,g} + \sum_{g'=1}^{g-1}\Sigma_{s0,j,g'\to g}\phi_{j,g'}\,. \quad \text{(Alg10.1)}$$

```
     For each $g$, the right side of this equation is known.
2. Solve the remaining multigroup diffusion problem for groups $g_{th}, \dots, G$:
```

$$-\frac{1}{\Delta x_j}\left[D_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}-\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}} - D_{j-\frac{1}{2},g}\frac{\phi_{j,g}-\phi_{j-1,g}}{\Delta x_{j-\frac{1}{2}}}\right] + \Sigma_{t,j,g}\phi_{j,g} - \sum_{g'=g_{th}}^{G}\Sigma_{s0,j,g'\to g}\phi_{j,g'} = q_{j,g} + \sum_{g'=1}^{g_{th}-1}\Sigma_{s0,j,g'\to g}\phi_{j,g'}\,. \quad \text{(Alg10.2)}$$

```
   Here, the right side of the equation is known.
```

Algorithm 11: "One-group sweep" approach for solving the fixed-source multigroup diffusion problem described in Equation (6.5). This algorithm does not describe the linear solver used in steps 1 and 2.

```
    Input: Initial guess $\phi^{(0)}_{j,g}$ for all thermal groups $g = g_{th}, \ldots, G$
    Result: $\phi_{j,g}$
1.  for $g = 1, \ldots, g_{th} - 1$:
        Solve the following one-group problem to obtain $\phi_{j,g}$ for all $j$:
```

$$-\frac{1}{\Delta x_j}\left[D_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}-\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}} - D_{j-\frac{1}{2},g}\frac{\phi_{j,g}-\phi_{j-1,g}}{\Delta x_{j-\frac{1}{2}}}\right] + \left[\Sigma_{t,j,g} - \Sigma_{s0,g\to g}\right]\phi_{j,g} = q_{j,g} + \sum_{g'=1}^{g-1}\Sigma_{s0,j,g'\to g}\phi_{j,g'}\,. \quad \text{(Alg11.1)}$$

```
        For each $g$, the right side of this equation is known.

2.  for $m = 1, \ldots, m_{max}$:
      for $g = g_{th}, \ldots, G$:
        Solve the following one-group problem to obtain $\phi^{(m)}_{j,g}$:
```

$$-\frac{1}{\Delta x_j}\left[D_{j+\frac{1}{2},g}\frac{\phi^{(m)}_{j+1,g}-\phi^{(m)}_{j,g}}{\Delta x_{j+\frac{1}{2}}} - D_{j-\frac{1}{2},g}\frac{\phi^{(m)}_{j,g}-\phi^{(m)}_{j-1,g}}{\Delta x_{j-\frac{1}{2}}}\right] + \left[\Sigma_{t,j,g} - \Sigma_{s0,j,g\to g}\right]\phi^{(m)}_{j,g}$$
$$= q_{j,g} + \sum_{g'=g+1}^{G}\Sigma_{s0,j,g'\to g}\phi^{(m-1)}_{j,g'} + \sum_{g'=g_{th}}^{g-1}\Sigma_{s0,j,g'\to g}\phi^{(m)}_{j,g'} + \sum_{g'=1}^{g_{th}-1}\Sigma_{s0,j,g'\to g}\phi_{j,g'}\,. \quad \text{(Alg11.2)}$$

```
        Here, the right side of the equation is known at every iteration.

    $m_{max}$ is the number of "upscatter sweeps" (i.e., the number of times we sweep over the
    thermal groups). In this thesis, $m_{max} = 2$ unless otherwise specified.
```

condition number worsens with problem/matrix size. In the standard approach, we solve all G equations simultaneously. In the reduced approach, we solve $g_{th} - 1$ one-group problems and then a G_{th}-group problem. Using the reduced multigroup approach reduces both the memory burden and the number of computations required.

When G is not too large (e.g., $G \approx 50$), the reduced multigroup approach is more efficient than the one-group sweep approach. However, as we have noted, the number of matrix nonzeros scales as G^2, and it is advantageous to avoid forming multigroup matrices when G is large (e.g., $G \approx 200$). For these large values of G, the one-group sweep approach is more efficient than the reduced multigroup approach in both memory and computational costs. A numerical comparison of all three approaches on large problems in MPACT is provided in Section 6.4.1.

One drawback of using the reduced multigroup or one-group sweep approaches is its impact on the communication requirements of the code. The amount of information communicated between processors should be similar for all three approaches, but the frequency and size of the messages sent will differ. Because of how the multigroup problem is divided up in each approach, the full multigroup approach will result in less frequent but larger messages, while the one-group sweep approach results in more frequent but smaller messages and the reduced multigroup approach falls somewhere in-between. Moreover, most linear solvers require synchronization between all processors at every linear solver iteration. This means that the number of "wait-all" points in the code (points where each processor has to wait until every other processor is finished with its current task) is proportional to the number of linear systems we break the multigroup problem into. In Section 8.2, we see that this can result in longer runtimes even if the number of computations has been reduced.

6.1.3 Extension to Arbitrary Coarse-Group Energy Structures

In this subsection, we consider the extension of Algorithm 9 to coarse energy grids with more than one group and to iteration schemes with more than two energy grids (e.g., schemes with intermediate grid(s) between multigroup and grey).

First, we show that it is possible to use the approach in Equations (6.3) to generate a diffusion equation on a coarsened energy grid with more than one group. We let the index c denote the coarse group defined by the union of fine groups $g_{c,1}, g_{c,2}, \dots, g_{c,G_c}$. C is the total number of coarse groups, and G_c is the number of fine groups in coarse group c. (In the grey diffusion equation, $C = 1$, and $G_1 = G$.) Then, the diffusion equation for coarse group c can be obtained by summing Equation (6.2) over fine groups $g_{c,1}, g_{c,2}, \dots, g_{c,G_c}$:

$$-\left[\frac{\left\langle D_{j+\frac{1}{2},j+1,c}\right\rangle}{\Delta x_j \Delta x_{j+\frac{1}{2}}}\right]\Phi_{j+1,c}-\left[\frac{\left\langle D_{j-\frac{1}{2},j-1,c}\right\rangle}{\Delta x_j \Delta x_{j-\frac{1}{2}}}\right]\Phi_{j-1,c}+\left[\frac{\left\langle D_{j+\frac{1}{2},j,c}\right\rangle}{\Delta x_j \Delta x_{j+\frac{1}{2}}}+\frac{\left\langle D_{j-\frac{1}{2},j,c}\right\rangle}{\Delta x_j \Delta x_{j-\frac{1}{2}}}+\left\langle \Sigma_{t,j,c}\right\rangle\right]\Phi_{j,c}$$
$$-\sum_{c'=1}^{C}\left\langle \Sigma_{s0,j,c'\to c}\right\rangle \Phi_{j,c'} = \lambda \left\langle \chi_{j,c}\right\rangle \sum_{c'=1}^{C}\left\langle \nu\Sigma_{f,j,c'}\right\rangle \Phi'_{j,c}\,. \quad (6.7)$$

Here, the grey scalar flux and bracketed quantities are given by:

$$\Phi_{j,c} \equiv \sum_{g\in c}\phi_{j,g}\,, \quad (6.8a)$$

$$\left\langle \Sigma_{t,j,c}\right\rangle \equiv \frac{1}{\Phi_{j,c}}\sum_{g\in c}\Sigma_{t,j,g}\phi_{j,g}\,, \quad (6.8b)$$

$$\left\langle \Sigma_{s0,j,c'\to c}\right\rangle \equiv \frac{1}{\Phi_{j,c'}}\sum_{g'\in c'}\sum_{g\in c}\Sigma_{s0,j,g'\to g}\phi_{j,g'}\,, \quad (6.8c)$$

$$\left\langle \nu\Sigma_{f,j,c}\right\rangle \equiv \frac{1}{\Phi_{j,c}}\sum_{g\in c}\nu\Sigma_{f,j,g}\phi_{j,g}\,, \quad (6.8d)$$

$$\left\langle \chi_{j,c}\right\rangle \equiv \sum_{g\in c}\chi_{j,g}\,, \quad (6.8e)$$

$$\left\langle D_{j_1,j_2,c}\right\rangle \equiv \begin{cases} \frac{1}{\Phi_{j_2,c}}\sum\limits_{g\in c}\left(D_{j_1,g}-\hat{D}_{j_1,g}\right)\phi_{j_2,g}\,, & \text{if } j_2 = j \text{ (if } j_2 \text{ is the current cell)}\,, \\ \frac{1}{\Phi_{j_2,c}}\sum\limits_{g\in c}\left(D_{j_1,g}+\hat{D}_{j_1,g}\right)\phi_{j_2,g}\,, & \text{if } j_2 \neq j \text{ (if } j_2 \text{ is a neighbor)}\,. \end{cases} \quad (6.8f)$$

In Equations (6.8), the summation over $g \in c$ denotes the summation over all fine groups g in coarse group c. As long as each coarse group c is defined as a union of existing fine groups, the procedure described by Equations (6.8) is algebraically valid.[1] After Equation (6.2) is solved, we update the multigroup scalar flux as follows for each fine group g in coarse group c:

$$\phi_{j,g}^{(\text{new})} = \frac{\Phi_{j,c}^{(\text{new})}}{\Phi_{j,c}^{(\text{old})}}\phi_{j,g}^{(\text{old})} = \frac{\Phi_{j}^{(\text{new})}}{\sum\limits_{g'\in c}\phi_{j,g'}^{(\text{old})}}\phi_{j,g}^{(\text{old})}\,. \quad (6.9)$$

In reactor physics, a standard approach has been to collapse from a multigroup problem to two-group problem (e.g., [3, 62, 63]). Typically, the two coarse groups are defined contiguously such that the fast group consists of all groups without upscattering and the thermal group consists of all

[1]We note that the equations in this subsection do not require that coarse groups be defined contiguously. In [110], the generation and use of discontiguous energy groups is explored in detail, and it may be possible to use some of those techniques to define advantageous coarse energy groups for Equation (6.7).

the remaining groups. This structure results in a relatively ideal nonzero structure for Equation (6.7) since $\langle \Sigma_{s0,j,2\to1} \rangle = 0$, $\langle \chi_{j,2} \rangle = 0$, and, unless there is fast fission, $\langle \nu\Sigma_{f,j,1} \rangle \ll \langle \nu\Sigma_{f,j,2} \rangle$. The use of a two-group coarse energy grid instead of a one-group grid can reduce the number of MED/MSED iterations required to converge, but a two-group diffusion problem requires more computational effort to solve than a grey diffusion problem. The Fourier analysis and numerical results in this thesis indicate that, in some problems, the use of two groups instead of one can result in a reduced runtime. However, this improvement (if any) is relatively minor in most cases, and it is sufficient to use a grey diffusion problem.

Finally, we note that defining an MED/MSED method with more than two energy grids is fairly straightforward with the mechanics we have defined in this chapter. This is because it is possible to rewrite Equation (6.7) so that it has the same coefficient structure as Equation (6.1). Once Equation (6.7) has been rewritten in the form of Equation (6.1), Equations (6.8) can be used to define a collapse to an energy grid with even fewer groups. Although multiple energy grids are not considered in this thesis, recent works by both Cornejo and Collins has indicated that having additional energy grids can provide tangible improvements in the overall runtime by reducing the computational effort required on the finest energy grid [66, 111].

6.1.4 Alternate Grey Diffusion Equation

Finally, we note that we have also explored the use of an alternate grey diffusion equation, derived from multiplying Equation (6.2) by a space-and-energy dependent function $f_{j,g}$ before coarsening in energy [10]. A Fourier analysis of this alternate grey diffusion equation indicates that choosing $f_{j,g}$ to be an estimate of the multigroup adjoint eigenfunction leads to an iteration scheme with an improved spectral radius and a reduced likelihood of instability. For reactor problems, however, this improvement is very small and likely does not justify the cost of computing $f_{j,g}$. We have not encountered any problems for which the use of this alternate grey diffusion has yielded tangible improvements. Nonetheless, one should consider this alternate grey diffusion equation if one encounters a problem in which the use of the standard grey diffusion equation results in instability. More details on this alternate grey diffusion equation are provided in Appendix D.

6.2 Fourier Analysis

6.2.1 Algorithm

In this subsection, we describe the MED algorithm for a 1-D multigroup diffusion eigenvalue problem in a homogeneous domain of size X, divided into J spatial cells of width Δx, with periodic boundary conditions. At iteration l, we have the estimates $\phi_g^{(l)}$ and $\lambda^{(l)}$ from the previous iteration. The process by which Algorithm 9 is used to obtain $\phi_g^{(l+1)}$ and $\lambda^{(l+1)}$ is described by the following equations:

$$\Phi_{j,c}^{(l)} \equiv \sum_{g\in c} \phi_{j,g}^{(l)}\,, \tag{6.10a}$$

$$\left\langle \Sigma_{t,j,c} \right\rangle^{(l)} \equiv \frac{1}{\Phi_{j,c}^{(l)}} \sum_{g\in c} \Sigma_{t,g}\phi_{j,g}^{(l)}\,, \tag{6.10b}$$

$$\left\langle \Sigma_{s0,j,c'\to c} \right\rangle^{(l)} \equiv \frac{1}{\Phi_{j,c'}^{(l)}} \sum_{g'\in c'} \sum_{g\in c} \Sigma_{s0,g'\to g}\phi_{j,g'}^{(l)}\,, \tag{6.10c}$$

$$\left\langle \nu\Sigma_{f,j,c} \right\rangle^{(l)} \equiv \frac{1}{\Phi_{j,c}^{(l)}} \sum_{g\in c} \nu\Sigma_{f,g}\phi_{j,g}^{(l)}\,, \tag{6.10d}$$

$$\left\langle \chi_c \right\rangle \equiv \sum_{g\in c} \chi_g\,, \tag{6.10e}$$

$$\left\langle D_{j,c} \right\rangle^{(l)} \equiv \frac{1}{\Phi_{j,c}^{(l)}} \sum_{g\in c} D_g\phi_{j,g}^{(l)}\,, \tag{6.10f}$$

$$\begin{aligned} &\frac{-\left\langle D_{j+1,c}\right\rangle^{(l)} \Phi_{j+1,c}^{(l')} + 2\left\langle D_{j,c}\right\rangle^{(l)} \Phi_{j,c}^{(l')} - \left\langle D_{j-1,c}\right\rangle^{(l)} \Phi_{j-1,c}^{(l')}}{\Delta x^2} + \left\langle \Sigma_{t,j,c}\right\rangle^{(l)} \Phi_{j,c}^{(l')} \\ &\qquad - \sum_{c'=1}^{C} \left\langle \Sigma_{s0,j,c'\to c}\right\rangle^{(l)} \Phi_{j,c'}^{(l')} = \lambda^{(l')} \left\langle \chi_c \right\rangle \sum_{c'=1}^{C} \left\langle \nu\Sigma_{f,j,c'}\right\rangle^{(l)} \Phi_{j,c}^{\prime(l')}\,, \end{aligned} \tag{6.11a}$$

$$\frac{1}{J}\sum_{j=1}^{J}\sum_{c=1}^{C} \Phi_{j,c}^{(l')} = 1\,, \tag{6.11b}$$

$$\phi_{j,g}^{(l')} = \frac{\Phi_{j,c}^{(l')}}{\Phi_{j,c}^{(l)}}\,\phi_{j,g}^{(l)} \quad \forall g \in c\,, \tag{6.12}$$

$$\frac{D_g}{\Delta x^2}\left[-\phi_{j+1,g}^{(l'')} + 2\phi_{j,g}^{(l'')} - \phi_{j-1,g}^{(l'')}\right] + \Sigma_{t,g}\phi_{j,g}^{(l'')} - \sum_{g'=1}^{G} \Sigma_{s0,g'\to g}\phi_{g'}^{(l'')} = \lambda^{(l')}\chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'}\phi_{j,g'}^{(l')}\,, \tag{6.13a}$$

$$\lambda^{(l+1)} = \lambda^{(l')}\left[\sum_{j=1}^{J}\sum_{g'=1}^{G} \nu\Sigma_{f,g'}\phi_{g'}^{(l'')}\right]^{-1} \sum_{j=1}^{J}\sum_{g'=1}^{G} \nu\Sigma_{f,g'}\phi_{j,g'}^{(l')}\,, \tag{6.13b}$$

$$\phi_{j,g}^{(l+1)} = \left[\frac{1}{J} \sum_{j=1}^{J} \sum_{g'=1}^{G} \phi_{j,g'}^{(l'')} \right]^{-1} \phi_{j,g}^{(l'')} \,. \tag{6.13c}$$

Here, we have not fixed the structure of the coarse energy grid. The MED method described in Algorithm 9 is obtained by setting $C = 1$, while a multilevel-in-energy method that uses a two-group coarse energy grid is obtained by setting $C = 2$. The $C = 2$ case is more common in reactor physics, and, as noted in Section 6.1.3, the two coarse groups are typically defined such that the "fast" group consists of all groups without upscattering and the "thermal" group consists of all the remaining groups[2]. In the following Fourier analysis, both options are considered.

Moreover, we note that we are not explicitly considering the CMFD correction factors $\hat{D}$ in this analysis. At the beginning of this chapter (near Equation (6.1)), we pointed out that the CMFD problem is identical in form to a diffusion problem with modified diffusion coefficients and total cross sections. Thus, the Fourier analysis here applies to both diffusion and CMFD problems as long as the diffusion operator is not rendered supercritical by the presence of the $\hat{D}$.

6.2.2 Analysis

We now perform a Fourier analysis of the algorithm described by Equations (6.10)-(6.13). We begin with the following Fourier ansatz:

$$\phi_{j,g}^{(l)} = \varphi_g + \epsilon a_g \omega^l e^{i\xi j \Delta x} \,, \tag{6.14a}$$
$$\Phi_{j,c}^{(l)} = \Upsilon_c \left[1 + \epsilon A_c \omega^l e^{i\xi j \Delta x} \right] \,, \tag{6.14b}$$
$$\lambda^{(l)} = \lambda_0 + \epsilon \lambda_1 \omega^l \,, \tag{6.14c}$$
$$\phi_{j,g}^{(l')} = \varphi_g + \epsilon a'_g \omega^l e^{i\xi j \Delta x} \,, \tag{6.14d}$$
$$\Phi_{j,c}^{(l')} = \Upsilon_c \left[1 + \epsilon A'_c \omega^l e^{i\xi j \Delta x} \right] \,, \tag{6.14e}$$
$$\lambda^{(l')} = \lambda_0 + \epsilon \lambda'_1 \omega^l \,, \tag{6.14f}$$
$$\phi_{j,g}^{(l'')} = \varphi_g + \epsilon a''_g \omega^{l+1} e^{i\xi j \Delta x} \,. \tag{6.14g}$$

In this ansatz, i is the imaginary number $\sqrt{-1}$, and j is the spatial index. φ_g and λ_0 are the solutions to the infinite-medium multigroup diffusion eigenvalue problem, and

$$\Upsilon_c \equiv \sum_{g \in c} \varphi_g \,. \tag{6.15}$$

[2]One can also define the two groups such that $\langle \chi_1 \rangle = 1$ and $\langle \chi_2 \rangle = 0$ in all fissile regions.

φ_g satisfies the normalization condition

$$\sum_{g=1}^{G} \varphi_g = 1\,. \tag{6.16}$$

As in Section 4.2, ω is the decay factor, and ξ is the spatial frequency of the error mode. To ensure periodicity and to avoid aliased frequencies, ξ should be chosen as follows:

$$\xi \in \left\{ \frac{2\pi k}{X} \middle| k \in \mathbb{Z}, 0 < k \leq \left\lfloor \frac{J}{2} \right\rfloor \right\}\,. \tag{6.17}$$

Here, $\lfloor \cdot \rfloor$ is the floor function (i.e., the function that rounds down to the nearest integer).

We begin the analysis by substituting the Fourier ansatz into Equations (6.10). First, Equation (6.10a) yields

$$A_c = \frac{1}{\Upsilon_c} \sum_{g\in c} \alpha_g\,. \tag{6.18}$$

Second, we express each of the flux-weighted quantities in Equations (6.10) except $\langle \Sigma_{s0,c'\to c}\rangle^{(l)}$ in the following form:

$$\begin{aligned}
\langle \Sigma_{j,c}\rangle^{(l)} &= \frac{1}{\Phi_{j,c}^{(l)}} \sum_{g\in c} \Sigma_g \phi_{j,g}^{(l)} \\
&= \frac{1}{\Upsilon_c} \left[1 + \epsilon\omega^l e^{i\xi j\Delta x} A_c\right]^{-1} \sum_{g\in c} \Sigma_g \left[\varphi_g + \epsilon a_g \omega^l e^{i\xi j\Delta x}\right] \\
&\approx \frac{1}{\Upsilon_c} \left[1 - \epsilon\omega^l e^{i\xi j\Delta x} A_c\right] \sum_{g\in c} \Sigma_g \left[\varphi_g + \epsilon a_g \omega^l e^{i\xi j\Delta x}\right] \\
&\approx \frac{1}{\Upsilon_c} \sum_{g\in c} \Sigma_g \left[\varphi_g + \epsilon\omega^l e^{i\xi j\Delta x} \left(a_g - A_c\varphi_g\right)\right] \\
&= \overline{\Sigma}_c + \epsilon\omega^l e^{i\xi j\Delta x} \Upsilon_c^{-1} \sum_{g\in c} \Sigma_g \left(a_g - A_c\varphi_g\right) \\
&= \overline{\Sigma}_c + \epsilon\omega^l e^{i\xi j\Delta x} \left(-\overline{\Sigma}_c A_c + \Upsilon_c^{-1} \sum_{g\in c} \Sigma_g a_g \right)\,.
\end{aligned} \tag{6.19}$$

Here, Σ can be replaced with $\nu\Sigma_f$, D, or Σ_t. The symbol $\approx$ marks the places in the derivation where $O(\epsilon^2)$ terms are dropped. For a cross-section or diffusion coefficient Σ, $\overline{\Sigma}_c$ is used to denote a coarse group cross section obtained by flux-weighting with the infinite-medium spectrum. For example,

$$\overline{\Sigma}_{t,c} \equiv \frac{1}{\Upsilon_c} \sum_{g\in c} \Sigma_{t,g}\varphi_g \, . \tag{6.20}$$

Finally, the coarse-group scattering cross-section is given by

$$\begin{aligned}
\left\langle \Sigma_{s0,j,c'\to c} \right\rangle^{(l)} &= \frac{1}{\Phi^{(l)}_{j,c'}} \sum_{g'\in c'} \sum_{g\in c} \Sigma_{s0,g'\to g}\phi^{(l)}_{j,g'} \\
&= \frac{1}{\Upsilon_{c'}} \left[1 + \epsilon\omega^l e^{i\xi j\Delta x} A_{c'}\right]^{-1} \sum_{g'\in c'} \sum_{g\in c} \Sigma_{s0,g'\to g} \left[\varphi_{g'} + \epsilon a_{g'}\omega^l e^{i\xi j\Delta x}\right] \\
&\approx \frac{1}{\Upsilon_{c'}} \left[1 - \epsilon\omega^l e^{i\xi j\Delta x} A_{c'}\right] \sum_{g'\in c'} \sum_{g\in c} \Sigma_{s0,g'\to g} \left[\varphi_{g'} + \epsilon a_{g'}\omega^l e^{i\xi j\Delta x}\right] \\
&\approx \frac{1}{\Upsilon_{c'}} \sum_{g'\in c'} \sum_{g\in c} \Sigma_{s0,g'\to g} \left[\varphi_{g'} + \epsilon\omega^l e^{i\xi j\Delta x} \left(a_{g'} - A_{c'}\varphi_{g'}\right)\right] \\
&\equiv \overline{\Sigma}_{s0,j,c'\to c} + \epsilon\omega^l e^{i\xi j\Delta x} \Upsilon^{-1}_{c'} \sum_{g'\in c'} \sum_{g\in c} \Sigma_{s0,j,g'\to g} \left(a_{g'} - A_{c'}\varphi_{g'}\right) \\
&= \overline{\Sigma}_{s0,j,c'\to c} + \epsilon\omega^l e^{i\xi j\Delta x} \left(-\overline{\Sigma}_{s0,j,c'\to c} A_{c'} + \Upsilon^{-1}_{c'} \sum_{g'\in c'} \sum_{g\in c} \Sigma_{s0,j,g'\to g} a_{g'} \right) \, .
\end{aligned} \tag{6.21}$$

Substituting Equations (6.14b), (6.18), (6.19), and (6.21) into Equation (6.11a) yields

$$\begin{aligned}
&-\frac{1}{\Delta x^2} \left\{ \overline{D}_c + \epsilon\omega^l e^{i\xi(j-1)\Delta x} \left(-\overline{D}_c A_c + \Upsilon^{-1}_c \sum_{g\in c} D_g a_g \right) \right\} \Upsilon_c \left[1 + \epsilon\omega^l A'_c \Upsilon_c e^{i\xi(j-1)\Delta x}\right] \\
&\quad + \frac{2}{\Delta x^2} \left\{ \overline{D}_c + \epsilon\omega^l e^{i\xi j\Delta x} \left(-\overline{D}_c A_c + \Upsilon^{-1}_c \sum_{g\in c} D_g a_g \right) \right\} \Upsilon_c \left[1 + \epsilon\omega^l A'_c \Upsilon_c e^{i\xi j\Delta x}\right] \\
&\quad - \frac{1}{\Delta x^2} \left\{ \overline{D}_c + \epsilon\omega^l e^{i\xi(j+1)\Delta x} \left(-\overline{D}_c A_c + \Upsilon^{-1}_c \sum_{g\in c} D_g a_g \right) \right\} \Upsilon_c \left[1 + \epsilon\omega^l A'_c \Upsilon_c e^{i\xi(j+1)\Delta x}\right] \\
&\quad + \left\{ \overline{\Sigma}_{t,c} + \epsilon\omega^l e^{i\xi j\Delta x} \left(-\overline{\Sigma}_{t,c} A_c + \Upsilon^{-1}_c \sum_{g\in c} \Sigma_{t,g} a_g \right) \right\} \Upsilon_c \left[1 + \epsilon A'_c \omega^l e^{i\xi j\Delta x}\right] \\
&\quad - \sum_{c'=1}^{C} \left\{ \overline{\Sigma}_{s0,j,c'\to c} + \epsilon\omega^l e^{i\xi j\Delta x} \left(-\overline{\Sigma}_{s0,j,c'\to c} A_{c'} \right.\right. \\
&\quad \left.\left. + \Upsilon^{-1}_{c'} \sum_{g'\in c'} \sum_{g\in c} \Sigma_{s0,j,g'\to g} a_{g'} \right) \right\} \Upsilon_{c'} \left[1 + \epsilon A'_{c'} \omega^l e^{i\xi j\Delta x}\right] = \left[\lambda_0 + \epsilon\lambda'_1\omega^l\right] \langle\chi_c\rangle \\
&\times \sum_{c'=1}^{C} \left\{ \overline{\nu\Sigma}_{f,c'} + \epsilon\omega^l e^{i\xi j\Delta x} \left(\overline{\nu\Sigma}_{f,c'} A_{c'} + \Upsilon^{-1}_{c'} \sum_{g\in c'} \nu\Sigma_{f,g} a_g \right) \right\} \Upsilon_{c'} \left[1 + \epsilon A'_{c'} \omega^l e^{i\xi j\Delta x}\right] \, .
\end{aligned} \tag{6.22}$$

To simplify this behemoth, we perform the following steps:

1. Eliminate all terms of $O(1)$. They cancel out from the definitions of φ_g and λ_0.
2. Drop all terms of $O(\epsilon^2)$ or smaller.
3. Divide the equation by ϵw^l.
4. Make the following substitution:

$$\frac{1}{\Delta x^2}\left[-e^{-i\xi\Delta x}+2+e^{i\xi\Delta x}\right]=\frac{4}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right). \tag{6.23}$$

   To obtain the equivalent Fourier analysis for a problem without spatial discretization, replace the above expression with:

$$\lim_{\Delta x\to 0}\frac{4}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)=\xi^2\,. \tag{6.24}$$

The result is the following equation:

$$\begin{aligned}e^{i\xi j\Delta x}\frac{4}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)&\left\{\overline{D}_c\Upsilon_c\left(A'_c-A_c\right)+\sum_{g\in c}D_g a_g\right\}\\&+e^{i\xi j\Delta x}\overline{\Sigma}_{t,c}\Upsilon_c\left(A'_c-A_c\right)+e^{i\xi j\Delta x}\sum_{g\in c}\Sigma_{t,g}a_g\\&-e^{i\xi j\Delta x}\sum_{c'=1}^{C}\left\{\overline{\Sigma}_{s0,j,c'\to c}\Upsilon_{c'}\left(A'_{c'}-A_{c'}\right)+\sum_{g'\in c'}\sum_{g\in c}\Sigma_{s0,j,g'\to g}a_{g'}\right\}\\&=\lambda'_1\left\langle\chi_c\right\rangle\sum_{c'=1}^{C}\overline{\nu\Sigma}_{f,c'}\Upsilon_{c'}+\lambda_0\left\langle\chi_c\right\rangle e^{i\xi j\Delta x}\sum_{c'=1}^{C}\overline{\nu\Sigma}_{f,c'}\Upsilon_{c'}\left(A'_{c'}-A_{c'}\right)\\&\qquad+\lambda_0\left\langle\chi_c\right\rangle e^{i\xi j\Delta x}\sum_{c'=1}^{C}\sum_{g\in c'}\nu\Sigma_{f,g}a_g\,.\end{aligned} \tag{6.25}$$

All of the terms except the one with λ'_1 depend on the spatial index j. Thus, this analysis only yields nontrivial results when

$$\boxed{\lambda'_1=0}\,. \tag{6.26}$$

Eliminating that term and dividing by $e^{i\xi j\Delta x}$ yields

$$\frac{4}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)\left\{\overline{D}_c\Upsilon_c\left(A_c'-A_c\right)+\sum_{g\in c}D_g a_g\right\}+\overline{\Sigma}_{t,c}\Upsilon_c\left(A_c'-A_c\right)+\sum_{g\in c}\Sigma_{t,g}a_g$$
$$-\sum_{c'=1}^{C}\left\{\overline{\Sigma}_{s0,j,c'\to c}\Upsilon_{c'}\left(A_{c'}'-A_{c'}\right)+\sum_{g'\in c'}\sum_{g\in c}\Sigma_{s0,j,g'\to g}a_{g'}\right\}$$
$$=\lambda_0\left\langle\chi_c\right\rangle\sum_{c'=1}^{C}\left\{\overline{\nu\Sigma}_{f,c'}\Upsilon_{c'}\left(A_{c'}'-A_{c'}\right)+\sum_{g\in c'}\nu\Sigma_{f,g}a_g\right\}. \quad (6.27)$$

Some reorganization yields

$$\left[\frac{4\overline{D}_c}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)+\overline{\Sigma}_{t,c}\right]\Upsilon_c\left(A_c'-A_c\right)-\sum_{c'=1}^{C}\left[\overline{\Sigma}_{s0,j,c'\to c}+\lambda_0\left\langle\chi_c\right\rangle\overline{\nu\Sigma}_{f,c'}\right]\Upsilon_{c'}\left(A_{c'}'-A_{c'}\right)$$
$$=-\frac{4}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)\sum_{g\in c}D_g a_g-\sum_{g\in c}\Sigma_{t,g}a_g+\sum_{c'=1}^{C}\sum_{g'\in c'}\left[\lambda_0\left\langle\chi_c\right\rangle\nu\Sigma_{f,g'}+\sum_{g\in c}\Sigma_{s0,j,g'\to g}+\right]a_{g'}. \quad (6.28)$$

Finally, we re-express the triple summation in the final term with a simpler double summation, and get

$$\left[\frac{4\overline{D}_c}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)+\overline{\Sigma}_{t,c}\right]\Upsilon_c\left(A_c'-A_c\right)-\sum_{c'=1}^{C}\left[\overline{\Sigma}_{s0,j,c'\to c}+\lambda_0\left\langle\chi_c\right\rangle\overline{\nu\Sigma}_{f,c'}\right]\Upsilon_{c'}\left(A_{c'}'-A_{c'}\right)$$
$$=-\frac{4}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)\sum_{g\in c}D_g a_g-\sum_{g\in c}\Sigma_{t,g}a_g+\sum_{g'=1}^{G}\left[\lambda_0\left\langle\chi_c\right\rangle\nu\Sigma_{f,g'}+\sum_{g\in c}\Sigma_{s0,j,g'\to g}\right]a_{g'}. \quad (6.29)$$

The above equation can be written in matrix form as:

$$\boxed{M_1\left(\boldsymbol{A}'-\boldsymbol{A}\right)=M_2\boldsymbol{a}}. \quad (6.30)$$

Here, M_1 is a $C\times C$ matrix, M_2 is a $C\times G$ matrix, $\boldsymbol{A}'-\boldsymbol{A}$ is a length-C vector, and $\boldsymbol{a}$ is a length-G vector.

Next, we substitute the Fourier ansatz into Equation (6.12).

$$\phi_{j,g}^{(l')}=\frac{\Phi_{j,c}^{(l')}}{\Phi_{j,c}^{(l)}}\phi_{j,g}^{(l)},$$

$$\varphi_g + \epsilon a'_g \omega^l e^{i\xi j\Delta x} = \left[1 + \epsilon A'_c \omega^l e^{i\xi j\Delta x}\right] \left[1 + \epsilon A_c \omega^l e^{i\xi j\Delta x}\right]^{-1} \left[\varphi_g + \epsilon a_g \omega^l e^{i\xi j\Delta x}\right] .$$

Expanding the products and eliminating terms of $O(\epsilon^2)$, we obtain an expression for a'_g in terms of A_c, A'_c, and a_g:

$$\varphi_g + \epsilon a'_g \omega^l e^{i\xi j\Delta x} \approx \left[1 + \epsilon A'_c \omega^l e^{i\xi j\Delta x}\right] \left[1 - \epsilon A_c \omega^l e^{i\xi j\Delta x}\right] \left[\varphi_g + \epsilon a_g \omega^l e^{i\xi j\Delta x}\right] ,$$

$$\varphi_g + \epsilon a'_g \omega^l e^{i\xi j\Delta x} \approx \varphi_g + \epsilon \omega^l e^{i\xi j\Delta x} \left[\varphi_g \left(A'_c - A_c\right) + a_g\right] ,$$

$$\boxed{a'_g = \varphi_g \left(A'_c - A_c\right) + a_g} . \tag{6.31}$$

As in Section 4.2, substituting the Fourier ansatz into Equations (6.13b) and (6.13c) yields

$$a''_g = a_g , \tag{6.32a}$$

$$\phi_g^{(l'')} = \phi_g^{(l+1)} , \tag{6.32b}$$

$$\lambda_1 = 0 . \tag{6.32c}$$

These results are due to the periodicity of the solution. Finally, we substitute the Fourier ansatz and Equations (6.32) into Equation (6.13a), note that all terms of $O(1)$ again cancel by the definitions of φ_g and λ_0, divide both sides by $\epsilon \omega^l e^{i\xi j\Delta x}$, and use Equation (6.23) to obtain

$$\omega \left[\frac{4D_g}{\Delta x^2} \sin^2\left(\frac{\xi \Delta x}{2}\right) + \Sigma_{t,g}\right] a_g - \omega \sum_{g'=1}^{G} \Sigma_{s0,g'\to g} a_{g'} = \lambda_0 \chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'} a'_{g'} . \tag{6.33}$$

In matrix notation, the above equation can be written as

$$\boxed{\omega M_3 \boldsymbol{a} = M_4 \boldsymbol{a}'} . \tag{6.34}$$

Both M_3 and M_4 are $G \times G$ matrices, and both $\boldsymbol{a}$ and $\boldsymbol{a}'$ are length-G vectors. Equations (6.30), (6.31), and (6.34) relate the variables ω, $\boldsymbol{a}$, $\boldsymbol{a}'$, and $\boldsymbol{A}' - \boldsymbol{A}$. They can be combined to eliminate $\boldsymbol{a}'$ and $\boldsymbol{A}' - \boldsymbol{A}$, yielding a $G \times G$ eigenvalue problem of the form

$$\boxed{\omega \boldsymbol{a} = M_5 \boldsymbol{a}} . \tag{6.35}$$

M_5 is given by a product of five matrices:

$$M_5 = M_3^{-1} M_4 M_{C\to G} M_1^{-1} M_2 . \tag{6.36}$$

Here, $M_{G\to C}$ is the $G \times C$ interpolation matrix for Equation (6.31); the entry in the g-th row and c-th column of $M_{G\to C}$ is defined by

$$M_{c\to g} \equiv [M_{C\to G}]_{g,c} \equiv \begin{cases} \dfrac{\varphi_g}{\Upsilon_c} + 1\,, & g \in c\,, \\ 0\,, & g \notin c\,. \end{cases} \tag{6.37}$$

For a given D_g, Δx, ξ, and set of cross sections, we can use Equation (6.35) to numerically compute ω. The spectral radius ρ is then given by the maximum magnitude of ω over all possible error mode frequencies ξ.

From this Fourier analysis, we see that the multilevel-in-energy method developed in this chapter, when linearized, is equivalent to a multigrid method. The right side of Equation (6.29) is the residual of the multigroup flux, mapped onto a coarser energy grid, and Equation (6.29) is used to solve for the coarse-grid error $\boldsymbol{A}' - \boldsymbol{A}$. In Equation (6.31), this error is then interpolated to the original energy grid and applied as a correction to the fine-grid scalar flux. Finally, we perform a "smoothing step" on the original energy grid by using Equation (6.33) to eliminate any remaining error modes that are strongly energy-dependent. The observation that Algorithm 9 is effectively a multigrid algorithm when linearized is consistent with the observations of [57], which showed that CMFD is equivalent to DSA when linearized.

6.2.3 Numerical Results

In this subsection, we take the analytical formulas derived in the previous subsection, and we compute sample decay factors and spectral radii numerically. The analysis in this chapter assumes that, at every MED/MSED iteration, both the grey eigenvalue problem (Equations (6.13)) and multigroup fixed-source problem (Equation (6.13a)) are solved exactly. Because of this, the spectral radii computed here are lower bounds for the true spectral radius. The case in which the multigroup fixed-source problem is not fully solved at every MSED iteration is considered later in Section 7.2.

The numerical results are presented in Table 6.1 and Figures 6.1 and 6.2. Cross-sections and diffusion coefficients used in the computations are obtained by homogenizing the center pin cell in problem 2a[3] for four different MPACT cross-section libraries. We used $X = 100$ for this analysis. From these results, we draw the following conclusions:

1. The spectral radius of MED (Algorithm 9) is approximately 0.16 as $G \to \infty$.[4]

[3]See Appendix C for more details.

[4]This differs slightly from the results of [10] because we are using cross-sections from a different pin cell.

2. Using two groups instead of one on the coarsest energy grid provides a tangible reduction in the spectral radius (from $\sim$0.16 to $\sim$0.1).

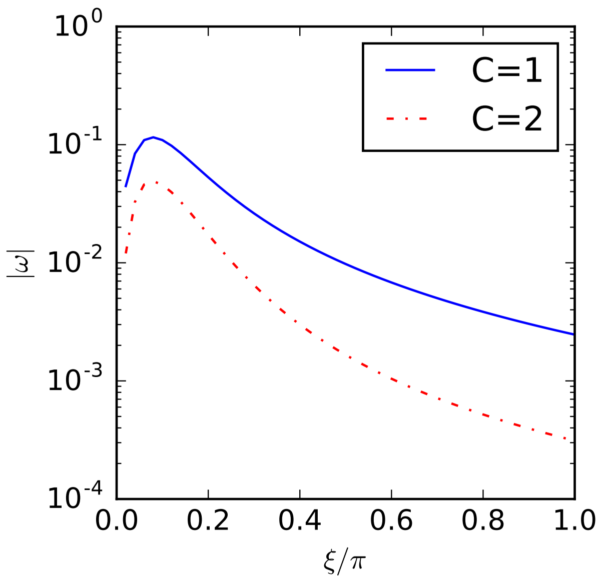
3. The spectral radii are very weakly dependent on Δx.

Table 6.1: Spectral radii computed via Fourier analysis for the MED method described by Algorithm 9 ($C = 1$) and a multilevel-in-energy method with $C = 2$. These spectral radii are computed by taking the maximum of the decay factors ω over ξ. These decay factors are shown in Figure 6.1.

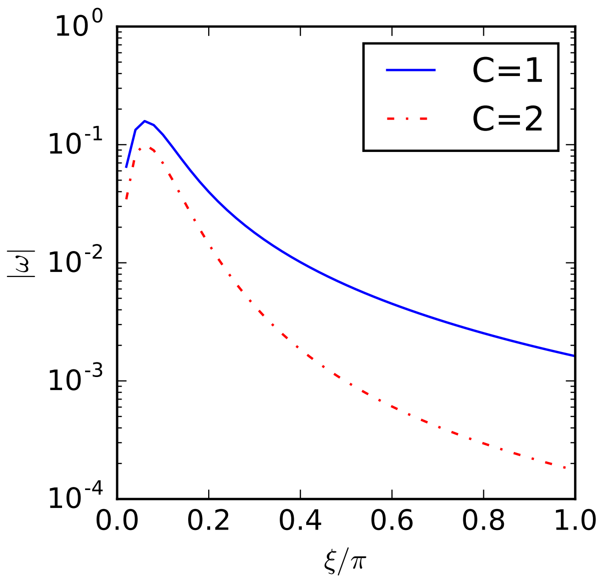
Library	$C = 1$	$C = 2$
MPACT 8G	0.115	0.049
MPACT 47G	0.158	0.100
MPACT 51G	0.162	0.105
MPACT 252G	0.164	0.107

The first and second conclusions are drawn from the results of Table 6.1 and Figure 6.1. These results indicate that MED/MSED has a small spectral radius and that the grey diffusion problem resolves a significant portion of the solution by itself, even for large G. We see that using a multilevel method with $C = 2$ (two-group diffusion) instead of $C = 1$ (grey diffusion) results in a tangible reduction to the already small spectral radius of MED. The relative importance of this difference can be better understood by considering the number of iterations required to converge to some tolerance. For example, we can compute the number of iterations needed to converge to a relative tolerance of 10^{-6}. In the case of $C = 1$, we have a spectral radius of 0.16, which means that approximately 8 iterations would be needed ($0.16^7 > 10^{-6} > 0.16^8$). On the other hand, we have a spectral radius of 0.1 for $C = 2$, which means that only $\sim$6 iterations would be needed to converge ($0.1^6 = 10^{-6}$). As noted earlier, however, this improvement in the spectral radius comes with a drawback. Because two-group problems are more expensive to solve than grey problems, additional work is required on the coarse energy grid. The numerical results of Section 6.4 provide a more realistic comparison of the two options.

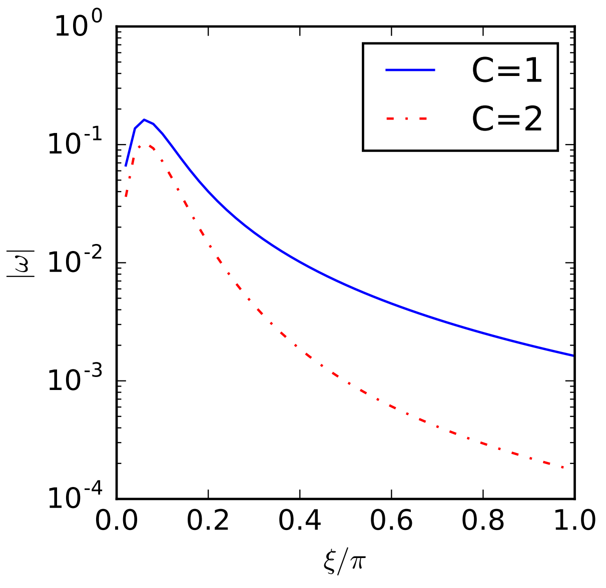
Although the spectral radius grows as G increases from $G = 8$ to $G = 47$, the spectral radii are nearly identical for all $G \in \{47, 51, 252\}$. These results indicate that MED should work well for all G, but they *do not* contradict the work in [66], which indicates that the introduction of intermediate energy grids can reduce the overall runtime. We reiterate that the analysis performed here assumes that the multigroup linear system in step 4 of Algorithm 9 is solved exactly; our analysis does not account for the cost of solving this system. Although these results indicate that the spectral radius does not grow as $G \to \infty$, the cost of each MED or MSED iteration grows with G due to the growth of the size of the multigroup system. This growth is superlinear, because the number of nonzeros in the matrix scales as G^2. To mitigate this cost, one can reduce the convergence requirements on the



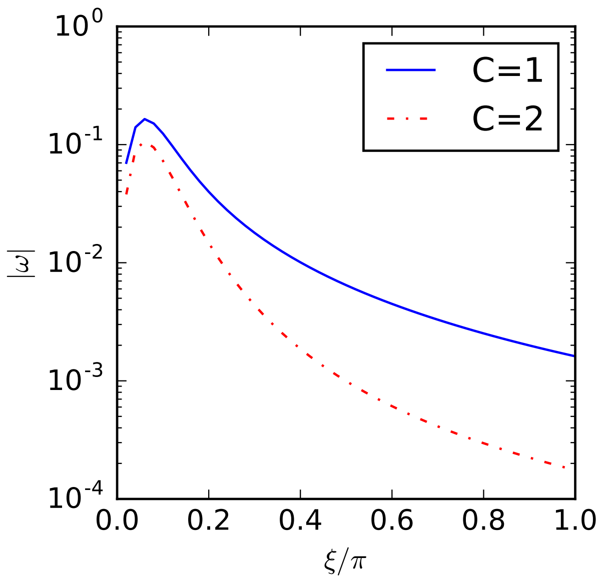
(a) MPACT 8G library.



(b) MPACT 47G library.



(c) MPACT 51G library.



(d) MPACT 252G library.

Figure 6.1: Decay factors from Fourier analysis of the MED method described by Algorithm 9 ($C = 1$) and a multilevel-in-energy method with $C = 2$. Here, C is the number of groups used on the coarse-group problem, and $\Delta x = 0$ (i.e., we are looking at the case without spatial discretization). Cross sections are obtained by a homogenization of the center pin-cell in problem 2a using four different MPACT libraries. The spectral radius is given by the maximum value of $|\omega|$ over all values of ξ.

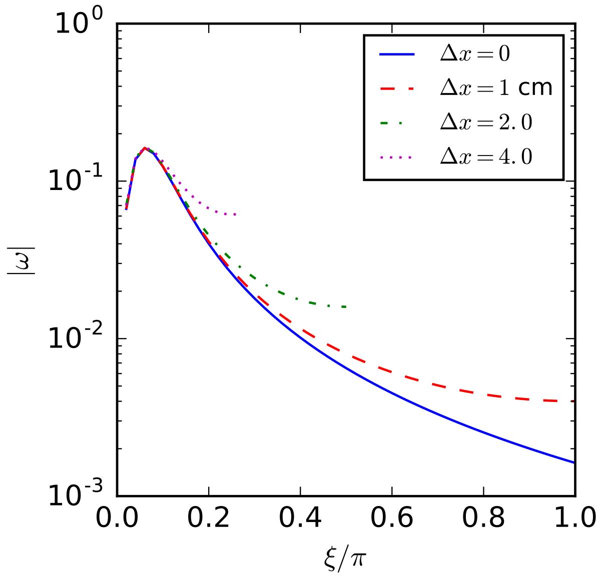


Figure 6.2: Decay factors from Fourier analysis of the MED method described by Algorithm 9 for different values of Δx. Here, the case using the MPACT 51G library (Figure 6.1c) is shown. For $\Delta x > 0$, the domain of ξ is truncated by the rule in Equation (6.17).

multigroup linear solve (e.g., use fewer upscattering iterations in Algorithm 11) and perform work on intermediate energy grids instead. Such an approach is beyond the scope of this thesis, but it is effectively what was done by Cornejo et al. in [66].

The third conclusion is evident in Figure 6.2, as the spectral radius (maximum value of $|\omega|$) is essentially the same for all of the curves in that plot. This conclusion is expected, since the spatial discretization only affects the difficulty of solving the linear systems in steps 2 and 4 of Algorithm 9. When we assume that the linear systems in steps 2 and 4 of Algorithm 9 are solved exactly, the spatial discretization no longer plays a role. In practice, however, a smaller Δx results in a larger, more ill-conditioned problem matrices. This can prevent MED/MSED from achieving the convergence rates shown in Table 6.1 if the linear solver used in MED/MSED does not produce adequately converged solutions in steps 2 and 4 of Algorithm 9. If solving the low-order system with C groups is difficult to due a large spatial mesh, the approach in [1] may be beneficial. There, Rhodes and Smith explored a method in which one iterates between a pin-wise G-group CMFD problem and an assembly-wise (i.e., spatially coarsened) 2-group CMFD problem. A tradeoff occurs when the low-order system is coarsened in both energy and space. The low-order problem becomes easier to solve, but the spectral radius of the overall method would be higher because the low-order problem is now a less accurate representation of the full multigroup problem.

6.3 1-D Diffusion Results

In this section, the 1-D code described in Section 3.1 is used to generate results for the WB-1D-1 and WB-1D-2 problems described in Appendix B. These results are shown in Tables 6.2 and 6.3. Here, we assess the benefit of using a grey diffusion system via the MED method described in Algorithm 9 for four different linear solvers: GMRES with an ILU preconditioner, BiCGSTAB with a block-Jacobi preconditioner, BiCGSTAB with an ILU preconditioner, and multigrid-in-space. (This details of multigrid are described in Chapter 7.) MED is compared to the Wielandt-shifted PI scheme (Algorithm 4) used to generate the results in Chapter 4. The adaptive PARCS WS described by Equation (2.40), with $c_0 = 0.01$, is used for the multigroup system in the PI scheme and for the grey system in the MED method.[5] As noted previously, one of the primary benefits of using a grey diffusion system is that a WS is no longer needed on the multigroup system.

The abbreviations used in Tables 6.2 and 6.3 are defined as follows. MGPIs stands for multigroup power iterations. For PI, this is simply the number of power iterations performed. For MED, this is the number of times the multigroup linear system is solved (i.e., the number of times step 4 is performed in Algorithm 9), and it is also equal to the number of MED iterations. MGLSIs stands for multigroup linear solver iterations; this is the total number of linear solver iterations required for all of the MGPIs in the simulation. GPIs (grey PIs) and GLSIs (linear solver iterations) are the corresponding quantities for the grey diffusion system in the MED method.

For all of the linear solvers and for both problems, the results in Tables 6.2 and 6.3 indicate that the use of a grey diffusion equation provides a significant speedup. When MED is used instead of Wielandt shifted PI, we see reductions in the runtime ranging from 30% to 82%. From the iteration counts in the tables, we see that the speedups are primarily due to a shift of the computational burden from the multigroup system to the grey system. For most of the problems, the number of MGLSIs is reduced by an order of magnitude by the use of a grey system. Although the MED method still requires a significant number of GLSIs, each GLSI only requires a small fraction of the computational effort that an MGLSI requires. As we noted earlier, this reduction in MGLSIs is a result of two primary factors: (1) the work performed by the grey diffusion system to resolve the overall spatial dependence, fission source, and eigenvalue, and (2) the lack of a WS on the multigroup system. The first factor reduces the number of MGPIs required, while the second factor reduces the number of MGLSIs required per MGPI.

[5]The value of c_0 in the PARCS WS had to be increased to 0.03 for the PI method with multigrid in Table 6.3 due to multigrid being unable to handle an overshifted diffusion system. When multigrid was used in the MED method, we were able to use $c_0 = 0.01$. This reinforces our assertion that it is easier to solve a shifted grey diffusion system than a shifted multigroup diffusion system.

Table 6.2: Comparison of MED (Algorithm 9) to Wielandt shifted PI (Algorithm 4) for the WB-1D-1 problem. Iteration counts and runtimes are shown for four different linear solver options.

Method	Linear Solver	MGPIs	MGLSIs	GPIs	GLSIs	Runtime [s]
MED	GMRES-ILU	8	20	70	70	12.8
PI	GMRES-ILU	15	90	–	–	19.3
MED	BiCGSTAB-BJ	8	241	70	5541	19.7
PI	BiCGSTAB-BJ	15	2118	–	–	110.7
MED	BiCGSTAB-ILU	8	11	70	70	12.7
PI	BiCGSTAB-ILU	15	47	–	–	18.4
MED	Multigrid	8	22	70	914	9.8
PI	Multigrid	15	189	–	–	17.5

Table 6.3: Comparison of MED (Algorithm 9) to Wielandt shifted PI (Algorithm 4) for the WB-1D-2 problem. Iteration counts and runtimes are shown for four different linear solver options.

Method	Linear Solver	MGPIs	MGLSIs	GPIs	GLSIs	Runtime [s]
MED	GMRES-ILU	7	18	72	79	9.5
PI	GMRES-ILU	13	75	–	–	17.7
MED	BiCGSTAB-BJ	7	223	80	7899	19.0
PI	BiCGSTAB-BJ	13	1767	–	–	91.1
MED	BiCGSTAB-ILU	7	10	79	79	9.4
PI	BiCGSTAB-ILU	13	34	–	–	15.3
MED	Multigrid	7	20	81	645	7.2
PI	Multigrid	20	195	–	–	20.6

An approximate sense of the relative costs of the MGLSIs and GLSIs in the 1-D code can be obtained by considering the following model:

$$(\text{\# of MGLSIs})(\text{Cost of MGLSI}) + (\text{\# of GLSIs})(\text{Cost of GLSI}) + (\text{Overhead Costs}) = \text{Runtime}\,. \tag{6.38}$$

There are three unknowns in this equation (the cost of an MGLSI, the cost of an GLSI, and the overhead cost), and we have four "data points" for each linear solver. We can perform a least-squares fit of the data in Tables 6.2 and 6.3 to the above model to obtain estimates for these three unknowns[6]. The results are shown in Table 6.4. The model in Equation (6.38) is far from a perfect model, and there are obvious flaws in the generation of the data in Table 6.4 (small sample size, overhead cost has a weak dependence on the number of PIs, some of the GLSIs costs are unrealistically small, etc.). Nonetheless, the model reinforces our claim that performing LSIs on the multigroup system is significantly more costly than performing LSIs on the grey system. In all of the cases, the estimated MGLSIs cost is more than an order of magnitude greater than the estimated GLSIs cost. Moreover, the model indicates that the bulk of the cost for ILU-preconditioned methods is in the overhead, which is consistent with our knowledge of the ILU preconditioner.

Lastly, we note that MED converged in 7 and 8 iterations for the two problems respectively. The outermost convergence criteria for these simulations is 10^{-6}. Thus, 7-8 iterations is consistent with the spectral radius of 0.16 predicted by the Fourier analysis ($0.16^7 > 10^{-6} > 0.16^8$). Although an idealized toy problem (homogeneous, periodic boundary conditions) is used in the Fourier analysis, the predicted spectral radii from this analysis can provide a reasonable estimate of the true convergence behavior in more realistic problems (heterogeneous, vacuum boundary conditions).

Table 6.4: Estimated LSI costs and overhead costs in the 1-D diffusion code. These numbers are obtained by a least squares fit of the data in Table 6.2 and 6.3 to the model in Equation (6.38).

Linear Solver	Overhead [s]	GLSI Cost [s]	MGLSI Cost [s]
GMRES-ILU	8.9	2.4E-26	1.1E-1
BiCGSTAB-BJ	3.5	5.9E-04	5.0E-2
BiCGSTAB-ILU	8.9	2.0E-14	2.0E-1
Multigrid	2.1E-12	8.2E-03	9.9E-2

[6]The fit was performed using SciPy's `optimize.lsq_linear` routine, with constraints ensuring that, for a given linear solver, the estimated costs are positive and the estimated overhead does not exceed any of the total runtimes [96].

6.4 Selected MPACT Results (2-D and 3-D CMFD)

In this section, we examine the performance of the MED method (i.e., the grey diffusion acceleration/ multilevel-in-energy method described in Algorithm 9) as a CMFD solver for 2-D and 3-D problems in MPACT. This section is organized into three subsections. The first subsection examines the performance of the three different approaches described in Section 6.1.2, the second subsection examines the potential benefits of using a two-group diffusion problem instead of a grey diffusion problem, and the final subsection compares MED to existing CMFD solvers in MPACT.

For all of the problems in MPACT simulated using MED, GMRES is used as the linear solver. (Again, we note that MED is effectively MSED without the multigrid-in-space linear solver described in Chapter 7.) Unless the one-group sweep approach (Algorithm 11) is used, the maximum number of GMRES permitted per PI is 100 if the problem uses the default $\lambda' = 2/3$ shift or no shift, and it is 200 otherwise. The convergence criteria and the iteration scheme of MPACT are described in Section 3.2.

6.4.1 Comparison of Multigroup Linear Solver Approaches in MED

First, we compare the performance of the three options for solving the multigroup linear system discussed in Section 6.1.2 – the standard approach, the reduced multigroup approach (Algorithm 10), and the one-group sweep approach (Algorithm 11). To do this, we consider the following two problems in MPACT: problem 5a-0 with the 51-group MPACT library and problem 5a-2D with the 252-group MPACT library. Problem 5a-0 is a quarter-core 3-D model simulated using 464 cores over 29 nodes, and problem 5a-2D is a 2-D model simulated using 32 cores over 4 nodes. Both problems were run on the Eos supercomputer at the Oak Ridge Leadership Computing Facility (OLCF). More details regarding these problems, which are VERA benchmark progression problems [6], can be found in Appendix C.

Iterations, runtimes, and memory costs are shown in Tables 6.5 and 6.6 for the 51-group and 252-group problems, respectively. In these tables, the “Mem.” column gives the maximum memory required by any processor over the course of the simulation in units of GB; this includes both the CMFD and non-CMFD components of MPACT. The “TSs” column gives the number of transport sweeps required over the entire simulation. TR is the total runtime required by the simulation, CR is the CMFD runtime required, and GR is the runtime required in the grey system. CR is the primary quantity of interest in this thesis, and the CR values are bolded in the tables.

The remaining abbreviations are the same as those used in Tables 6.2 and 6.3. However, for the reduced multigroup and one-group sweep approaches, MGLSIs have a different meaning. For the reduced multigroup approach, the number listed under MGLSIs only accounts for the LSIs performed on the thermal system. It does not include the LSIs performed on the one-group problems used to solve the non-thermal groups. The purpose of the MGLSIs column is to provide a sense of the amount of work performed on the multigroup system. For the reduced multigroup approach, the bulk of the work lies in the MGLSIs performed on the thermal groups, so this is the quantity that is reported. For the one-group sweep approach, however, there are no "multigroup" LSIs since only one-group is solved at a time at any step of Algorithm 11. Thus, the value reported under MGLSIs for the one-group sweep approach is the sum of all the one-group LSIs performed on each group of the multigroup system.

Table 6.5: Comparison of three different multigroup solver options for MED in MPACT for problem 5a-0 with the 51-group MPACT library. Iteration counts, runtimes, and memory costs are shown for each option; runtimes are given in seconds while memory is given in GB.

Approach	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Standard	12	37	2140	160	21794	2.82	23	**132**	1022
Reduced MG	12	37	1226	160	21794	2.67	23	**72**	988
1G Sweep	12	71	42700	258	32069	2.62	38	**98**	1021

Table 6.6: Comparison of three different multigroup solver options for MED in MPACT for problem 5a-2D with the 252-group MPACT library.

Approach	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Standard	12	36	2366	173	23879	5.2	18	**567**	1983
Reduced MG	12	35	1365	171	23762	4.32	19	**192**	1451
1G Sweep	12	56	220846	229	29698	3.84	29	**139**	1396

As predicted in Section 6.1.2, the reduced multigroup approach is significantly better than the standard approach in terms of both runtime and memory for both problems considered. The results also reinforce our assertion in Section 6.1.2 that the one-group sweep approach is more efficient for large values of G while the reduced multigroup approach is more efficient for intermediate and smaller values of G. In Table 6.5, we see that the reduced multigroup approach is faster than the one-group sweep approach for the 51-group problem, with only a minor increase in the memory required. For the 252-group problem shown in Table 6.6, however, we see that the one-group sweep approach is faster and requires less memory than the reduced multigroup approach. We note that the difference in MGPIs required in the 252-group problem is due to the incomplete convergence of the multigroup system via Algorithm 11. If m_{max} in Algorithm 11 were increased to a much larger number, the number of MGPIs would be the same for all of the methods in Tables 6.5 and 6.6.

We believe that the conditions under which the one-group sweep approach is more efficient than the reduced multigroup approach are the same as the conditions under which the MED/MSED algorithm would benefit from intermediate energy grids and/or coarse energy grids with more than 1 group. As indicated by Tables 6.5 and 6.6, a very large number of LSIs are needed on the multigroup system for the one-group approach. This number can be significantly reduced by the introduction of intermediate grids (as evidenced in [66]) or by using a coarse energy grid with 2 or more groups (this is demonstrated in the next subsection).

In light of these results, we adopt the following conventions in the remainder of this thesis for the MED method. Unless otherwise stated, the MED method will solve any problems in MPACT with 252-groups or more using the one-group sweep approach, and it will solve any problems in MPACT with 51-groups or fewer using the reduced multigroup approach. For MSED, the one-group sweep approach is used for problems with 252-groups, and the reduced multigroup approach is used for smaller problems with 51-groups or fewer. However, we are still using the standard approach for full-core problems due to inefficiencies in our implementation related to MPI communication between processors. This issue is discussed in Section 8.2.

6.4.2 Grey Diffusion vs. 2-Group Diffusion

In this section, we compare the performance of two approaches: the MED method with a grey diffusion system described by Algorithm 9, and an analogous method that uses a 2-group diffusion system instead of a grey diffusion system. We consider the same two problems from the previous section: problem 5a-0 with 51-groups and problem 5a-2D with 252-groups. The results are shown in Tables 6.7 and 6.8, respectively, and the abbreviations used in these two tables are the same as those used in Tables 6.5 and 6.6. The only difference is that "grey" refers to a 2-group system in the $C = 2$ case. Moreover, as noted in the previous section, we use the reduced multigroup approach in Table 6.7 and the one-group sweep approach in Table 6.8. We remind the reader C is the variable used to denote the number of groups in the coarse energy grid. ($C = 1$ refers to a grey diffusion system, while $C = 2$ refers to a two-group diffusion system.) It is *not* related to the number of energy grids being used.

Table 6.7: Grey diffusion vs. 2-group diffusion for problem 5a-0 in MPACT with 51 groups. Iteration counts, runtimes, and memory costs are shown for each option; runtimes are given in seconds while memory is given in GB.

Approach	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
$C = 1$	12	37	1226	160	21794	2.67	23	**72**	988
$C = 2$	12	33	1047	136	18053	2.70	37	**79**	998

Table 6.7 indicates that the differences between $C = 1$ and $C = 2$ are negligible for the 51-group problem. Although using $C = 2$ results in fewer MGLSIs, this savings is offset by the extra cost of

Table 6.8: Grey diffusion vs. 2-group diffusion for problem 5a-2D in MPACT with 252 groups.

Approach	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
$C = 1$	12	56	220846	229	29698	3.84	29	**139**	1396
$C = 2$	12	46	119101	197	26708	3.84	25	**101**	1353

the larger coarse energy grid. That is, LSIs on the coarse energy grid (listed as GLSIs) are more costly for higher C. For 252-group problem in Table 6.8, using $C = 2$ reduces the CMFD runtime required by $\sim 30\%$. As noted in Section 6.2, the difference in performance between $C = 1$ and $C = 2$ for the 252-group problem is due to differences in the cost of solving the fixed-source multigroup CMFD problem. The spectral radii for the two values of C are similar, but the cost of solving the fixed-source multigroup problem differ for this 252-group problem. When $C = 2$ is used, fewer MGLSIs are required because the coarse energy grid is able to resolve some of the energy dependence, reducing the burden on the linear solver for the multigroup system. The results here are consistent with our earlier claim that the use of additional groups in the coarse grid becomes more beneficial as G increases.

Although we see that using $C = 2$ can provide additional speedups for large G, the consideration of $C > 1$ is mostly beyond the scope of the work in this thesis. The purpose of this section is only to illuminate the situations in which using $C > 1$ may be beneficial.

6.4.3 Comparison of MED to Other CMFD Solvers

In this section, we compare the performance of MED to other CMFD solvers in MPACT to quantify the improvements provided by the use of a grey diffusion system in Algorithm 9. Here, we consider seven problems: 4a, Peach Bottom 2-D (PB), 5a-0, 5a-2D, 5a-0-FC, AP1000-BOL, and AP1000-EOC. Iterations, runtimes, and memory costs are shown for these problems in Tables 6.9-6.15. In this thesis, runtimes are given in seconds while memory is given in GB. All of the problems use the 51-group library except problem 5a-2D, which uses the 252-group library. Problems 4a, PB, 5a-0, and 5a-2D were run on Eos, while the remaining three problems were run on Titan. (Both Eos and Titan are machines at the Oak Ridge Leadership Computing Facility (OLCF).) All of the problems are 3-D problems except for problems PB and 5a-2D. The "numbered" problems (4a, 5a-0, 5a-2D, 5a-0-FC) are drawn from the VERA Progression Problems [112]. Problem 4a consists of nine Westinghouse 17x17 fuel assemblies, while problems 5a-0 and 5a-2D are quarter-core models of the Watts Bar Unit 1 reactor. Problem 5a-0-FC is a full-core (FC) version of problem 5a-0, and AP1000-BOC and AP1000-EOC are full-core hot zero power (HZP) models of the AP1000 reactor at beginning of cycle (BOC) and end of cycle (EOC), respectively. Appendix C provides additional details for these problems.

Table 6.9: Comparison of MED to other MPACT CMFD solvers for the 3-D, 9 assembly problem 4a. Iteration counts, runtimes, and memory costs are shown for each option. In all of the tables in this thesis, runtimes are given in seconds while memory is given in GB.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	12	104	9919	–	–	0.89	–	**152**	367
PI+PWS	12	49	6910	–	–	0.89	–	**115**	340
RBBJ	14	280	42000	–	–	0.84	–	**336**	589
GD	12	595	–	–	–	0.96	–	**29**	241
MED	12	38	936	147	17699	0.87	6	**15**	230

Table 6.10: Comparison of MED to other MPACT CMFD solvers for the 2-D, full-core Peach Bottom (PB) problem.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	157	576	46615	–	–	0.94	–	**275**	1319
PI+PWS	21	153	24862	–	–	0.94	–	**168**	428
GD	27	4714	–	–	–	0.97	–	**95**	373
MED	27	91	2668	1108	181572	0.94	50	**65**	350

Table 6.11: Comparison of MED to other MPACT CMFD solvers for the 3-D, quarter-core problem 5a-0.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	21	96	9600	–	–	2.72	–	**632**	1998
PI+PWS	12	52	8002	–	–	2.72	–	**558**	1484
RBBJ	15	300	45000	–	–	2.56	–	**1551**	2684
GD	12	749	–	–	–	3.01	–	**130**	1026
MED	12	37	1226	160	21794	2.67	23	**72**	988

Table 6.12: Comparison of MED to other MPACT CMFD solvers for the quarter-core problem 5a-2D with MPACT's 252 group library.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	12	89	8900	–	–	4.74	–	**2904**	4510
PI+PWS	9	48	8275	–	–	4.74	–	**2943**	4433
RBBJ	12	213	31950	–	–	4.33	–	**14386**	15907
GD	–	–	–	–	–	–	–	–	DNC
MED	12	56	220846	229	29698	3.84	29	**139**	1396

Table 6.13: Comparison of MED to other MPACT CMFD solvers for the 3-D, full-core problem 5a-0-FC.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	24	94	9400	–	–	2.39	–	**1918**	5015
PI+PWS	12	53	8351	–	–	2.39	–	**1770**	4124
RBBJ	–	–	–	–	–	–	–	–	DNC
GD	12	1034	–	–	–	2.54	–	**1830**	4030
MED	12	37	1273	156	21453	2.16	224	**436**	2474

Table 6.14: Comparison of MSED to other MPACT CMFD solvers for the 3-D, full-core AP1000-BOC problem.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	27	126	12600	–	–	2.17	–	**2337**	6033
PI+PWS	11	53	8990	–	–	2.17	–	**1713**	3916
RBBJ	16	316	47400	–	–	2.11	–	**2755**	5414
GD	8	643	–	–	–	–	–	–	DNC
MED	11	33	1670	192	30614	2.19	353	**587**	3235

Table 6.15: Comparison of MSED to other MPACT CMFD solvers for the 3-D, full-core AP1000-EOC problem.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	95	417	41700	–	–	2.93	–	**2776**	19268
RBBJ	44	880	132000	–	–	2.80	–	**6939**	17548
GD	15	1502	–	–	–	2.96	–	**1987**	6778
MED	15	41	2814	280	45016	2.94	298	**441**	5278

In these results, the "default" CMFD solver in MPACT refers to using PI with a fixed WS of $\lambda' = 2/3$, RBBJ is the same as the "default" solver except that "red-black" block Jacobi is used as the linear solver instead of GMRES, "PI+PWS" uses the PARCS WS instead of $\lambda' = 2/3$, and GD refers to the Generalized Davidson method, preconditioned with an algebraic multigrid preconditioner. In Section 3.2, additional details regarding these four methods and their implementation in MPACT is provided. Due to time and supercomputing resource constraints, results for some of the methods are not available in all problems.

For each of the seven problems, we see that the MED method provides a significant reduction in the runtime compared to the default MPACT method. These speedups are highly problem-dependent, and they range from $\sim$4-10x for the 51-group problems. For the 252-group problem in Table 6.12, the speedup is $\sim$21x. As in the 1-D results, this speedup is primarily due to the shifting of much of the workload from the multigroup system to the grey system. Although MED requires many LSIs, it reduces the number of MGLSIs required by approximately an order of magnitude. Similar speedups are seen when MED is compared to the PI+PWS method. Moreover, for all of the problems except AP1000-EOC, we see that the MED method has a smaller memory burden because it does not need to form the full multigroup matrix. This reduction in memory cost is especially noticeable for the 252-group problem 5a-2d in Table 6.12. MED uses the one-group sweep approach in the 252 group problem, and it uses the reduced multigroup approach for five of the remaining six problems, which use 51 groups. The standard approach is used for problem AP1000-EOC due to performance issues in the MPI communication; these issues are discussed further in Section 8.2.

For most of the problems, the savings in the CMFD runtime does not fully capture the benefit of using MED. This is because extra transport sweeps are often required when the default CMFD solver is used instead of MED. For example, the savings in total runtime is $\sim$2 times greater than the savings in CMFD runtime in the problem 5a-0, and it is nearly 9 times greater in the AP1000-EOC problem. These results reaffirm the need for an efficient CMFD solver in MPACT. Because the maximum number of PIs allowed per CMFD solver is 20, the default CMFD solver is not able to converge the CMFD problem sufficiently well. As a result, the overall transport scheme is not able to achieve the spectral radius predicted for the odCMFD method [5]. For the default CMFD solver, previous empirical studies determined that 20 was an appropriate limit on the number of PIs to optimize the tradeoff between time spent in the CMFD problem and time spent in the transport problem. With MED, this tradeoff does not play a role because we can "fully" converge[7] CMFD in

[7]The convergence criteria for CMFD, described in Section 3.2, is actually relatively "loose" compared to the convergence criteria used in the 1-D diffusion code. This is because CMFD is an accelerator, and its solution is only needed to speed up the transport calculation. CMFD only needs to be converged to the point at which we do not see any degradation in the convergence rate of transport sweeps. With the default CMFD solver, however, we usually unable to reach this point within 20 PIs.

an amount of time that is approximately an order of magnitude smaller than the total runtime of the problem.

The GD method also provides a significant speedup over the default and PI+PWS methods in three of the seven problems. However, it does not appear to scale well to larger problems (3-D full-core or 252-groups), and it is not as efficient or robust. For the five problems in which GD converges, MED provides a factor of $\sim$1.5-4.5x speedup over the GD method and has a smaller memory burden. Moreover, GD fails to converge for the 252-group problem 5a-2D and the AP1000-BOC problem. These failures are likely due to a combination of condition number growth and insufficient convergence of the inner loops of the algebraic multigrid preconditioner.

Compared to the other methods, RBBJ typically has a slightly reduced memory cost. The storage costs of its LU decompositions of the diffusion operator in each spatial cell is more than offset by the fact that it is implemented internally and does not require redundant storage in the form of PETSc matrices. However, it is not robust (it fails for problem 5a-0-FC), it does not scale well with problem size or G, and it takes significantly longer than the default method in most of the problems.

The results in this section indicate that, even without the multigrid-in-space linear solver described in the next chapter, the MED method already accomplishes the initial goal of reducing the computational burden of CMFD to $\sim$10% or less of the overall computational cost for most problems. The only exceptions are problems 5a-0-FC and AP1000-BOL, for which the CMFD runtime with MED still constitutes $\sim$18% of the overall runtime. We emphasize again that the benefits of using MED can go beyond the reduction in the CMFD runtime. MED allows us to fully converge the CMFD system in an efficient manner, allowing the outer CMFD-accelerated transport scheme to achieve the low spectral radius predicted by [5].

In the next chapter, we see that the introduction of the multigrid-in-space linear solver can further reduce the CMFD runtime. However, the marginal improvements provided by the multigrid-in-space linear solver are relatively small compared to the total cost of the MPACT method because of the speedups that have already been provided by the grey diffusion (multilevel-in-energy) component of MSED. That is, there is a limit to how much benefit we can extract from further optimizing the CMFD solver since it is only a small percentage of the overall runtime when MED is used. (For some of the smaller, non-full-core problems, MED can actually be faster than MSED.) Nonetheless, for 3-D, full-core problems (e.g., 5a-0-FC and the AP1000 problems), the speedup is still somewhat significant since the percentage of the total runtime devoted to solving CMFD with MED is higher. Moreover, the speedups provided by MSED over MED are important for multigroup diffusion codes in which the cost of solving the multigroup diffusion eigenvalue problem is not dwarfed by the presence of a transport system.

CHAPTER 7

Multigrid in Space

In this chapter, we discuss the second of the two major components of MSED – its multigrid-in-space linear solver. Mathematicians have used multigrid for decades, and the theory behind it, which is summarized in Section 2.2.3, is well-developed. However, the multigrid method described in this chapter features a novel treatment of the *multigroup* diffusion fixed-source problem. Fourier analyses are provided to assess the performance of the method as a standalone linear solver for multigroup fixed-source diffusion problems and its performance in the MSED framework described in Algorithm 8. This chapter includes numerical results from the 1-D diffusion code and MPACT that demonstrate the efficiency of the multigrid linear solver. In MPACT, we see that MSED can provide a speedup over the MED method introduced in the previous chapter when the problem is sufficiently large spatially. Compared to MED, MSED provides a $\sim$2x speedup in the CMFD runtime for full-core problems. Compared to the original CMFD solver in MPACT, the speedup in the CMFD runtime ranges from $\sim$6-12x.

7.1 Theory and Implementation

7.1.1 Theory

This subsection provides a brief summary of the theory covered in Section 2.2.3. There, all the variables and indices used to describe the general multigrid procedure are defined.

The purpose of the multigrid-in-space linear solver in MSED is to efficiently solved fixed-source diffusion problems in steps 2b and 4 of Algorithm 8. With a finite-difference spatial discretization in 1-D, this fixed-source problem can generally be written in the following form:

$$-\frac{1}{\Delta x_j}\left[D_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}-\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}} - D_{j-\frac{1}{2},g}\frac{\phi_{j,g}-\phi_{j-1,g}}{\Delta x_{j-\frac{1}{2}}}\right] + \Sigma_{t,j,g}\phi_{j,g} - \sum_{g'=1}^{G}\Sigma_{s0,j,g'\to g}\phi_{j,g'} = q_{j,g}\,. \tag{6.5 revisited}$$

If we let A be a matrix representing the operator on the left side of this equation, $\boldsymbol{b}$ be a vector whose components are $q_{j,g}$, and $\boldsymbol{x}$ be a vector whose components are $\phi_{j,g}$, we can use the procedure defined by Algorithm 2 to perform a multigrid V-cycle on this linear system. The coarse-grid operators $A_{(p)}$ are recursively defined using the Galerkin approach:

$$A_{(p)} = I_{(p+1)}^{(p)} A_{(p+1)} I_{(p)}^{(p+1)} . \tag{2.30 revisited}$$

Algorithm 2 and Equation (2.30) describe all the mechanics of multigrid except for two aspects: the smoother and the interpolation operators $I_{(p)}^{(p+1)}$. In both the 1-D diffusion code and MPACT, the restriction operator is simply defined to be the transpose of the interpolation operator:

$$I_{(p+1)}^{(p)} \equiv \left[I_{(p)}^{(p+1)} \right]^T . \tag{2.29 revisited}$$

Thus, to fully define our implementation of multigrid in the 1-D diffusion code and MPACT, we only need to define the smoother, and then either the interpolation or restriction operator. In the following subsections, we define these two aspects for both the 1-D diffusion code and MPACT.

As discussed in Section 2.2.3, the multigrid approach used in MSED combines notions of both algebraic and geometric multigrid. The Galerkin triple product in Equation (2.30) is a staple of algebraic multigrid methods, as it allows for the definition of coarse-grid operators without knowledge of the actual geometry. In our work, we use this Galerkin triple product, but we define all of the $I_{(p)}^{(p+1)}$ using information about the actual layout of the grid itself. This is possible for the CMFD problem in MPACT because of the simplicity of the CMFD grid. Although there may be mismatched boundaries between assemblies, the grid within each assembly is uniform and rectangular. More details regarding this are provided in Section 7.1.3.

7.1.2 Implementation in 1-D Diffusion Code

In the 1-D diffusion code, the interpolation operator is defined in a manner similar to the standard approach found in most multigrid texts (e.g., [9]) for the 1-D Laplace problem. If the one-group sweep approach (Algorithm 11) is used, the standard interpolation approach does not require modification. However, for step 2 of the reduced multigroup approach (Algorithm 10), the interpolation operator in MSED must consider and preserve the group-dependence of the scalar flux ϕ. The following definition is a natural extension of the standard multigrid interpolation approach to multigroup problems:

$$\phi_{(p),2j-1,g} = \phi_{(p-1),j,g} , \tag{7.1a}$$

$$\phi_{(p),2j,g} = \frac{1}{2} \left[\phi_{(p-1),j,g} + \phi_{(p-1),j+1,g} \right] . \tag{7.1b}$$

Here, $\phi_{(p),j,g}$ is the group g scalar flux in the j-th spatial cell of spatial grid p. (We defined p in Section 2.2.3 so that $p = 0$ is the coarsest grid and $p = P - 1$ is the original grid.) Equation (7.1a) indicates that half the points ($\phi_{(p),2j-1,g}$) will be obtained via injection of values directly from the coarse grid, while Equation (7.1b) indicates that the remaining half ($\phi_{(p),2j,g}$) will be obtained by averaging the two closest coarse grid values. From these two equations, the interpolation matrix $I_{(p-1)}^{(p)}$ can be defined.

Although the original fluxes $\phi_{j,g}$ are cell-averaged quantities, it is possible (and optimal [113, 114]) to treat them as if they were nodal quantities because of the finite-difference spatial discretization. That is, we are using the fact that Equation (6.5) would look the same if ϕ represented nodal quantities rather than cell-averaged quantities. We note that interpolation and restriction in multigrid acts on the residuals ($\boldsymbol{r}$) and errors ($\boldsymbol{\varepsilon}$) rather than the fluxes ($\boldsymbol{\phi}$), but it is easier to formulate Equations (7.1) in terms of the fluxes.

A visualization of the interpolation is shown in Figure 7.1. When the number of fine-grid points is even, there will be an extra node on the right side of the fine grid. In this thesis, our standard treatment of this node is to simply include it in the coarse grid. In Figure 7.1, this would result in an extra vertical arrow (indicating an injection from the coarse grid to fine grid) on the right side. Using a spatial collapse in which the leftmost and rightmost nodes from the fine grid are present in all of the coarse grids allows us to avoid special treatment for the boundary conditions.

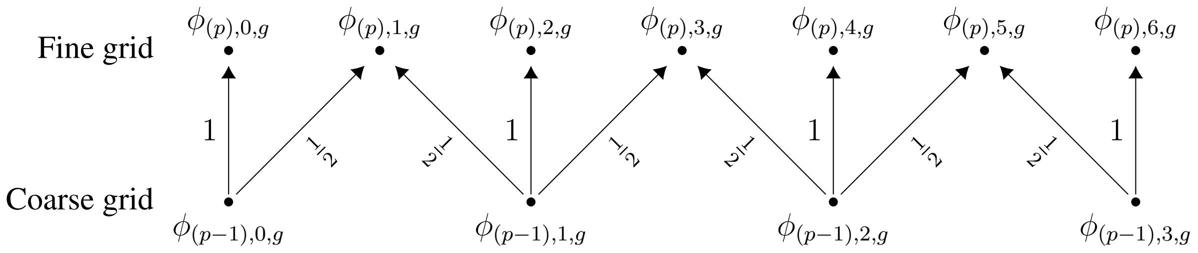


Figure 7.1: Interpolation operator for a multigroup fixed-source diffusion problem with finite-difference spatial discretization in 1-D. This is a visualization of Equations (7.1). The numbers on the arrows correspond to the coarse-grid weights used to define fine-grid quantities.

In the work in this thesis, we do not encounter problems for which we have to interpolate across nonuniform spatial cells or problems with problematic (highly discontinuous) diffusion coefficients. Nonetheless, we now describe straightforward approaches for dealing with these two situations, should they arise:

- In the case of interpolation or restriction across nonuniform spatial cells, one can define $\phi_{(p),2j,g}$ in Equation (7.1b) by linearly interpolating between the values at $\phi_{(p-1),j,g}$ and $\phi_{(p-1),j+1,g}$ [9]. For a uniform grid, $\phi_{(p),2j,g}$ is halfway between these two values; for a nonuniform grid, the weights will depend on the relative spatial cell sizes.

- For situations in which the convergence of multigrid is degraded due to highly discontinuous diffusion coefficients, we refer the reader to the work performed by Alcouffe in [34]. There, Alcouffe develops an interpolation scheme that is robust for problems in which the diffusion coefficient can vary by several orders of magnitude between spatial cells. To account for the impact of these sharp variations, Alcouffe treats the diffusion coefficients as scalings of the spatial cell Δx. The final result is an interpolation operator that treats $D\nabla\phi$ as if it were a linear function of space. We note that, in [34], the interpolation is only done for one-group problems. When this approach is applied to a multigroup problem, one can either use the one-group sweep approach described in Algorithm 11 or use group-dependent interpolation coefficients to collapse all groups simultaneously.

Using the Galerkin triple product (Equation (2.30)) and the interpolation operator defined above (Equations (7.1)), a formulation for the coarse-grid operator can be derived. For simplicity, we consider a homogeneous problem on a uniform grid. The purpose of this derivation is to provide the reader with a sense of what the coarse-grid operator looks like. In practice, we form the problem, interpolation, and restriction matrices and use Equation (2.30) to compute the coarse-grid operators directly.

First, we substitute Equations (7.1) into Equation (6.5) for spatial cells $2j$, $2j-1$, and $2j-2$:

$$-\frac{D_g}{\Delta x^2}\left[\left(\phi_{(P-2),j+1,g}-\frac{\phi_{(P-2),j,g}+\phi_{(P-2),j+1,g}}{2}\right)-\left(\frac{\phi_{(P-2),j,g}+\phi_{(P-2),j+1,g}}{2}-\phi_{(P-2),j,g}\right)\right]$$
$$+\Sigma_{t,g}\frac{\phi_{(P-2),j,g}+\phi_{(P-2),j+1,g}}{2}-\sum_{g'=1}^{G}\Sigma_{s0,g'\to g}\frac{\phi_{(P-2),j,g'}+\phi_{(P-2),j+1,g'}}{2}=q_{2j,g}\ , \quad (7.2a)$$

$$-\frac{D_g}{\Delta x^2}\left[\left(\frac{\phi_{(P-2),j,g}+\phi_{(P-2),j+1,g}}{2}-\phi_{(P-2),j,g}\right)-\left(\phi_{(P-2),j,g}-\frac{\phi_{(P-2),j-1,g}+\phi_{(P-2),j,g}}{2}\right)\right]$$
$$+\Sigma_{t,g}\phi_{(P-2),j,g}-\sum_{g'=1}^{G}\Sigma_{s0,g'\to g}\phi_{(P-2),j,g'}=q_{2j-1,g}\ , \quad (7.2b)$$

$$-\frac{D_g}{\Delta x^2}\left[\left(\phi_{(P-2),j,g}-\frac{\phi_{(P-2),j-1,g}+\phi_{(P-2),j,g}}{2}\right)-\left(\frac{\phi_{(P-2),j-1,g}+\phi_{(P-2),j,g}}{2}-\phi_{(P-2),j-1,g}\right)\right]$$
$$+\Sigma_{t,g}\frac{\phi_{(P-2),j-1,g}+\phi_{(P-2),j,g}}{2}-\sum_{g'=1}^{G}\Sigma_{s0,g'\to g}\frac{\phi_{(P-2),j-1,g'}+\phi_{(P-2),j,g'}}{2}=q_{2j-2,g}\ . \quad (7.2c)$$

We note that $(P-1)$ refers to the finest grid, so $(P-2)$ is the coarse grid immediately below it. The leakage term cancels for the even spatial cells, and the leakage term for the odd spatial cell can be rearranged. Some simplification yields

$$\Sigma_{t,g} \frac{\phi_{(P-2),j,g} + \phi_{(P-2),j+1,g}}{2} - \sum_{g'=1}^{G} \Sigma_{s0,g'\to g} \frac{\phi_{(P-2),j,g'} + \phi_{(P-2),j+1,g'}}{2} = q_{2j,g} \,, \tag{7.3a}$$

$$\begin{aligned} &- \frac{D_g}{(2\Delta x)^2} \left[2\phi_{(P-2),j+1,g} - 4\phi_{(P-2),j,g} + 2\phi_{(P-2),j-1,g}\right] \\ &\qquad + \Sigma_{t,g}\phi_{(P-2),j,g} - \sum_{g'=1}^{G} \Sigma_{s0,g'\to g}\phi_{(P-2),j,g'} = q_{2j-1,g} \,, \end{aligned} \tag{7.3b}$$

$$\Sigma_{t,g} \frac{\phi_{(P-2),j-1,g} + \phi_{(P-2),j,g}}{2} - \sum_{g'=1}^{G} \Sigma_{s0,g'\to g} \frac{\phi_{(P-2),j-1,g'} + \phi_{(P-2),j,g'}}{2} = q_{2j-2,g} \,. \tag{7.3c}$$

Finally, we weight the three equations by 1/4, 1/2, and 1/4, respectively, and sum them together. This is equivalent to applying the transpose of the interpolation operator as the restriction operator. (We have multiplied the weights of the restriction by $1/2$ to produce a coarse-grid operator that is more visually consistent with the fine-grid operator; the scaling of the weights does not affect Algorithm 2 since both the operator and the residual are restricted using the same weights.) The result is the following:

$$\begin{aligned} &- \frac{D_g}{(2\Delta x)^2} \left[\phi_{(P-2),j+1,g} - 2\phi_{(P-2),j,g} + \phi_{(P-2),j-1,g}\right] \\ &\quad + \frac{1}{8}\Sigma_{t,g} \left[\phi_{(P-2),j-1,g} + \phi_{(P-2),j+1,g}\right] - \frac{1}{8} \sum_{g'=1}^{G} \Sigma_{s0,g'\to g} \left[\phi_{(P-2),j+1,g'} + \phi_{(P-2),j-1,g}\right] \\ &\qquad + \frac{3}{4}\Sigma_{t,g}\phi_{(P-2),j,g} - \frac{3}{4} \sum_{g'=1}^{G} \Sigma_{s0,g'\to g}\phi_{(P-2),j,g'} = q_{(P-2),j,g} \,. \end{aligned} \tag{7.4}$$

The following conclusions are apparent from this result:

1. If $\Sigma_{t,g} = 0$ and $\Sigma_{s0,g'\to g} = 0$, the result is the same as the fine-grid operator, except for a grid of size $2\Delta x$.

2. The contribution from the removal terms "bleed" into the leakage terms as we coarsen.

The first conclusion implies that the Galerkin approach (Equation (2.30)) and the purely geometric multigrid approach (homogenizing the cross-sections and directly discretizing the diffusion operator on the coarse grid) generate the same coarse-grid operators when there are no removal terms. The second conclusion means that the contributions from neighboring cells are generally not diagonal in

g in the coarse-grid operators. This is a drawback of the Galerkin approach, as it reduces the sparsity of the coarse-grid operators. Nonetheless, the Galerkin approach is taken in our implementations because of its robustness. As discussed in Section 2.2.3 and [15], multigrid methods that use Galerkin coarse-grid operators satisfy a "variational principle" that ensures convergence as long as the smoother converges. In future work, alternative geometric approaches may be considered to reduce the computational and memory costs of computing coarse-grid operators. However, because the aforementioned variational principle may not hold, such approaches should be carefully analyzed to ensure stability.

Finally, we discuss the smoothing mechanism. For the 1-D diffusion code, we use the "red-black" block Jacobi (RBBJ) smoother described in Algorithm 12. The treatment of the energy-dependence is "exact" in the sense that the $G \times G$ scattering blocks are solved for each spatial cell. This "exact" treatment minimizes the spectral radius of the multigrid V-cycle applied to Equation (6.5), but its computational complexity does not scale well with G. (Solving a $G \times G$ linear system is $O(G^3)$.) Moreover, if LU factorization is used to avoid repeated applications of Gaussian elimination to the $G \times G$ blocks, the LU factorization of each $G \times G$ block must be stored, significantly increasing the memory cost of the code. This cost is not prohibitive for a simple 1-D diffusion test code, but it is not ideal for MPACT. In the next subsection, we describe a different approach with a less stringent treatment of the energy-dependence. This approach has computational complexity $O(G^2)$ rather than $O(G^3)$ and performs sufficiently well for MPACT.

Algorithm 12: A "red-black" block Jacobi smoother iteration for Equation (6.5), used in the 1-D diffusion code. This smoother is meant to be used in the multigrid V-cycle described by Algorithm 2.

Input: $\phi_{j,g}^{(q)}$

Result: $\phi_{j,g}^{(q+\frac{1}{2})}$

1. Smooth on the odd values of j (or the even values if using 0-based indexing). In the context of multigrid, this includes all of the grid points that are members of both the current fine grid ($P-1$) and the next coarse grid ($P-2$), except possibly the rightmost point.

$$\left[\frac{D_{j+\frac{1}{2},g}}{\Delta x_j \Delta x_{j+\frac{1}{2}}} + \frac{D_{j-\frac{1}{2},g}}{\Delta x_j \Delta x_{j-\frac{1}{2}}} + \Sigma_{t,j,g}\right]\phi_{j,g}^{(q+\frac{1}{2})} - \sum_{g'=1}^{G}\Sigma_{s0,j,g'\to g}\phi_{j,g'}^{(q+\frac{1}{2})} = q_{j,g} + \frac{1}{\Delta x_j}\left[\frac{D_{j+\frac{1}{2},g}}{\Delta x_{j+\frac{1}{2}}}\phi_{j+1,g}^{(q)} + \frac{D_{j-\frac{1}{2},g}}{\Delta x_{j-\frac{1}{2}}}\phi_{j-1,g}^{(q)}\right]. \quad \text{(Alg12.1)}$$

2. Smooth on the remaining values of j.

$$\left[\frac{D_{j+\frac{1}{2},g}}{\Delta x_j \Delta x_{j+\frac{1}{2}}} + \frac{D_{j-\frac{1}{2},g}}{\Delta x_j \Delta x_{j-\frac{1}{2}}} + \Sigma_{t,j,g}\right]\phi_{j,g}^{(q+\frac{1}{2})} - \sum_{g'=1}^{G}\Sigma_{s0,j,g'\to g}\phi_{j,g'}^{(q+\frac{1}{2})} = q_{j,g} + \frac{1}{\Delta x_j}\left[\frac{D_{j+\frac{1}{2},g}}{\Delta x_{j+\frac{1}{2}}}\phi_{j+1,g}^{(q+\frac{1}{2})} + \frac{D_{j-\frac{1}{2},g}}{\Delta x_{j-\frac{1}{2}}}\phi_{j-1,g}^{(q+\frac{1}{2})}\right]. \quad \text{(Alg12.2)}$$

Here, all values of $\phi^{(q+\frac{1}{2})}$ on the right side have already been computed in step 1.

7.1.3 Implementation in MPACT

Because of higher dimensionality and parallelism, the interpolation operator used for the 1-D diffusion code cannot be applied to the MPACT code directly. We first discuss the treatment of higher dimensions. The approach taken in MPACT is a natural extension of the approach described for 1-D problems. For a 2-D problem, Figure 2.2 demonstrates how a fine grid can be coarsened to a coarse grid by keeping only the points whose x and y indices are both odd. When the number of rows (or columns) is not an odd number, one can keep an additional row (or column) when restricting to simplify the treatment of boundary conditions. This is demonstrated in Figure 7.2. As in the 1-D case, keeping this additional row/column simplifies the treatment of boundary conditions.

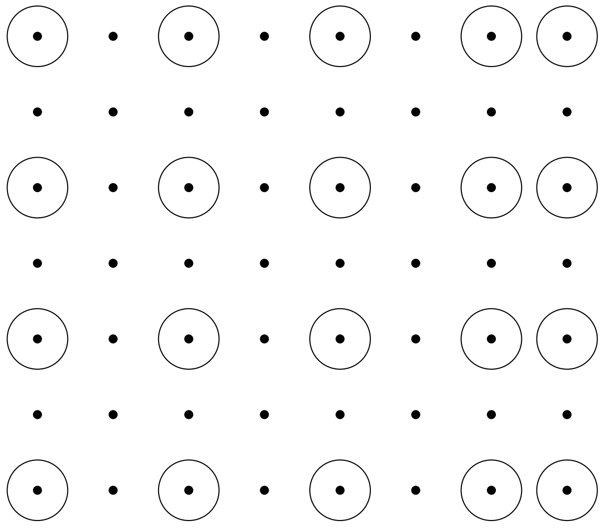


Figure 7.2: A sample coarsening for a 2-D Cartesian grid with an even number of columns. Dots represent nodes on the fine grid, while circles represent nodes on the coarse grid. This is the same as Figure 2.2 except for the presence of the additional column of nodes on the right.

With the coarse grid defined in Figure 7.2 (or Figure 2.2), the interpolation operator for the 2-D scalar fluxes $\phi_{i,j,g}$ can be defined as follows:

$$\phi_{(p),2i-1,2j-1,g} = \phi_{(p-1),i,j,g} , \tag{7.5a}$$

$$\phi_{(p),2i-1,2j,g} = \frac{1}{2}\left[\phi_{(p-1),i,j,g} + \phi_{(p-1),i,j+1,g}\right] , \tag{7.5b}$$

$$\phi_{(p),2i,2j-1,g} = \frac{1}{2}\left[\phi_{(p-1),i,j,g} + \phi_{(p-1),i+1,j,g}\right] , \tag{7.5c}$$

$$\phi_{(p),2i,2j,g} = \frac{1}{4}\left[\phi_{(p),2i-1,2j,g} + \phi_{(p),2i+1,2j,g} + \phi_{(p),2i,2j-1,g} + \phi_{(p),2i,2j+1,g}\right] . \tag{7.5d}$$

The approach is the same as that of the interpolation scheme in 1-D; all of the fine-grid points are defined by linearly interpolating from the nearest coarse-grid points. As in 1-D, nonuniform interpolation weights can be used when highly discontinuous diffusion coefficients or nonuniform meshes are present.

To handle parallelism, we treat parallel boundaries in the same manner that we treat domain boundaries. This results in an interpolation operator that does not need to communicate across processors; it is similar to the one described by [28]. Since restriction is just the transpose of interpolation in our implementation, it also does not require any inter-processor communication. A visualization of this approach is shown in Figure 7.3 for a 2-D spatial domain split over four processors. Presently, all problems of interest in MPACT are partitioned such that there is one spatial plane per processor. As a result, no three-dimensional interpolation is needed in MPACT.

Nonetheless, a suitable interpolation operator can be defined in three-dimensions by the same approach used to define the two-dimensional interpolation operator.

As noted in the previous subsection, a smoother that scales better with G than RBBJ is sought. We have found that the successive over-relaxation (SOR) smoother from PETSc, with no relaxation, performs adequately for our multigrid implementation. It is effectively a Gauss-Seidel (GS) iteration on the problem matrix, except that information from neighboring processors is lagged. In standard GS, the operation on row i of the matrix is done with updated information from rows $i' < i$. However, the unknowns corresponding to these rows may reside on other processors, and processors do not communicate until a full smoothing step has been completed. This smoother is described in detail in Algorithm 13. Because the treatment of the energy-dependence is no longer "exact" in the smoother, the bulk of the convergence of the energy-dependence is left to the solver on the coarsest grid. The GS smoothing steps help with the energy-dependence, but it is relatively slow since the convergence rate of GS is typically about equal to the scattering ratio. Although the SOR smoother from PETSc has performed well in our current implementation of multigrid in MPACT, we acknowledge that it is likely possible to improve upon it in future work.

Algorithm 13: A GS smoother iteration for Equation (6.5); this is the smoother that is used in MPACT. This smoother is meant to be used in the multigrid V-cycle described by Algorithm 2.

Input: $\phi_{j,g}^{(q)}$

Result: $\phi_{j,g}^{(q+\frac{1}{2})}$

1. Smooth on the odd values of j (or the even values if using 0-based indexing). In the context of multigrid, this includes all of the grid points that are members of both the current grid (p) and the next coarse grid ($p-1$), except possibly the rightmost point.

$$\left[\frac{D_{j+\frac{1}{2},g}}{\Delta x_j \Delta x_{j+\frac{1}{2}}} + \frac{D_{j-\frac{1}{2},g}}{\Delta x_j \Delta x_{j-\frac{1}{2}}} + \Sigma_{t,j,g} - \Sigma_{s0,j,g\to g}\right] \phi_{j,g}^{(q+\frac{1}{2})} = q_{j,g} + \sum_{g'=1}^{g-1} \Sigma_{s0,j,g'\to g}\phi_{j,g'}^{(q+\frac{1}{2})} + \sum_{g'=g+1}^{G} \Sigma_{s0,j,g'\to g}\phi_{j,g'}^{(q)} + \frac{1}{\Delta x_j}\left[\frac{D_{j+\frac{1}{2},g}}{\Delta x_{j+\frac{1}{2}}}\phi_{j+1,g}^{(q)} + \frac{D_{j-\frac{1}{2},g}}{\Delta x_{j-\frac{1}{2}}}\phi_{j-1,g}^{(q[+\frac{1}{2}])}\right]. \quad \text{(Alg13.1)}$$

   In the second leakage term on the right side, the iteration index is $(q+\frac{1}{2})$ if $j-1$ lives on the same processor as j, and it is (q) otherwise.

2. Communicate across processors where necessary to update information about neighboring values of $\phi_{j,g}$.

Several additional differences exist between the implementations of multigrid in the 1-D diffusion code and in MPACT. First, in MPACT, a smoother iteration is performed between interpolation

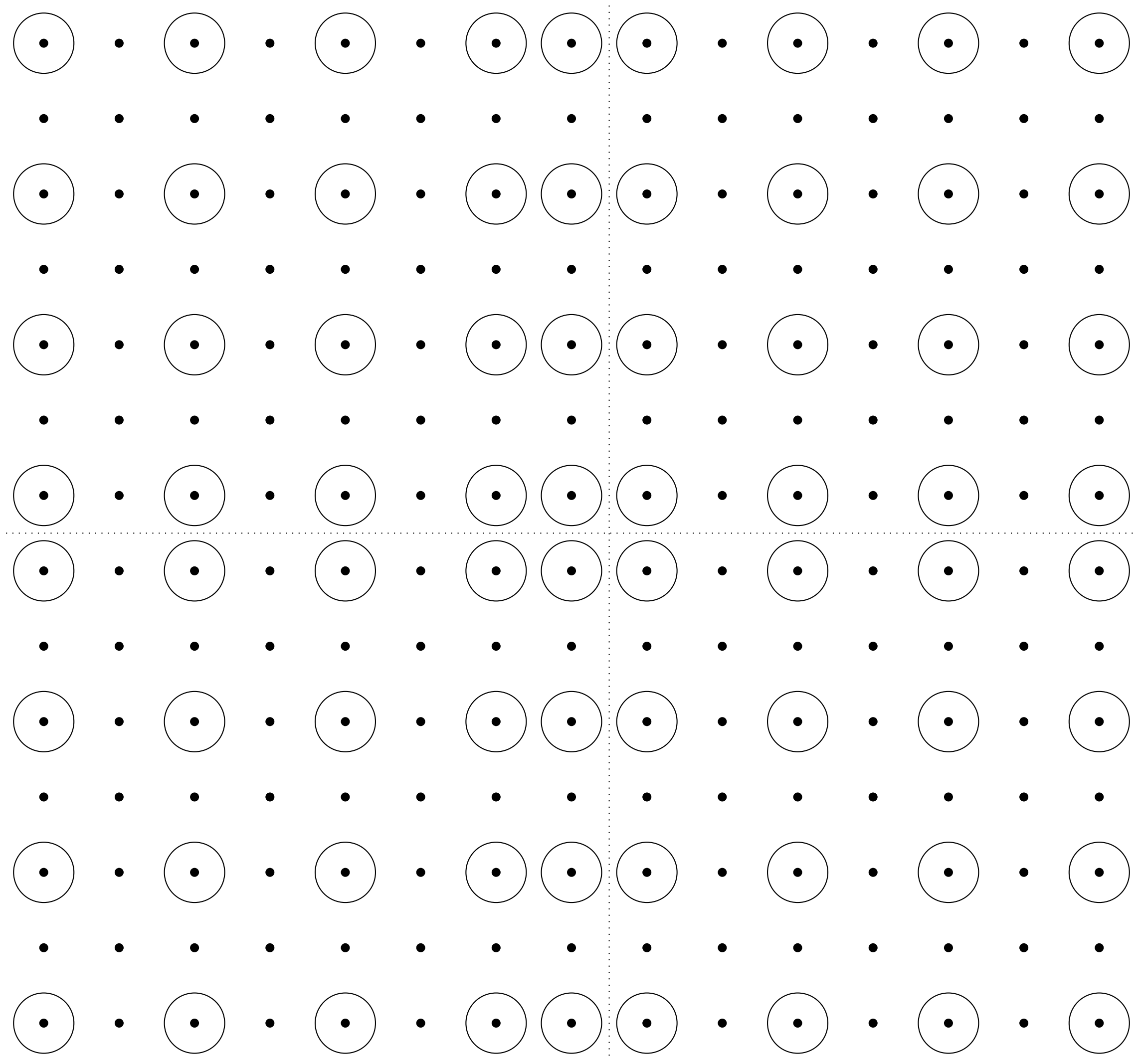


Figure 7.3: A sample coarsening for a 2-D Cartesian grid decomposed on four different processors. Dots represent nodes on the fine grid, circles represent nodes on the coarse grid, and parallel boundaries are marked with dotted lines. With this definition for the coarse grid, no inter-processor communication is needed for the interpolation step.

steps as well as between restriction steps. In the 1-D diffusion code, smoothing is only performed between restriction steps. We have found that this extra smoothing step helps to mitigate the penalty in the convergence rate of multigrid incurred from using GS instead of RBBJ.

Second, the number of spatial grids is limited in MPACT. Because the 1-D diffusion code does not have parallel decomposition, we can easily coarsen to an arbitrarily small spatial grid. In MPACT, however, it is not possible to restrict to a grid with fewer unknowns than processors without communication between processors. The primary reason we coarsen beyond two grids in multigrid is to arrive at a grid for which a direct solve is more computationally feasible. (The spectral radius for a multigrid method is minimized with two grids, but this would require solving the grid $P - 2$ error problem exactly, which is generally not feasible except for small problems.) For parallel codes, the ratio of communication costs to computational costs increases as the number of unknowns per processor decreases. Thus, there is a limit to how much one can decrease the cost of solving the problem on the coarsest grid. In order to preserve the communication-avoiding nature of the interpolation and restriction operators, our implementation in MPACT stops coarsening when it reaches a spatial grid for which a processor has four unknowns.

Third, in MPACT, 15 GMRES iterations are used instead of a direct solve on the coarsest grid. With four or more unknowns per processor and hundreds or thousands of processors, the number of unknowns on the coarsest grid is still large enough that a direct solve is not computationally efficient. From our numerical tests, we have not seen a significant improvement in the convergence rate of multigrid from using a direct solver (e.g., SuperLU [115]) instead of GMRES on the coarsest grid. (Although the number of multigrid iterations required can be reduced slightly, there is a significant slowdown because of the large computational cost of using a direct solver on the coarsest-grid solve.) Although we have found 15 GMRES iterations to be sufficient for our problems of interest, we acknowledge that there may be problems for which 15 iterations is insufficient. For scaling to more than $O(1000)$ processors, it may be necessary to communicate between processors to restrict to a coarsest grid with even fewer unknowns. Moreover, for problems in which the coarse-grid operator is nearly singular (e.g., transient problems or multigroup problems with WS), it may not be possible to avoid using a direct solver.

Finally, we note that, when using our multigrid method in higher dimensions, we face the additional problem of "stencil growth." Whereas the original fine-grid operator in CMFD is a 7-point stencil (the equation for cell (i, j, k) has terms from cells $(i \pm 1, j, k)$, $(i, j \pm 1, k)$, and $(i, j, k \pm 1)$, the coarse-grid operators have stencils that may be as large as 27 points due to the presence of terms from cells $(i \pm 1, j \pm 1, k \pm 1)$. The increase in the stencil size results in significant increases in the computational and memory burdens for computing the coarse-grid operators. This is a common problem for algebraic multigrid methods [116]; in purely algebraic multigrid methods, coarse-grid operators must often be truncated to prevent prohibitive setup costs. Fortunately, the stencil in our multigrid method (which is not purely algebraic) does not grow beyond 27 points. (In most problems, the average stencil size is about $\sim$22 on the coarse grids.) As noted in the previous subsection, we use the Galerkin approach despite these drawbacks for its robustness and guaranteed stability (as long as the smoother is stable). Geometric alternatives to the Galerkin approach may be considered in the future, but their stability properties would have to be carefully analyzed.

7.2 Fourier Analysis

In this section, we perform Fourier analyses of two processes: (1) solving a fixed-source multigroup diffusion problem using multigrid, and (2) an MSED iteration using the multigrid solver. In the latter case, the number of V-cycles on the multigroup linear system (Q) is allowed to vary, so that we can determine the impact of this parameter on the spectral radius of the overall MSED algorithm.

7.2.1 Multigrid on Fixed-Source Diffusion Problems

7.2.1.1 Algorithm

The goal of this section is study the impact of a multigrid V-cycle on the linear system

$$-\frac{D_g}{\Delta x^2}\left[\phi_{i+1,g} - 2\phi_{i,g} + \phi_{i-1,g}\right] + \sum_{g'=1}^{G} \Sigma_{r,g'\to g}\phi_{i,g'} = \chi_g \,. \tag{7.6}$$

Here, $\Sigma_{r,g'\to g}$ is the removal cross section, defined by

$$\Sigma_{r,g'\to g} = \begin{cases} \Sigma_{t,g} - \Sigma_{s0,g\to g}\,, & \text{if } g' = g\,, \\ -\Sigma_{s0,g'\to g}\,, & \text{otherwise}\,. \end{cases} \tag{7.7}$$

As in previous Fourier analyses, this is a 1-D problem on a homogeneous domain of size X with periodic boundary conditions, divided into J spatial cells of size Δx. The energy-dependent, space-independent source is χ_g.

For this problem, a two-grid V-cycle with an RBBJ smoother is defined by the following equations:

$$\frac{2D_g}{\Delta x^2}\phi_{2j-1,g}^{(q+\frac{1}{2})} + \sum_{g'=1}^{G}\Sigma_{r,g'\to g}\phi_{2j-1,g'}^{(q+\frac{1}{2})} = \chi_g + \frac{D_g}{\Delta x^2}\left[\phi_{2j,g}^{(q)} + \phi_{2j-2,g}^{(q)}\right]\,, \tag{7.8a}$$

$$\frac{2D_g}{\Delta x^2}\phi_{2j,g}^{(q+\frac{1}{2})} + \sum_{g'=1}^{G}\Sigma_{r,g'\to g}\phi_{2j,g'}^{(q+\frac{1}{2})} = \chi_g + \frac{D_g}{\Delta x^2}\left[\phi_{2j+1,g}^{(q+\frac{1}{2})} + \phi_{2j-1,g}^{(q+\frac{1}{2})}\right]\,, \tag{7.8b}$$

$$r^{(q+\frac{1}{2})}_{(1),2j,g} = 0\,, \tag{7.9a}$$

$$\begin{aligned}r^{(q+\frac{1}{2})}_{(1),2j-1,g} &= \frac{D_g}{\Delta x^2}\left[\phi^{(q+\frac{1}{2})}_{2j,g} + \phi^{(q+\frac{1}{2})}_{2j-2,g}\right] + \chi_g - \left[\frac{2D_g}{\Delta x^2} + \Sigma_{t,g}\right]\phi^{(q+\frac{1}{2})}_{2j-1,g} - \sum_{g'=1}^{G}\Sigma_{r,g'\to g}\phi^{(q+\frac{1}{2})}_{2j-1,g'} \\ &= \frac{D_g}{\Delta x^2}\left[\phi^{(q+\frac{1}{2})}_{2j,g} + \phi^{(q+\frac{1}{2})}_{2j-2,g}\right] - \frac{D_g}{\Delta x^2}\left[\phi^{(q)}_{2j,g} + \phi^{(q)}_{2j-2,g}\right]\,, \end{aligned}\tag{7.9b}$$

$$r^{(q)}_{(0),j,g} = \frac{1}{4}\left[r^{(q+\frac{1}{2})}_{(1),2j-2,g} + r^{(q+\frac{1}{2})}_{(1),2j-1,g} + r^{(q+\frac{1}{2})}_{(1),2j,g}\right] = \frac{1}{2}r^{(q+\frac{1}{2})}_{(1),2j-1,g}\,, \tag{7.10}$$

$$\begin{aligned}&-\frac{D_g}{(2\Delta x)^2}\left[\varepsilon^{(q+1)}_{(0),j+1,g} - 2\varepsilon^{(q+1)}_{(0),j,g} + \varepsilon^{(q+1)}_{(0),j-1,g}\right] + \frac{1}{8}\sum_{g'=1}^{G}\Sigma_{r,g'\to g}\left[\varepsilon^{(q+1)}_{(0),j+1,g} + \varepsilon^{(q+1)}_{(0),j-1,g}\right] \\ &\qquad + \frac{3}{4}\sum_{g'=1}^{G}\Sigma_{r,g'\to g}\varepsilon^{(q+1)}_{(0),j,g'} = r^{(q)}_{(0),i}\,,\end{aligned}\tag{7.11}$$

$$\varepsilon^{(q+1)}_{(1),2j,g} = \frac{1}{2}\left[\varepsilon^{(q+1)}_{(0),j,g} + \varepsilon^{(q+1)}_{(0),j+1,g}\right]\,, \tag{7.12}$$

$$\phi^{(q+1)}_{2j,g} = \phi^{(q+\frac{1}{2})}_{2j,g} + \varepsilon^{(q+1)}_{(1),2j,g}\,. \tag{7.13}$$

Here, q is used as the iteration index. Equations (7.8) describe a single RBBJ smoother iteration. The residual after this iteration is computed in Equations (7.9), and it is restricted to the coarse grid ($p = 0$) in Equation (7.10). A coarse-grid solve is performed in Equation (7.11); the resulting error is interpolated in Equation (7.12) and used to correct the fine-grid scalar flux in Equation (7.13). We note that computation of the odd-indexed scalar fluxes at superscripts (q) or $(q + 1)$ is not necessary because of the red-black nature of the smoother.

7.2.1.2 Analysis

We begin with the following Fourier ansatz:

$$\phi^{(q)}_{2j,g} = \phi_{0,g} + \epsilon a_g \omega^q e^{i\xi(2j)\Delta x}\,, \tag{7.14a}$$

$$\phi^{(q+\frac{1}{2})}_{2j,g} = \phi_{0,g} + \epsilon \alpha_{e,g}\omega^q e^{i\xi(2j)\Delta x}\,, \tag{7.14b}$$

$$\phi^{(q+\frac{1}{2})}_{2j-1,g} = \phi_{0,g} + \epsilon \alpha_{o,g}\omega^q e^{i\xi(2j-1)\Delta x}\,, \tag{7.14c}$$

$$\varepsilon^{(q+1)}_{(0),j,g} = \epsilon b_g \omega^q e^{i\xi(2j-1)\Delta x}\,. \tag{7.14d}$$

Here, $\phi_{0,g}$ is the solution to Equation (7.6). i is the imaginary number $\sqrt{-1}$, j is the spatial index, ω is the decay factor, and ξ is the spatial frequency of the error mode. To ensure periodicity and to avoid aliased frequencies, ξ should be chosen as follows:

$$\xi \in \left\{ \frac{2\pi k}{X} \middle| k \in \mathbb{Z}, 0 \leq k \leq \left\lfloor \frac{J}{4} \right\rfloor \right\} . \tag{7.15}$$

Because of the odd-even split in the RBBJ smoother, error modes cannot have a frequency greater than $\frac{\pi}{4X}$. (In Section 6.2, the maximum permitted frequency was $\frac{\pi}{2X}$.)

First, we consider the smoothing step. When we substitute the Fourier ansatz into Equations (7.8), all the terms of $O(1)$ cancel from the definition of $\phi_{0,g}$. Dividing the resulting equations by $\epsilon\omega^q e^{i\xi(2j-1)\Delta x}$ and $\epsilon\omega^q e^{i\xi(2j)\Delta x}$ respectively yields

$$\frac{2D_g}{\Delta x^2}\alpha_{o,g} + \sum_{g'=1}^{G} \Sigma_{r,g'\to g}\alpha_{o,g'} = \frac{2D_g}{\Delta x^2} a_g \cos(\xi \Delta x) \,, \tag{7.16a}$$

$$\frac{2D_g}{\Delta x^2}\alpha_{e,g} + \sum_{g'=1}^{G} \Sigma_{r,g'\to g}\alpha_{e,g'} = \frac{2D_g}{\Delta x^2} \alpha_{o,g} \cos(\xi \Delta x) \,. \tag{7.16b}$$

The right sides of these two equations have been simplified using the identity

$$e^{i\xi\Delta x} + e^{-i\xi\Delta x} = 2\cos(\xi\Delta x) \,. \tag{7.17}$$

After dividing both sides by $\frac{2D_g}{\Delta x^2}\cos(\xi\Delta x)$, Equations (7.16) can be succinctly expressed in matrix notation as

$$R_1 \boldsymbol{\alpha}_o = \boldsymbol{a} \,, \tag{7.18a}$$

$$R_1 \boldsymbol{\alpha}_e = \boldsymbol{\alpha}_o \,. \tag{7.18b}$$

Combing these two equations yields

$$\boxed{\boldsymbol{\alpha}_e = (R_1)^{-2} \boldsymbol{a}} \,. \tag{7.19}$$

Here, R_1 is a $G \times G$ matrix, and $\boldsymbol{\alpha}_e$, $\boldsymbol{\alpha}_o$, and $\boldsymbol{a}$ are length-G vectors whose terms consist of $\alpha_{e,g}$, $\alpha_{o,g}$, and a_g, respectively.

After the smoothing step, we consider the computation of the residual and its subsequent restriction. Substituting the ansatz into Equations (7.9) and (7.10) yields

$$r^{(q)}_{(0),j,g} = \epsilon\omega^q \frac{D_g}{\Delta x^2} e^{i\xi(2j-1)\Delta x} \cos(\xi\Delta x) \left[\alpha_{e,g} - a_g\right] \,. \tag{7.20}$$

We note that the terms of $O(1)$ cancel in Equation (7.9b). Next, we insert this result and the Fourier ansatz into the coarse grid equation (Equation (7.11)) and divide both sides of the equation by $\epsilon\omega^q\xi e^{i\xi(2j-1)\Delta x}$ to obtain

$$\frac{D_g}{(2\Delta x)^2}\left[2-e^{-2i\xi\Delta x}-e^{2i\xi\Delta x}\right]b_g+\frac{\cos(2\xi\Delta x)}{4}\sum_{g'=1}^{G}\Sigma_{r,g'\to g}b_{g'} + \frac{3}{4}\sum_{g'=1}^{G}\Sigma_{r,g'\to g}b_{g'} = \frac{D_g}{\Delta x^2}\cos(\xi\Delta x)\left[\alpha_{e,g}-a_g\right] . \quad (7.21)$$

Then, we use the equality

$$2-e^{-2i\xi\Delta x}-e^{2i\xi\Delta x}=4\sin^2(\xi\Delta x) \quad (7.22)$$

and rearrange some of the terms in Equation (7.21) to obtain the following result:

$$\frac{D_g}{\Delta x^2}\sin^2(\xi\Delta x)b_g+\left[\frac{\cos(2\xi\Delta x)}{4}+\frac{3}{4}\right]\sum_{g'=1}^{G}\Sigma_{r,g'\to g}b_{g'} = \frac{D_g}{\Delta x^2}\cos(\xi\Delta x)\left[\alpha_{e,g}-a_g\right] . \quad (7.23)$$

In matrix notation, the above equation can be expressed as

$$R_2\boldsymbol{b}=(\boldsymbol{\alpha}_e-\boldsymbol{a}) . \quad (7.24)$$

We substitute in our earlier result in Equation (7.19) to obtain:

$$\boxed{R_2\boldsymbol{b}=\left((R_1)^{-2}-I\right)\boldsymbol{a}} . \quad (7.25)$$

Here, R_2 is a $G\times G$ matrix, and I is the identity matrix of size $G\times G$.

Finally, we consider the interpolation of the coarse-grid error and subsequent fine-grid error correction. Substituting the ansatz into Equations (7.12) and (7.13) yields

$$\boxed{\omega a_g=\alpha_g+\cos(\xi\Delta x)b_g} . \quad (7.26)$$

Combining Equations (7.19), (7.25) and (7.26) results in

$$\boxed{\omega\boldsymbol{a}=M\boldsymbol{a}} , \quad (7.27)$$

where

$$\boxed{M\equiv(R_1)^{-2}+\cos(\xi\Delta x)\,(R_2)^{-1}\left(R_1^{-2}-I\right)} . \quad (7.28)$$

Finally, to obtain the decay factors ρ, we can solve the eigenvalue problem defined by Equation (7.27) for the decay factors ω, and find the maximum value of $|\omega|$ over all error frequencies ξ.

7.2.1.3 Analytical Results for Grey Fixed-Source Problems

In the grey setting, the matrices R_1 and R_2 are scalars, defined as follows:

$$R_1 = \frac{1}{\cos(\xi\Delta x)}\left[1 + \frac{\Sigma_a \Delta x^2}{2D}\right] , \tag{7.29a}$$

$$R_2 = \sin(\xi\Delta x)\tan(\xi\Delta x) + \frac{\Sigma_a \Delta x^2}{4D\cos(\xi\Delta x)}\left[3 + \cos(2\xi\Delta x)\right] . \tag{7.29b}$$

Moreover, we have $\omega = M$. The above expressions depend on two dimensionless quantities:

$$\theta_1 \equiv \xi\Delta x \, , \tag{7.30a}$$

$$\theta_2 = \frac{\Sigma_a \Delta x^2}{2D} \, . \tag{7.30b}$$

Using θ_1 and θ_2, R_1 and R_2 can be re-expressed as

$$R_1 = \frac{1+\theta_2}{\cos\theta_1} , \tag{7.31a}$$

$$R_2 = \sin\theta_1 \tan\theta_1 + \frac{\theta_2}{2\cos\theta_1}\left[3 + \cos(2\theta_1)\right] . \tag{7.31b}$$

Substituting these two results into Equation (7.28), we obtain

$$\omega = \left(\frac{\cos\theta_1}{1+\theta_2}\right)^2 + \frac{\cos(\theta_1)}{\sin\theta_1\tan\theta_1 + \frac{\theta_2}{2\cos\theta_1}\left[3+\cos(2\theta_1)\right]}\left[\left(\frac{\cos\theta_1}{1+\theta_2}\right)^2 - 1\right] . \tag{7.32}$$

θ_2 can physically vary from 0 to ∞, while θ_1 is bounded between 0 and $\frac{\pi}{2}$ by Equation (7.15). From the above expression, we can see that $\omega = M \to 0$ when $\theta_2 \to 0$ or $\theta_2 \to \infty$. A numerical maximization of $|\omega|$ over the physically relevant values of θ_1 and θ_2 gives a value of

$$|\omega|_{max} \approx 0.125 \tag{7.33}$$

at $\theta_1 = 0$ and $\theta_2 = 1$. This tells us that the spectral radius for the multigrid-in-space linear solver is very low for grey 1-D problems, regardless of the problem parameters. For reference, a spectral radius of 0.125 means that a problem will converge to a tolerance of 10^{-8} in just 9 iterations.

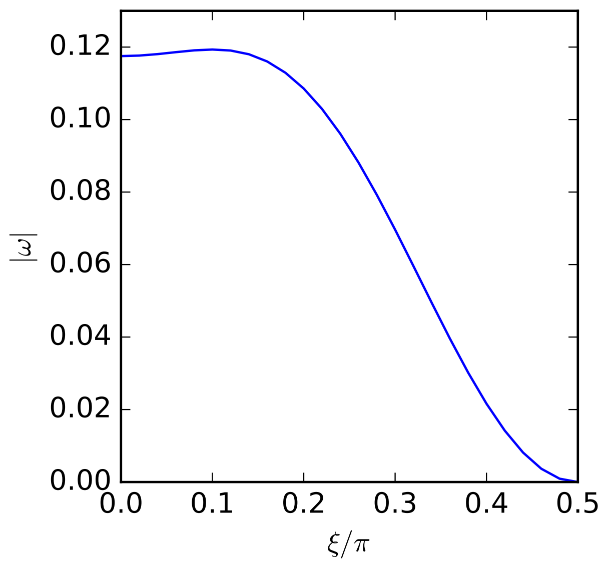
7.2.1.4 Numerical Results for Multigroup Problems

In this subsection, we solve for $|\omega|$ in Equation (7.27) using the same four different cross section sets considered in the Fourier analysis in Section 6.2. $|\omega|$ is plotted as a function of $\theta_1 = \xi \Delta x$ in Figures 7.4 for a problem with $\Delta x = 1$ cm and $X = 100$ cm. The spectral radius ρ is the maximum value of $|\omega|$ over all values of ξ; this is reported in Table 7.1. Figure 7.5 is provided to show that the spectral radius does not depend on Δx.

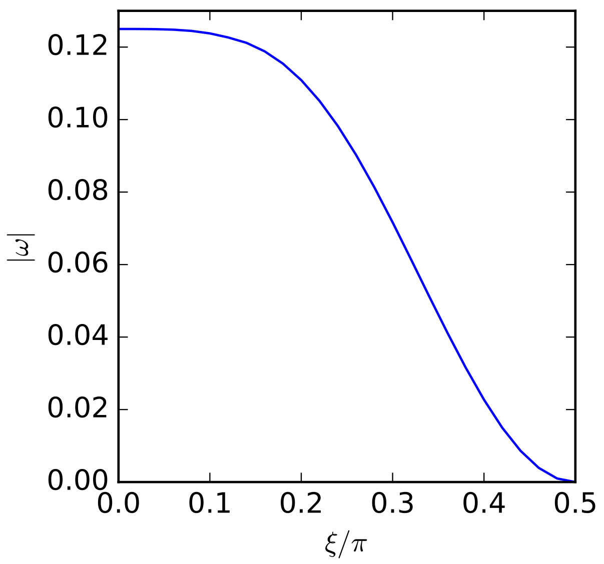
Table 7.1: Spectral radii computed for a Fourier analysis of multigrid with an RBBJ smoother on a fixed-source problem with $\Delta x = 1$ cm and $X = 100$ cm. These spectral radii are computed by taking the maximum of the decay factors ω over ξ. These decay factors are shown in Figure 7.4.

Library	ρ
MPACT 8G	0.119
MPACT 47G	0.125
MPACT 51G	0.125
MPACT 252G	0.125

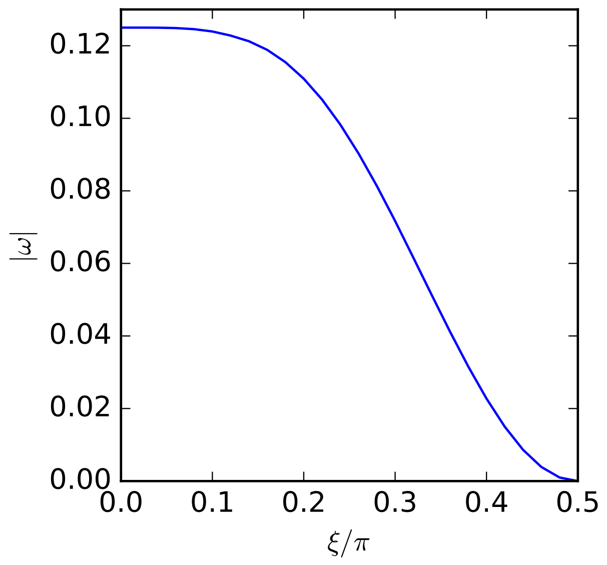
For all of the cross-section libraries, the resulting spectral radius is approximately 0.125, and this value is attained at or near $\xi = 0$. These results indicate that the convergence of the multigrid method is essentially the same for both multigroup and grey fixed-source diffusion problems. Moreover, the spectral radius appears to be insensitive to the energy-dependence of the problem, as ρ is bounded by 0.125 for any of the four libraries we considered. Although the analysis performed in this section is for an idealized problem (homogeneous, periodic boundary conditions, uniform grid, $P = 2$), we have observed convergence rates similar to 0.125 in practice. From our experience, the only situation in which the multigrid method with the RBBJ smoother fails to converge is when the diffusion operator is supercritical (i.e., when an overly aggressive WS is applied). This is consistent with the previously discussed variational property of multigrid methods that use the Galerkin approach – such methods are stable as long as the smoother is stable.



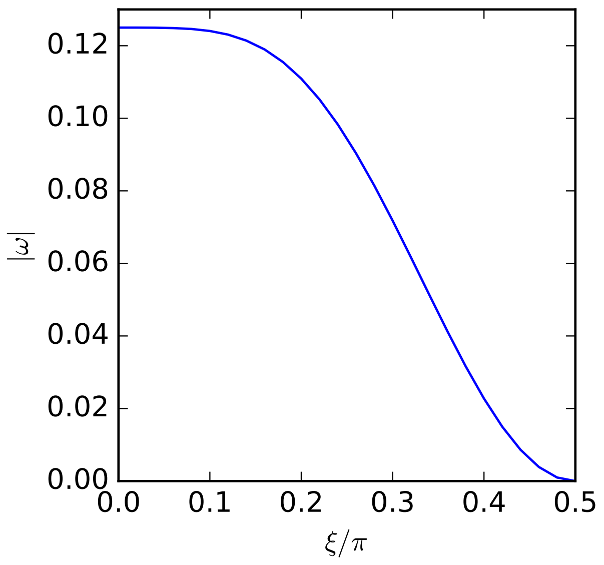
(a) MPACT 8G library.



(b) MPACT 47G library.



(c) MPACT 51G library.



(d) MPACT 252G library.

Figure 7.4: Decay factors from Fourier analysis of a two-grid multigrid scheme with an RBBJ smoother on a fixed-source problem with $\Delta x = 1$ cm and $X = 100$ cm for four different cross section sets. Cross sections are obtained by a homogenization of the center pin-cell in problem 2a using four different MPACT libraries. The spectral radius is given by the maximum value of $|\omega|$ over all values of ξ.

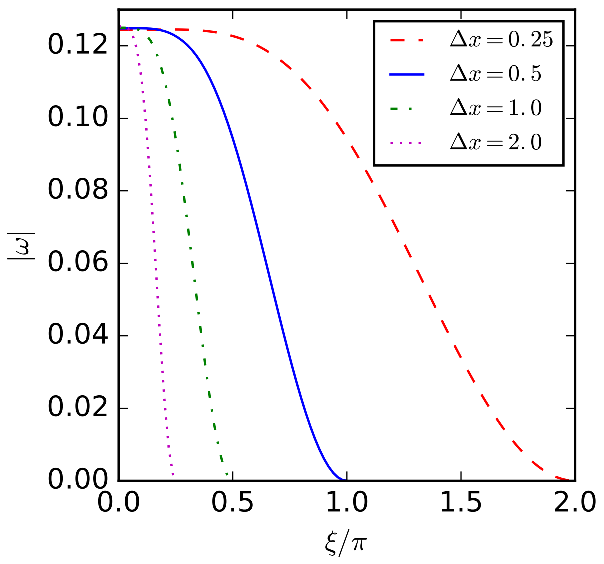


Figure 7.5: Decay factors from Fourier analysis of multigrid with an RBBJ smoother on a fixed-source problem for different Δx and $X = 100$ cm. Here, the case using the MPACT 51G library (Figure 7.4c) is shown. The domain of ξ is truncated by the rule in Equation (7.15).

7.2.2 MSED

7.2.2.1 Algorithm

The goal of this section is to study the impact of multigrid V-cycles on an MSED iteration. The Fourier analysis here is the same as the Fourier analysis of the MED method in Section 6.2 (with $C = 1$) except in the work done on the multigroup equation to obtain $\phi^{(l+1)}$ from $\phi^{(l+\frac{1}{2})}$. In Section 6.2, the multigroup problem is solved exactly to produce $\phi^{(l+1)}$. Here, we perform Q multigrid V-cycles on the multigroup system to obtain $\phi^{(l+1)}$.

The problem we wish to solve is

$$\frac{D_g}{\Delta x^2}\left[-\phi_{j+1,g} + 2\phi_{j,g} - \phi_{j-1,g}\right] + \Sigma_{t,g}\phi_{j,g} - \sum_{g'=1}^{G} \Sigma_{s0,g'\to g}\phi_{g'} = \lambda\chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'}\phi_{j,g'}\,, \tag{7.34a}$$

subject to the normalization condition

$$\frac{1}{J}\sum_{j=1}^{J}\sum_{g}\phi_{j,g} = 1\,. \tag{7.34b}$$

As in Section 6.2, the domain is a 1-D homogeneous medium of width X with periodic boundary conditions, divided into J spatial cells of size Δx.

For this problem, an MSED iteration is defined by the following equations:

$$\Phi_j^{(l)} \equiv \sum_{g=1}^{G} \phi_{j,g}^{(l)}\,, \tag{7.35a}$$

$$\langle \Sigma_{a,j} \rangle^{(l)} \equiv \frac{1}{\Phi_j^{(l)}} \left[\sum_{g=1}^{G} \Sigma_{t,g} \phi_{j,g}^{(l)} - \sum_{g'=1}^{G} \sum_{g=1}^{G} \Sigma_{s0,g'\to g} \phi_{j,g'}^{(l)} \right] = \frac{1}{\Phi_j^{(l)}} \sum_{g=1}^{G} \Sigma_{a,g} \phi_{j,g}^{(l)}\,, \tag{7.35b}$$

$$\langle \nu\Sigma_{f,j} \rangle^{(l)} \equiv \frac{1}{\Phi_j^{(l)}} \sum_{g=1}^{G} \nu\Sigma_{f,g} \phi_{j,g}^{(l)}\,, \tag{7.35c}$$

$$\langle D_j \rangle^{(l)} \equiv \frac{1}{\Phi_j^{(l)}} \sum_{g=1}^{G} D_g \phi_{j,g}^{(l)}\,, \tag{7.35d}$$

$$-\frac{\langle D_{j+1} \rangle^{(l)} \Phi_{j+1}^{(l+\frac{1}{2})} + \langle D_{j-1} \rangle^{(l)} \Phi_{j-1}^{(l+\frac{1}{2})}}{\Delta x^2} + \left[2 \langle D_j \rangle^{(l)} + \langle \Sigma_{a,j} \rangle^{(l)} \right] \Phi_j^{(l+\frac{1}{2})} = \lambda^{(l+\frac{1}{2})} \langle \nu\Sigma_{f,j} \rangle^{(l)} \Phi_j^{(l+\frac{1}{2})}\,, \tag{7.36a}$$

$$\frac{1}{J} \sum_{j=1}^{J} \Phi_j^{(l+\frac{1}{2})} = 1\,, \tag{7.36b}$$

$$\phi_{j,g}^{(l+\frac{1}{2},0)} = \frac{\Phi_j^{(l+\frac{1}{2})}}{\Phi_j^{(l)}} \phi_j^{(l)}\,, \tag{7.37}$$

$$\frac{2D_g}{\Delta x^2} \phi_{2j-1,g}^{(l+\frac{1}{2},q+\frac{1}{2})} + \sum_{g'=1}^{G} \Sigma_{r,g'\to g} \phi_{2j-1,g'}^{(l+\frac{1}{2},q+\frac{1}{2})} = \lambda^{(l+\frac{1}{2})} \chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'} \phi_{2j-1,g'}^{(l+\frac{1}{2},0)} + \frac{D_g}{\Delta x^2} \left[\phi_{2j,g}^{(l+\frac{1}{2},q)} + \phi_{2j-2,g}^{(l+\frac{1}{2},q)} \right]\,, \tag{7.38a}$$

$$\frac{2D_g}{\Delta x^2} \phi_{2j,g}^{(l+\frac{1}{2},q+\frac{1}{2})} + \sum_{g'=1}^{G} \Sigma_{r,g'\to g} \phi_{2j,g'}^{(l+\frac{1}{2},q+\frac{1}{2})} = \lambda^{(l+\frac{1}{2})} \chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'} \phi_{2j,g'}^{(l+\frac{1}{2},0)} + \frac{D_g}{\Delta x^2} \left[\phi_{2j+1,g}^{(l+\frac{1}{2},q+\frac{1}{2})} + \phi_{2j-1,g}^{(l+\frac{1}{2},q+\frac{1}{2})} \right]\,, \tag{7.38b}$$

$$r_{(1),2j,g}^{(l+\frac{1}{2},q+\frac{1}{2})} = 0\,, \tag{7.39a}$$

$$\begin{aligned}
r_{(1),2j-1,g}^{(l+\frac{1}{2},q+\frac{1}{2})} &= \frac{D_g}{\Delta x^2}\left[\phi_{2j,g}^{(l+\frac{1}{2},q+\frac{1}{2})} + \phi_{2j-2,g}^{(l+\frac{1}{2},q+\frac{1}{2})}\right] + \lambda^{(l+\frac{1}{2})}\chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'}\phi_{2j-1,g'}^{(l+\frac{1}{2},0)} \\
&\quad - \left[\frac{2D_g}{\Delta x^2} + \Sigma_{t,g}\right]\phi_{2j-1,g}^{(l+\frac{1}{2},q+\frac{1}{2})} - \sum_{g'=1}^{G}\Sigma_{r,g'\to g}\phi_{2j-1,g'}^{(l+\frac{1}{2},q+\frac{1}{2})} \\
&= \frac{D_g}{\Delta x^2}\left[\phi_{2j,g}^{(l+\frac{1}{2},q+\frac{1}{2})} + \phi_{2j-2,g}^{(l+\frac{1}{2},q+\frac{1}{2})}\right] - \frac{D_g}{\Delta x^2}\left[\phi_{2j,g}^{(l+\frac{1}{2},q)} + \phi_{2j-2,g}^{(l+\frac{1}{2},q)}\right]\,,
\end{aligned} \tag{7.39b}$$

$$r_{(0),j,g}^{(l+\frac{1}{2},q)} = \frac{1}{4}\left[r_{(1),2j-2,g}^{(l+\frac{1}{2},q+\frac{1}{2})} + r_{(1),2j-1,g}^{(l+\frac{1}{2},q+\frac{1}{2})} + r_{(1),2j,g}^{(l+\frac{1}{2},q+\frac{1}{2})}\right] = \frac{1}{2}r_{(1),2j-1,g}^{(l+\frac{1}{2},q+\frac{1}{2})}\,, \tag{7.40}$$

$$\begin{aligned}
-\frac{D_g}{(2\Delta x)^2}\left[\varepsilon_{(0),j+1,g}^{(l+\frac{1}{2},q+1)} - 2\varepsilon_{(0),j,g}^{(l+\frac{1}{2},q+1)} + \varepsilon_{(0),j-1,g}^{(l+\frac{1}{2},q+1)}\right] + \frac{1}{8}\sum_{g'=1}^{G}\Sigma_{r,g'\to g}\left[\varepsilon_{(0),j+1,g}^{(l+\frac{1}{2},q+1)} + \varepsilon_{(0),j-1,g}^{(l+\frac{1}{2},q+1)}\right] \\
+ \frac{3}{4}\sum_{g'=1}^{G}\Sigma_{r,g'\to g}\varepsilon_{(0),j,g'}^{(l+\frac{1}{2},q+1)} = r_{(0),i}^{(l+\frac{1}{2},q)}\,,
\end{aligned} \tag{7.41}$$

$$\varepsilon_{(1),2j,g}^{(l+\frac{1}{2},q+1)} = \frac{1}{2}\left[\varepsilon_{(0),j,g}^{(l+\frac{1}{2},q+1)} + \varepsilon_{(0),j+1,g}^{(l+\frac{1}{2},q+1)}\right]\,, \tag{7.42}$$

$$\phi_{2j,g}^{(l+\frac{1}{2},q+1)} = \phi_{2j,g}^{(l+\frac{1}{2},q+\frac{1}{2})} + \varepsilon_{(1),2j,g}^{(l+\frac{1}{2},q+1)}\,. \tag{7.43}$$

$$\lambda^{(l+1)} = \lambda^{(l+\frac{1}{2})}\left[\sum_{j=1}^{J}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}\phi_{g'}^{(l+\frac{1}{2},Q)}\right]^{-1}\sum_{j=1}^{J}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}\phi_{j,g'}^{(l+\frac{1}{2},0)}\,, \tag{7.44a}$$

$$\phi_{j,g}^{(l+1)} = \left[\frac{1}{J}\sum_{j=1}^{J}\sum_{g'=1}^{G}\phi_{j,g'}^{(l+\frac{1}{2},Q)}\right]^{-1}\phi_{j,g}^{(l+\frac{1}{2},Q)}\,. \tag{7.44b}$$

The notation here is consistent with the notation used in the Fourier analyses of Sections 6.2 and 7.2.1. The only new variable is Q, which represents the number of multigrid V-cycles performed on the multigroup system per MSED iteration. In the following analysis, we will allow Q to vary and assess its impact on the predicted spectral radius of the MSED method. When $Q \to \infty$, we recover the spectral radii computed in Section 6.2.

7.2.2.2 Analysis

We begin this analysis with the following Fourier ansatz:

$$\phi_{2j,g}^{(l)} = \varphi_g + \epsilon a_{e,g}\omega^l e^{i\xi(2j)\Delta x}\,, \tag{7.45a}$$

$$\phi_{2j-1,g}^{(l)} = \varphi_g + \epsilon a_{o,g}\omega^l e^{i\xi(2j-1)\Delta x}\,, \tag{7.45b}$$

$$\Phi_{2j}^{(l)} = 1 + \epsilon A_e\omega^l e^{i\xi(2j)\Delta x}\,, \tag{7.45c}$$

$$\Phi_{2j-1}^{(l)} = 1 + \epsilon A_o\omega^l e^{i\xi(2j-1)\Delta x}\,, \tag{7.45d}$$

$$\lambda^{(l)} = \lambda_0 + \epsilon\lambda_1\omega^l\,, \tag{7.45e}$$

$$\Phi_{2j}^{(l+\frac{1}{2})} = 1 + \epsilon \mathcal{A}_e\omega^l e^{i\xi(2j)\Delta x}\,, \tag{7.45f}$$

$$\Phi_{2j-1}^{(l+\frac{1}{2})} = 1 + \epsilon \mathcal{A}_o\omega^l e^{i\xi(2j-1)\Delta x}\,, \tag{7.45g}$$

$$\phi_{2j,g}^{(l+\frac{1}{2},q)} = \varphi_g + \epsilon \alpha_{e,q,g}\omega^l e^{i\xi(2j)\Delta x}\,, \tag{7.45h}$$

$$\phi_{2j-1,g}^{(l+\frac{1}{2},q)} = \varphi_g + \epsilon \alpha_{o,q,g}\omega^l e^{i\xi(2j-1)\Delta x}\,, \tag{7.45i}$$

$$\varepsilon_{(0),j,g}^{(l+\frac{1}{2},q+1)} = \epsilon b_{q+1,g}\omega^l e^{i\xi(2j-1)\Delta x}\,, \tag{7.45j}$$

$$\lambda^{(l+\frac{1}{2})} = \lambda_0 + \epsilon\lambda_1'\omega^l\,. \tag{7.45k}$$

Again, i is the imaginary number $\sqrt{-1}$, j is the spatial index, ω is the decay factor, and ξ is the spatial frequency of the error mode. As in Section 6.2, φ_g and λ_0 are the solutions to the infinite-medium multigroup diffusion eigenvalue problem; φ_g is defined to satisfy the normalization condition

$$\sum_{g=1}^{G} \varphi_g = 1\,. \tag{7.46}$$

The values of ξ are restricted to the following set to ensure periodicity and avoid aliasing:

$$\xi \in \left\{ \frac{2\pi k}{X} \middle| k \in \mathbb{Z}, 0 \le k \le \left\lfloor \frac{J}{4} \right\rfloor \right\}\,. \tag{7.47}$$

By the same reasoning used in the Fourier analysis in Section 6.2, we have the following results:

$$\lambda_1 = 0\,, \tag{7.48}$$

$$\lambda_1' = 0\,, \tag{7.49}$$

$$\phi_{j,g}^{(l+1)} = \phi_{j,g}^{(l+\frac{1}{2},Q)}\,, \tag{7.50}$$

$$\omega a_{e|o,g} = \alpha_{e|o,Q,g}\,, \tag{7.51}$$

$$A_{e|o} = \sum_{g=1}^{G} a_{e|o,g} \,. \tag{7.52}$$

Here and throughout this analysis, the symbol | is used to reduce repetition. In Equation (7.51), $e|o$ indicates that the equation is valid with either the left value (e) or the right value (o). That is, Equation (7.51) implies that the following two equations hold:

$$a_{e,g} = \alpha_{e,Q,g} \,, \tag{7.53a}$$

$$a_{o,g} = \alpha_{o,Q,g} \,. \tag{7.53b}$$

Next, we substitute the Fourier ansatz into Equations (7.35) to obtain expressions for the grey cross-sections. The derivations are nearly the same as those in Section 6.2, and the details are omitted for brevity.

$$\left\langle D_{2j|2j-1} \right\rangle^{(l)} = \overline{D} + \epsilon\omega^l e^{i\xi(2j|2j-1)\Delta x} \left(-\overline{D} A_{e|o} + \sum_{g=1}^{G} \Sigma_g a_{e|o,g} \right) \,, \tag{7.54a}$$

$$\left\langle \nu\Sigma_{a,2j|2j-1} \right\rangle^{(l)} = \overline{\nu\Sigma}_a + \epsilon\omega^l e^{i\xi(2j|2j-1)\Delta x} \left(-\overline{\nu\Sigma}_f A_{e|o} + \sum_{g=1}^{G} \Sigma_g a_{e|o,g} \right) \,, \tag{7.54b}$$

$$\left\langle \nu\Sigma_{f,2j|2j-1} \right\rangle^{(l)} = \overline{\nu\Sigma}_f + \epsilon\omega^l e^{i\xi(2j|2j-1)\Delta x} \left(-\overline{\nu\Sigma}_f A_{e|o} + \sum_{g=1}^{G} \Sigma_g a_{e|o,g} \right) \,. \tag{7.54c}$$

For a cross section or diffusion coefficient Σ, $\overline{\Sigma}$ denotes its corresponding grey quantity, obtained by flux-weighting with φ_g. For example,

$$\overline{\Sigma}_a \equiv \sum_{g=1}^{G} \Sigma_{a,g} \varphi_g \,. \tag{7.55}$$

We now substitute Equations (7.54) and the Fourier ansatz into Equation (7.36a):

$$\begin{aligned}
&-\frac{1}{\Delta x^2}\left\{\overline{D}+\epsilon\omega^l e^{i\xi(2j-1|2j-2)\Delta x}\left(-\overline{D}A_{o|e}+\sum_{g=1}^{G}D_g a_{o|e,g}\right)\right\}\left[1+\epsilon\omega^l\mathcal{A}_{o|e}e^{i\xi(2j-1|2j-2)\Delta x}\right]\\
&\quad+\frac{2}{\Delta x^2}\left\{\overline{D}+\epsilon\omega^l e^{i\xi(2j|2j-1)\Delta x}\left(-\overline{D}A_{e|o}+\sum_{g=1}^{G}D_g a_{e|o,g}\right)\right\}\left[1+\epsilon\omega^l\mathcal{A}_{e|o}e^{i\xi(2j|2j-1)\Delta x}\right]\\
&\quad-\frac{1}{\Delta x^2}\left\{\overline{D}+\epsilon\omega^l e^{i\xi(2j+1|2j)\Delta x}\left(-\overline{D}A_{o|e}+\sum_{g=1}^{G}D_g a_{o|e,g}\right)\right\}\left[1+\epsilon\omega^l\mathcal{A}_{o|e}e^{i\xi(2j+1|2j)\Delta x}\right]\\
&\quad+\left\{\overline{\Sigma}_a+\epsilon\omega^l e^{i\xi(2j|2j-1)\Delta x}\left(-\overline{\Sigma}_a A_{e|o}+\sum_{g=1}^{G}\Sigma_{a,g}a_{e|o,g}\right)\right\}\left[1+\epsilon\mathcal{A}_{e|o}\omega^l e^{i\xi(2j|2j-1)\Delta x}\right]\\
&=\lambda_0\left\{\overline{\nu\Sigma}_f+\epsilon\omega^l e^{i\xi(2j|2j-1)\Delta x}\left(\overline{\nu\Sigma}_f A_{e|o}+\sum_{g=1}^{G}\nu\Sigma_{f,g}a_{e|o,g}\right)\right\}\left[1+\epsilon\mathcal{A}_{e|o}\omega^l e^{i\xi(2j|2j-1)\Delta x}\right]\,.
\end{aligned} \tag{7.56}$$

All of the $O(1)$ terms cancel by definition of φ_g and λ_0, and we drop any remaining $O(\epsilon^2)$ terms. After dividing the result by $\epsilon\omega^l e^{i\xi(2j|2j-1)\Delta x}$, we obtain

$$\begin{aligned}
&-\frac{2}{\Delta x^2}\cos(\xi\Delta x)\left[\overline{D}\left(\mathcal{A}_{o|e}-A_{o|e}\right)+\sum_{g=1}^{G}D_g a_{o|e,g}\right]+\frac{2}{\Delta x^2}\left[\overline{D}\left(\mathcal{A}_{e|o}-A_{e|o}\right)+\sum_{g=1}^{G}D_g a_{e|o,g}\right]\\
&\quad+\overline{\Sigma}_a\left(\mathcal{A}_{e|o}-A_{e|o}\right)+\sum_{g=1}^{G}\Sigma_{a,g}a_{e|o,g}=\lambda_0\overline{\nu\Sigma}_f\left(\mathcal{A}_{e|o}-A_{e|o}\right)+\lambda_0\sum_{g=1}^{G}\nu\Sigma_{f,g}a_{e|o,g}\,.
\end{aligned} \tag{7.57}$$

From the definition of φ_g, we can derive the identity

$$\overline{\Sigma}_a=\lambda_0\overline{\nu\Sigma}_f\,, \tag{7.58}$$

which is applied to Equation (7.57) to obtain

$$\begin{aligned}
&-\frac{2}{\Delta x^2}\cos(\xi\Delta x)\left[\overline{D}\left(\mathcal{A}_{o|e}-A_{o|e}\right)+\sum_{g=1}^{G}D_g a_{o|e,g}\right]+\frac{2}{\Delta x^2}\left[\overline{D}\left(\mathcal{A}_{e|o}-A_{e|o}\right)+\sum_{g=1}^{G}D_g a_{e|o,g}\right]\\
&\quad+\sum_{g=1}^{G}\Sigma_{a,g}a_{e|o,g}=\lambda_0\sum_{g=1}^{G}\nu\Sigma_{f,g}a_{e|o,g}\,.
\end{aligned} \tag{7.59}$$

With a bit of rearrangement, we can express the above equation in matrix form as follows:

$$\boxed{H_1 \begin{bmatrix} \mathcal{A}_e - A_e \\ \mathcal{A}_o - A_o \end{bmatrix} = H_2 \begin{bmatrix} \boldsymbol{a}_e \\ \boldsymbol{a}_o \end{bmatrix}}\,. \tag{7.60}$$

Here, H_1 is a 2×2 matrix, H_2 is a $2 \times 2G$ matrix, and $\boldsymbol{a}_e$ and $\boldsymbol{a}_o$ are length-G column vectors.

Next, we substitute the Fourier ansatz into the grey-to-multigroup interpolation step (Equation (7.37)) to obtain an expression for $\alpha_{e|o,0,g}$ in terms of $\mathcal{A}_{e|o}$ and $a_{e|o,g}$. The derivation is identical to the derivation in Equation (6.31), and the final result is given by

$$\boxed{\alpha_{e|o,0,g} = \varphi_g \left(\mathcal{A}_{e|o} - A_{e|o}\right) + a_{e|o,g}}\,. \tag{7.61}$$

The remainder of this derivation focuses on the Q multigrid V-cycles performed on the multigroup system. This is similar to the derivation performed in Section 7.2.1; the only difference is that the source has an extra $O(\epsilon)$ term from the fission source. First, we consider the RBBJ smoothing step. Substituting the Fourier ansatz into Equations (7.38) yields

$$\frac{2D_g}{\Delta x^2}\alpha_{o,q+\frac{1}{2},g} + \sum_{g'=1}^{G} \Sigma_{r,g'\to g}\alpha_{o,q+\frac{1}{2},g'} = \lambda_0 \chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'}\alpha_{o,0,g'} + \frac{2D_g}{\Delta x^2}\alpha_{e,q,g}\cos(\xi\Delta x)\,, \tag{7.62a}$$

$$\frac{2D_g}{\Delta x^2}\alpha_{e,q+\frac{1}{2},g} + \sum_{g'=1}^{G} \Sigma_{r,g'\to g}\alpha_{e,q+\frac{1}{2},g'} = \lambda_0 \chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'}\alpha_{e,0,g'} + \frac{2D_g}{\Delta x^2}\alpha_{o,q+\frac{1}{2},g}\cos(\xi\Delta x)\,. \tag{7.62b}$$

As usual, the $O(1)$ terms have cancelled out, and we have divided the equations by $\epsilon\omega^l e^{i\xi(2j-1)\Delta x}$ and $\epsilon\omega^l e^{i\xi(2j)\Delta x}$, respectively. In matrix notation, the above equations can be expressed as:

$$H_3\boldsymbol{\alpha}_{o,q+\frac{1}{2}} = F\boldsymbol{\alpha}_{o,0} + H_4\boldsymbol{\alpha}_{e,q}\,, \tag{7.63a}$$

$$\begin{aligned} H_3\boldsymbol{\alpha}_{e,q+\frac{1}{2}} &= F\boldsymbol{\alpha}_{e,0} + H_4\boldsymbol{\alpha}_{o,q+\frac{1}{2}} \\ &= F\boldsymbol{\alpha}_{e,0} + H_4H_3^{-1}F\boldsymbol{\alpha}_{o,0} + H_4H_3^{-1}H_4\boldsymbol{\alpha}_{e,q}\,. \end{aligned} \tag{7.63b}$$

H_3 is a $G \times G$ matrix, H_4 is a diagonal $G \times G$ matrix, F is a rank-1 $G \times G$ matrix, and the $\boldsymbol{\alpha}_{e|o}$ are all length-G column vectors.

The residual, restriction, coarse-grid solve, interpolation, and fine-grid correction steps are identical to those analyzed in Section 7.2.1, and they result in the following expressions:

$$R_2\boldsymbol{b}_{q+1} = \boldsymbol{\alpha}_{e,q+\frac{1}{2}} - \boldsymbol{\alpha}_{e,q}\,, \tag{7.64}$$

$$\boldsymbol{\alpha}_{e,q+1} = \boldsymbol{\alpha}_{e,q+\frac{1}{2}} + \cos(\xi\Delta x)\boldsymbol{b}_{q+1}\,, \tag{7.65}$$

$$\boldsymbol{\alpha}_{o,q+1} = \boldsymbol{\alpha}_{o,q+\frac{1}{2}} + \boldsymbol{b}_{q+1}\,. \tag{7.66}$$

These first two equations are analogous to Equations (7.24) and Equations (7.26), and the $G \times G$ matrix R_2 has the same definition here as it does in Equation (7.24). Combining Equations (7.63), (7.64), and (7.65) allows us to express $\alpha_{e,q+1,g}$ in terms of $\alpha_{e,q,g}$ and $\alpha_{e|o,0,g}$:

$$\begin{aligned}
\boldsymbol{\alpha}_{e,q+1} &= \boldsymbol{\alpha}_{e,q+\frac{1}{2}} + \cos(\xi\Delta x)\boldsymbol{b}_{q+1} \\
&= \left[\cos(\xi\Delta x)R_2^{-1} + I\right]\boldsymbol{\alpha}_{e,q+\frac{1}{2}} - \cos(\xi\Delta x)R_2^{-1}\boldsymbol{\alpha}_{e,q} \\
&= \left[\cos(\xi\Delta x)R_2^{-1} + I\right]H_3^{-1}\left\{F\boldsymbol{\alpha}_{e,0} + H_4H_3^{-1}F\boldsymbol{\alpha}_{o,0} + H_4H_3^{-1}H_4\boldsymbol{\alpha}_{e,q}\right\} \\
&\qquad - \cos(\xi\Delta x)R_2^{-1}\boldsymbol{\alpha}_{e,q} \\
&= \left[\cos(\xi\Delta x)R_2^{-1} + I\right]H_3^{-1}\left[F\boldsymbol{\alpha}_{e,0} + H_4H_3^{-1}F\boldsymbol{\alpha}_{o,0}\right] \\
&\qquad + \left\{\left[\cos(\xi\Delta x)R_2^{-1} + I\right]\left[H_3^{-1}H_4\right]^2 - \cos(\xi\Delta x)R_2^{-1}\right\}\boldsymbol{\alpha}_{e,q}\,.
\end{aligned} \tag{7.67}$$

This is a recursive relation, which can be solved to obtain an explicit formula for $\boldsymbol{\alpha}_{e,q}$ in terms of the starting variables $\boldsymbol{\alpha}_{e|o,0}$:

$$\boldsymbol{\alpha}_{e,q} = \left[H_5 - I\right]^{-1}\left[(H_5)^q - I\right]\left[\cos(\xi\Delta x)R_2^{-1} + I\right]H_3^{-1}\left[F\boldsymbol{\alpha}_{e,0} + H_4H_3^{-1}F\boldsymbol{\alpha}_{o,0}\right] + (H_5)^q\,\boldsymbol{\alpha}_{e,0}\,. \tag{7.68}$$

Here,

$$H_5 \equiv \left[\cos(\xi\Delta x)R_2^{-1} + I\right]\left[H_3^{-1}H_4\right]^2 - \cos(\xi\Delta x)R_2^{-1}\,, \tag{7.69}$$

and we have used the identity

$$\sum_{q'=0}^{q-1}(H_5)^q = \left[H_5 - I\right]^{-1}\left[H_5^q - I\right]\,. \tag{7.70}$$

Substitution of Equation (7.67) into Equation (7.68) shows that Equation (7.68) satisfies the recursive relation for all integers $q \geq 1$.

Next, we need an equivalent expression for $\boldsymbol{\alpha}_{o,q}$. Combining Equations (7.63), (7.64), and (7.66) allows us to express $\alpha_{o,q,g}$ in terms of $\alpha_{o,q-1,g}$ and $\alpha_{e|o,0,g}$:

$$\begin{aligned}
\boldsymbol{\alpha}_{o,q+1} &= \boldsymbol{\alpha}_{o,q+\frac{1}{2}} + \boldsymbol{b}_{q+1} \\
&= \boldsymbol{\alpha}_{o,q+\frac{1}{2}} + R_2^{-1}\boldsymbol{\alpha}_{e,q+\frac{1}{2}} - R_2^{-1}\boldsymbol{\alpha}_{e,q} \\
&= H_3^{-1}\left[F\boldsymbol{\alpha}_{o,0} + H_4\boldsymbol{\alpha}_{e,q}\right] + R_2^{-1}H_3^{-1}\left\{F\boldsymbol{\alpha}_{e,0} + H_4H_3^{-1}F\boldsymbol{\alpha}_{o,0} + H_4H_3^{-1}H_4\boldsymbol{\alpha}_{e,q}\right\} \\
&\qquad - R_2^{-1}\boldsymbol{\alpha}_{e,q}
\end{aligned}$$

$$
\begin{aligned}
&= \left[H_3^{-1}F + R_2^{-1}H_3^{-1}H_4H_3^{-1}F\right]\boldsymbol{\alpha}_{o,0} + R_2^{-1}H_3^{-1}F\boldsymbol{\alpha}_{e,0} \\
&\qquad + \left\{H_3^{-1}H_4 + R_2^{-1}\left(H_3^{-1}H_4\right)^2 - R_2^{-1}\right\}\boldsymbol{\alpha}_{e,q}\,, \\
&\equiv \left[H_3^{-1}F + R_2^{-1}H_3^{-1}H_4H_3^{-1}F\right]\boldsymbol{\alpha}_{o,0} + R_2^{-1}H_3^{-1}F\boldsymbol{\alpha}_{e,0} + H_6\boldsymbol{\alpha}_{e,q}\,. \qquad (7.71)
\end{aligned}
$$

Here, we have defined

$$H_6 \equiv H_3^{-1}H_4 + R_2^{-1}\left(H_3^{-1}H_4\right)^2 - R_2^{-1}\,. \qquad (7.72)$$

By combining Equations (7.68) and (7.71), we can express $\boldsymbol{\alpha}_{e|o,Q}$ in terms of $\boldsymbol{\alpha}_{e|o,0}$ using matrix notation:

$$\boxed{\begin{bmatrix}\boldsymbol{\alpha}_{e,Q}\\ \boldsymbol{\alpha}_{o,Q}\end{bmatrix} = \begin{bmatrix}H_{7,ee,Q} & H_{7,eo,Q}\\ H_{7,oe,Q} & H_{7,oo,Q}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_{e,0}\\ \boldsymbol{\alpha}_{o,0}\end{bmatrix}}\,. \qquad (7.73)$$

For any positive integer q, the four $G \times G$ matrices are given by

$$
\begin{aligned}
H_{7,ee,q} &= \left[H_5 - I\right]^{-1}\left[(H_5)^q - I\right]\left[\cos(\xi\Delta x)R_2^{-1} + I\right]H_3^{-1}F + (H_5)^q\,, && (7.74a)\\
H_{7,eo,q} &= \left[H_5 - I\right]^{-1}\left[(H_5)^q - I\right]\left[\cos(\xi\Delta x)R_2^{-1} + I\right]H_3^{-1}H_4H_3^{-1}F\,, && (7.74b)\\
H_{7,oe,q} &= R_2^{-1}H_3^{-1}F + H_6H_{7,ee,q-1}\,, && (7.74c)\\
H_{7,oo,q} &= H_3^{-1}F + R_2^{-1}H_3^{-1}H_4H_3^{-1}F + H_6H_{7,eo,q-1}\,. && (7.74d)
\end{aligned}
$$

Finally, we combine Equations (7.51), (7.73), (7.61), and (7.60) to obtain an eigenvalue problem for ω:

$$
\begin{aligned}
\omega\begin{bmatrix}\boldsymbol{a}_e\\ \boldsymbol{a}_o\end{bmatrix} &= \begin{bmatrix}\boldsymbol{\alpha}_{e,Q}\\ \boldsymbol{\alpha}_{o,Q}\end{bmatrix}\\
&= \begin{bmatrix}H_{7,ee,Q} & H_{7,eo,Q}\\ H_{7,oe,Q} & H_{7,oo,Q}\end{bmatrix}\begin{bmatrix}\boldsymbol{\alpha}_{e,0}\\ \boldsymbol{\alpha}_{o,0}\end{bmatrix}\\
&= \begin{bmatrix}H_{7,ee,Q} & H_{7,eo,Q}\\ H_{7,oe,Q} & H_{7,oo,Q}\end{bmatrix}\left\{\begin{bmatrix}\boldsymbol{\varphi} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\varphi}\end{bmatrix}\begin{bmatrix}\mathcal{A}_e - A_e\\ \mathcal{A}_o - A_o\end{bmatrix} + \begin{bmatrix}\boldsymbol{a}_e\\ \boldsymbol{a}_o\end{bmatrix}\right\}\\
&= \boxed{\begin{bmatrix}H_{7,ee,Q} & H_{7,eo,Q}\\ H_{7,oe,Q} & H_{7,oo,Q}\end{bmatrix}\left\{\begin{bmatrix}\boldsymbol{\varphi} & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\varphi}\end{bmatrix}H_1^{-1}H_2 + I\right\}\begin{bmatrix}\boldsymbol{a}_e\\ \boldsymbol{a}_o\end{bmatrix}}\,. \qquad (7.75)
\end{aligned}
$$

As before, the spectral radius is given by the maximum value of $|\omega|$ over all error frequencies ξ, and ω can be computed by solving the eigenvalue problem in Equation (7.75).

From these results, we can also obtain two other results. To recover the decay factors and spectral radii derived for the MED method in which the multigroup problem is solved exactly, we can let $Q \to \infty$ in the above derivation. (Numerically, Q – the number of V-cycles performed on the multigroup system per MSED iteration – can be increased until convergence.) To obtain the spectral radius for an iteration scheme that uses the RBBJ smoother as a linear solver on the multigroup problem without multigrid-in-space, we can set R_2^{-1} to zero in all of the above expressions. This is equivalent to zeroing the error correction from the coarse spatial grid.

7.2.2.3 Numerical Results

Now, we compute estimates of the spectral radius of an MSED iteration by solving for ω in Equation (7.75) and taking its maximum magnitude over all permitted values of ξ. In this analysis, we consider a domain with $X = 100$ cm and $\Delta x = 1$ cm. (Previous Fourier analyses have indicated that the spectral radius is insensitive to Δx.) The cross sections used here are the same as those used in the Fourier analyses of Sections 6.2 and 7.2.1. Various values of Q are considered to assess the impact of the multigrid V-cycles on the multigroup system in an MSED iteration.

The results of the numerical computations are summarized in Table 7.6 and Figure 7.6. At $Q = 0$, the spectral radius is 1 since we are not solving the multigroup system at all. As $Q \to \infty$, we see that the spectral radius converges to the spectral radii from Table 6.1. This is expected: to generate the results in Table 6.1, we assumed that the multigroup system is solved exactly. Here, we are performing Q multigrid V-cycles on that system instead of solving it exactly. This agreement provides reassurance that the analysis we have performed here is correct and consistent with the analysis performed in Section 6.2.

Table 7.2: Spectral radii from Fourier analyses of MSED for different values of Q (the number of V-cycles on the multigroup system per MSED iteration) for four different cross section sets. A plot of the data in this table is shown in Figure 7.6. Here, the values under $Q = \infty$ represent the spectral radius of an MSED iteration in which the multigroup problem is solved exactly; these values are obtained from the $C = 1$ column in Table 6.1.

Library	$Q = 0$	$Q = 1$	$Q = 2$	$Q = 3$	$Q = \infty$
MPACT 8G	1.000	0.128	0.115	0.115	0.115
MPACT 47G	1.000	0.165	0.158	0.158	0.158
MPACT 51G	1.000	0.169	0.162	0.162	0.162
MPACT 252G	1.000	0.171	0.164	0.164	0.164

The most important feature of the data in Table 7.2 is that the spectral radius converges to the $Q = \infty$ value (to at least 3 decimal places) in just two V-cycles. This means that, for any of the four cross sections sets we have considered, performing 2 multigrid V-cycles in MSED is effectively equivalent to solving the multigroup problem exactly. Thus, as described in Algorithm 8, we optimized the design of MSED by only performing two multigrid V-cycles on the multigroup system per MSED iteration.[1] This design choice for MSED is especially desirable because the V-cycles performed on the multigroup system are the most computationally expensive part of an MSED iteration.

We still need many multigrid V-cycles on the grey diffusion eigenvalue problem, but these V-cycles are significantly cheaper than the V-cycles performed on the multigroup system. We note that we do not consider the cost of solving the grey eigenvalue problem in this analysis. In practice, even after we restrict Q to 2, the runtime required by the grey system is typically insignificant relative to the runtime required by the multigroup system.

The results of this Fourier analysis are surprising, and they indicate that there is a symbiotic relationship between the multilevel-in-energy and multigrid-in-space components of MSED. In Section 7.2.1, we found that the spectral radius for solving a fixed-source problem with multigrid is about 1/8. A naive guess would indicate that we would need $\sim$10 V-cycles to converge the multigroup fixed-source problem in MSED sufficiently well. However, the Fourier analysis tells us that the use of the grey diffusion problem has already eliminated many of the slowest converging modes for multigrid linear solver on the multigroup system. In Figure 7.7, we see that this mutually beneficial relationship is generally not present when other linear solvers are used for the multigroup system. When the RBBJ smoother is used as a linear solver, $O(100)$ or more iterations are needed to attain an MSED spectral radius that matches the $Q = \infty$ value.

[1] In fact, most of the benefit is already obtained after just one V-cycle, but we have found that negative scalar fluxes can appear when only one V-cycle is performed in the early iterations. Moreover, because our implementation will have more than two grids, its convergence rate will not match the idealized two-grid convergence rate. The application of a second V-cycle mitigates some of this drawback. When the problem is sufficiently converged and there are no negative fluxes, one multigrid V-cycle is typically sufficient.

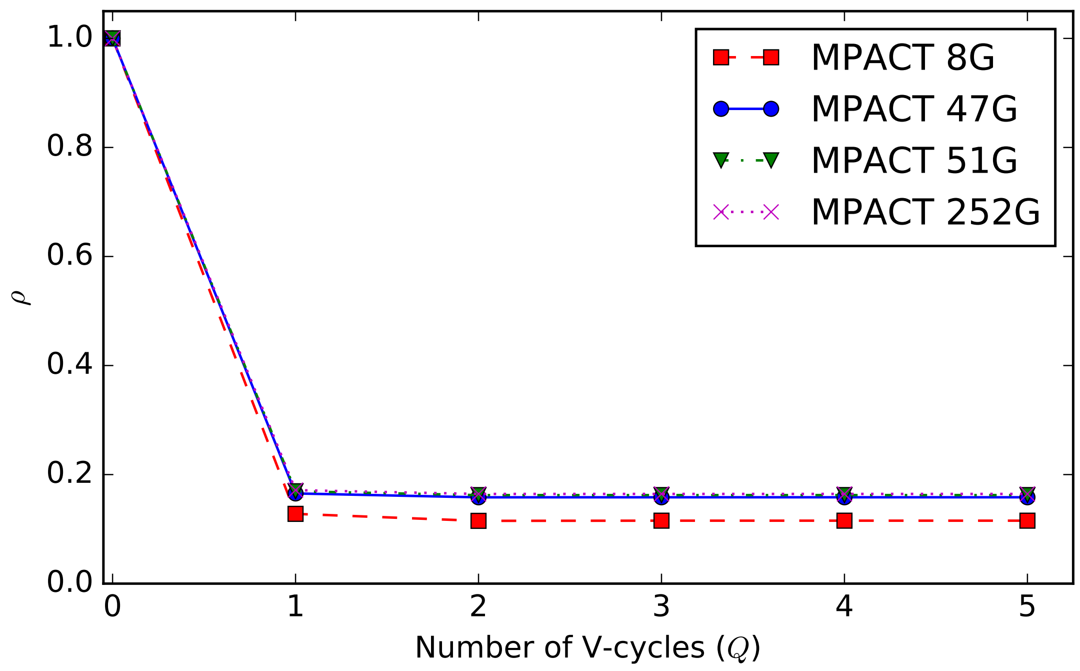


Figure 7.6: Spectral radii from Fourier analyses of MSED plotted against Q (the number of V-cycles on the multigroup system per MSED iteration) for four different cross section sets. The exact numerical values from the data in this plot is reported in Table 7.2.

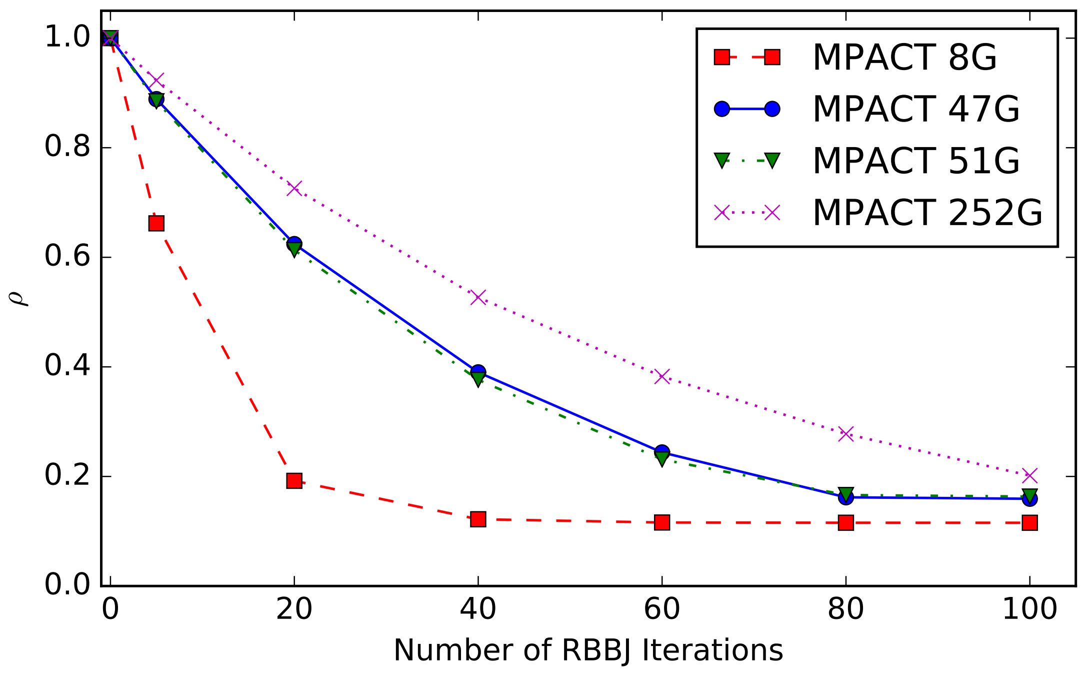


Figure 7.7: Spectral radii from Fourier analyses of MSED plotted against the number of multigroup RBBJ iterations per MSED iteration. Unlike Figure 7.6, only RBBJ iterations are performed – there are no multigrid V-cycles.

7.3 1-D Diffusion Results

We now revisit the 1-D simulations performed for Section 6.3 to assess the benefit provided by the multigrid-in-space linear solver. In Section 6.3, we assessed the benefit of leveraging a grey diffusion eigenvalue problem to solve the multigroup diffusion eigenvalue problem in the 1-D code described by Section 3.1. Two problems – the WB-1D-1 and WB-1D-2 problems described in Appendix B – are considered. In Tables 7.3 and 7.4, we reproduce the results from Tables 6.2 and 6.3 for these two problems, and add an extra result corresponding to the MSED method in Algorithm 8. (This extra result is bolded in each table.) The acronyms used in the tables have already been described in Section 6.3. The results in this section are updated versions of the results in [10].

We note that, for multigrid, the average number of GLSIs (V-cycles) per GPI are $\sim$8 and $\sim$12 for the two problems considered. The convergence criteria for the linear solver is 10^{-6}, and convergence in 10 iterations is equivalent to a spectral radius of $\sim$0.25. This is twice what was predicted by the Fourier analysis in Section 7.2.1. The primary reason for the difference stems from the fact that the implementation has more than two spatial grids while the Fourier analysis only considered the case with two grids. In the Fourier analysis, we assumed that the error on the second finest grid is solved exactly. When more than two grids are used in a V-cycle, however, this is not the case. From our experience, we have found that using multiple grids increases the spectral radius by a factor of two compared to just using two grids. This growth in the spectral radius can be eliminated by using more complex grid traversals such as a W-cycle rather than a V-cycle. However, we have found that it does not generally lead to an improvement in the overall runtime since W-cycles are costlier than V-cycles.

In Section 6.3, we saw that, even without restrictions on Q in Algorithm 8, multigrid outperforms all of the other linear solvers considered for the MSED method. Here, we see that, when we enforce $Q \leq 2$ (bolded results in Tables 7.3 and 7.4), we can further reduce the number of MGLSIs required, leading to a modest $\sim$10-25% reduction in the overall runtime. Overall, the runtimes for MED (without multigrid) and PI are at least $\sim$1.4 times and $\sim$2.5 times larger than the MSED runtime, respectively.

Table 7.3: Comparison of various solvers for the WB-1D-1 multigroup diffusion eigenvalue problem. Iteration counts and runtimes are shown for four different linear solver options. All of the results in this table are reproduced from Table 6.2, except the bolded line.

Method	Linear Solver	MGPIs	MGLSIs	GPIs	GLSIs	Runtime [s]
MED	GMRES-ILU	8	20	70	70	12.8
PI	GMRES-ILU	15	90	–	–	19.3
MED	BiCGSTAB-BJ	8	241	70	5541	19.7
PI	BiCGSTAB-BJ	15	2118	–	–	110.7
MED	BiCGSTAB-ILU	8	11	70	70	12.7
PI	BiCGSTAB-ILU	15	47	–	–	18.4
MSED	Multigrid (No limit on Q)	7	22	70	914	9.8
MSED	**Multigrid** ($Q \leq 2$)	**7**	**13**	**69**	**894**	**7.3**
PI	Multigrid	15	189	–	–	17.5

Table 7.4: Comparison of various solvers for the WB-1D-2 multigroup diffusion eigenvalue problem. Iteration counts and runtimes are shown for four different linear solver options. All of the results in this table are reproduced from Table 6.2, except the bolded line.

Method	Linear Solver	MGPIs	MGLSIs	GPIs	GLSIs	Runtime [s]
MED	GMRES-ILU	7	18	72	79	9.5
PI	GMRES-ILU	13	75	–	–	17.7
MED	BiCGSTAB-BJ	7	223	80	7899	19.0
PI	BiCGSTAB-BJ	13	1767	–	–	91.1
MED	BiCGSTAB-ILU	7	10	79	79	9.4
PI	BiCGSTAB-ILU	13	34	–	–	15.3
MSED	Multigrid (No limit on Q)	7	20	81	645	7.2
MSED	**Multigrid** ($Q \leq 2$)	**7**	**12**	**83**	**655**	**6.5**
PI	Multigrid	20	195	–	–	20.6

7.4 MPACT Results (2-D and 3-D CMFD)

In this section, we quantify the marginal speedups provided by the use of a multigrid-in-space linear solver in MSED and compare the performance of the MSED method defined by Algorithm 8 to other CMFD solvers in MPACT. As in Section 6.4, we consider the following seven problems: 4a, Peach Bottom 2-D (PB), 5a-0, 5a-2D, 5a-0-FC, AP1000-BOC, and AP1000-EOC. Iterations, runtimes, and memory costs are shown for these problems in Tables 7.5-7.11. The only new aspect of Tables 7.5-7.11 are the results using the MSED method; the other results in these tables are reproduced from Tables 6.9-6.15 for the convenience of the reader. The details of these problems were summarized in Section 6.4, and additional information can be found in Appendix C. Definitions of the different solvers considered in Tables 7.5-7.11 can be found in Sections 6.4 and 3.2.

To facilitate the analysis of these results, we remind the reader of the definitions of the acronyms used in the column headers of Tables 7.5-7.11. In these tables, the "Mem." column gives the maximum memory required by any processor over the course of the simulation in units of GB; this includes both the CMFD and non-CMFD components of MPACT. The "TSs" column gives the number of transport sweeps required over the entire simulation. TR is the total runtime required by the simulation, CR is the CMFD runtime required (this includes the runtime required by both the grey and multigroup systems in MSED), and GR is the runtime required in the grey system. CR is the primary quantity of interest in this thesis; these values are bolded in the tables. In this thesis, all runtimes are given in seconds.

MGPIs stands for multigroup power iterations. For "default", "PI+PWS," or "RBBJ," this is simply the number of power iterations performed. For MED or MSED, this is the number of times the multigroup linear system is solved (i.e., the number of times step 4 is performed in Algorithm 9), and it is also equal to the number of MED or MSED iterations. For GD, MGPIs is the number of GD iterations. MGLSIs stands for multigroup linear solver iterations; this is the total number of linear solver iterations required for all of the MGPIs in the simulation (not applicable to GD). GPIs (grey PIs) and GLSIs (linear solver iterations) are the corresponding quantities for the grey diffusion system in the MED and MSED methods.

In Tables 7.5-7.7, we see that the grey runtime (GR) is reduced by the use of a multigrid linear solver, but this is not true of the runtime required on the multigroup system. The total CMFD runtime is actually higher with MSED than MED for problems 7.5 and 7.7. In Table 7.6, we see that the difference in CMFD runtime is entirely accounted for by the difference in the grey runtime. In the previous chapter, we noted that the use of a grey diffusion system allows us to avoid the application of a WS on the multigroup system. Moreover, the use of the reduced multigroup approach described in Algorithm 10 allows us to reduce the size of the multigroup system. Both of these factors result in a multigroup fixed-source diffusion problem that is relatively well-conditioned

Table 7.5: Comparison of MSED to other MPACT CMFD solvers for 3-D, 9 assembly problem 4a. Iteration counts, runtimes, and memory costs are shown for each option; runtimes are given in seconds while memory is given in GB.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	12	104	9919	–	–	0.89	–	**152**	367
PI+PWS	12	49	6910	–	–	0.89	–	**115**	340
RBBJ	14	280	42000	–	–	0.84	–	**336**	589
GD	12	595	–	–	–	0.96	–	**29**	241
MED	12	38	936	147	17699	0.87	6	**15**	230
MSED	12	36	72	158	790	0.99	4	**22**	237

Table 7.6: Comparison of MSED to other MPACT CMFD solvers for the 2-D, full-core Peach Bottom (PB) problem.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	157	576	46615	–	–	0.94	–	**275**	1319
PI+PWS	21	153	24862	–	–	0.94	–	**168**	428
GD	27	4714	–	–	–	0.97	–	**95**	373
MED	27	91	2668	1108	181572	0.94	50	**65**	350
MSED	27	98	196	1278	6390	0.98	34	**51**	334

Table 7.7: Comparison of MSED to other MPACT CMFD solvers for the 3-D, quarter-core problem 5a-0.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	21	96	9600	–	–	2.72	–	**632**	1998
PI+PWS	12	52	8002	–	–	2.72	–	**558**	1484
RBBJ	15	300	45000	–	–	2.56	–	**1551**	2684
GD	12	749	–	–	–	3.01	–	**130**	1026
MED	12	37	1226	160	21794	2.67	23	**72**	988
MSED	12	36	72	172	860	2.78	16	**71**	982

Table 7.8: Comparison of MSED to other MPACT CMFD solvers for the quarter-core problem 5a-2D with MPACT's 252 group library.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	12	89	8900	–	–	4.74	–	**2904**	4510
PI+PWS	9	48	8275	–	–	4.74	–	**2943**	4433
RBBJ	12	213	31950	–	–	4.33	–	**14386**	15907
GD	–	–	–	–	–	–	–	–	DNC
MED	12	56	220846	229	29698	3.84	29	**139**	1396
MSED	12	56	42642	261	1305	3.85	29	**230**	1491

Table 7.9: Comparison of MED to other MPACT CMFD solvers for the 3-D, full-core problem 5a-0-FC.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	24	94	9400	–	–	2.39	–	**1918**	5015
PI+PWS	12	53	8351	–	–	2.39	–	**1770**	4124
GD	12	1034	–	–	–	2.54	–	**1830**	4030
RBBJ	–	–	–	–	–	–	–	–	DNC
MED	12	37	1273	156	21453	2.16	224	**436**	2474
MSED	12	37	74	184	920	2.84	59	**233**	1476

Table 7.10: Comparison of MSED to other MPACT CMFD solvers for the 3-D, full-core AP1000-BOC problem.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	27	126	12600	–	–	2.17	–	**2337**	6033
PI+PWS	11	53	8990	–	–	2.17	–	**1713**	3916
GD	8	643	–	–	–	–	–	–	DNC
RBBJ	16	316	47400	–	–	2.11	–	**2755**	5414
MED	11	33	1670	192	30614	2.19	353	**587**	3235
MSED	11	33	66	201	1005	2.80	52	**200**	2045

Table 7.11: Comparison of MSED to other MPACT CMFD solvers for the 3-D, full-core AP1000-EOC problem.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	95	417	41700	–	–	2.93	–	**2776**	19268
GD	15	1502	–	–	–	2.96	–	**1987**	6778
RBBJ	44	880	132000	–	–	2.80	–	**6939**	17548
MED	15	41	2814	280	45016	2.94	298	**441**	5278
MSED	15	41	82	349	1745	3.50	124	**247**	4498

and can be solved by GMRES in relatively few iterations (~ 30). For comparison, we note that 200 or more GMRES iterations are often needed to solve the grey fixed-source diffusion problem, due to the presence of the WS. Even though we only perform two multigrid V-cycles per MGPI, it is difficult for multigrid to perform better than GMRES when only 30 iterations are needed, especially if one accounts for the costs of setting up the coarse-grid operators in multigrid. (The coarse-grid operator setup costs are included in the CR and GR columns for MSED.) This is particularly true of the one-group systems in the one-group sweep approach used for the 252-group problem 5a-2D in Table 7.8. These one-group problems are even easier for GMRES to solve because of their small sizes. (In Chapter 2, we noted that one of the drawbacks of GMRES and other Krylov methods is their dependency on the condition number of the matrix, which typically grows with problem size.) As a result, MED is nearly $\sim$2x faster than MSED for the 252-group problem 5a-2D.

However, as we discussed in Section 2.2.3, the primary benefit of multigrid is the insensitivity of its convergence rate to problem size and, consequently, its optimal scaling with problem size. For the 3-D, full-core problems in Tables 7.9-7.11, we see that the use of the multigrid linear solver results in a $\sim$2x speedup over GMRES for *full-core* problems. That is, for problems 5a-0-FC, AP1000-BOC, and AP1000-EOC, the CMFD runtime required by "MSED" is approximately half of that required by MED. In many of these results, we note that the difference in CMFD runtime cannot account for all of the difference in total runtime, despite the fact that the number of transport sweeps is identical. This is because of issues in the design of the MPI communication in MPACT; more discussion on this issue is provided in Section 8.2.

One unfortunate drawback of using multigrid is the memory cost of its coarse-grid operators. In all of the problems (the larger ones in particular), the memory required by MSED is greater than the memory required by MED. Because high-performance computing systems tend to be memory-limited rather than flop-limited, this is an undesirable result. We note that one reason for the difference in memory between MED and MSED is the fact that we cannot use the reduced multigroup approach for full-core problems, due to the aforementioned MPI communication issues. These issues are discussed further in Section 8.2. Future work will strive to reduce the memory burden of MSED by (1) resolving the MPI issues so that we can use the reduced multigroup approach and (2) considering the use of alternative, less memory-intensive coarse-grid operators.

The comparison to the other CMFD solvers has already been discussed in Section 6.4 for the MED method. The corresponding comparisons to the MSED method are similar and can be readily discerned by combining the discussion in Section 6.4 and the discussion above comparing MED to MSED. Like MED, MSED also reduces the amount of work required by the transport sweeper by converging the CMFD sufficiently well and allowing the outer odCMFD-accelerated transport scheme to achieve the spectral radii predicted by [5]. Overall, MSED provides a speedup of $\sim$6-12x compared to the default CMFD solver in MPACT. For the 3-D, full-core problems, the speedup range is $\sim$8-12x.

To summarize, MSED, which uses multigrid as its linear solver, is $\sim$2x faster than MED, which uses GMRES as its linear solver, for *full-core* problems. For smaller problems (2-D and/or not full-core), multigrid provides a speedup over GMRES on the grey system, but not on the multigroup system, which is relatively well-conditioned due to the lack of a WS. In short, multigrid performs better than GMRES only when the problem size is sufficiently large. Thus, we recommend the use of GMRES (or another linear solver) for the multigroup system when using MSED on smaller problems. In all cases, however, the ill-conditioning caused by the WS on the grey diffusion problem makes it optimal to use multigrid there instead of GMRES.

CHAPTER 8

Additional Results and Discussion

In this chapter, we discuss and present results for several topics that were not covered in the previous two chapters. First, we discuss the issues with MPI communication in MPACT that were noted in Chapters 6 and 7. Second, we examine the performance of MSED under different parallel domain decompositions. Finally, we also consider problems with thermohydraulic and xenon feedback, and a 32-statepoint depletion problem. In order to simulate problems with feedback, we had to develop a "limited" MSED method. This is described and explained in Section 8.4. A summary of the overall performance of MSED is provided at the conclusion of this chapter.

All of the problems in this chapter were run on the Titan supercomputer at the Oak Ridge Leadership Computing Facility (OLCF) using the 51-group MPACT library. Details regarding the operation of the MPACT code and the different CMFD solvers available in MPACT are provided in Section 3.2. More details on the problems used in this chapter are given in Appendix C.

8.1 A Brief Comment on MSED vs. a Well-Known Two-Level CMFD Scheme

Before we discuss any of the aforementioned topics, we provide a brief commentary regarding MSED versus the well-known scheme described in [3] to assist the reader in contextualizing the performance of MSED. As noted in Chapter 1, the method in [3] is a two-level CMFD scheme in which a two-group CMFD problem on an assembly-wise mesh (i.e., significantly coarsened in space) is used as a lower-order accelerator to GS iterations on the standard pinwise multigroup CMFD problem. Unfortunately, this scheme is not available in MPACT for a direct comparison, and we were unable to implement it in MPACT due to the time constraints for this dissertation work. Thus, we can only provide a speculative comparison of the two methods based on the results of [3]. The work in [3] is implemented in the CASMO code, which solves 2-D, full-core problems rather than 3-D problems. Thus, the most comparable results from this thesis would be those of the 2-D, full-core PB problem, shown in Table 7.6.

In Table III of [3], we see that it takes somewhere between 2 and 5 fine-grid (i.e., the multigroup, pinwise grid) CMFD iterations in the two-level CMFD scheme to minimize the number of transport sweeps required. In Table 7.6 of this thesis, we see that the number of MSED iterations required per CMFD solve is approximately $\sim$3, so the outer spectral radii are somewhat comparable. (With MSED, the first few CMFD solves require more MSED iterations, while the subsequent/later CMFD solves only require 2 or 3 at most.) The minimum number of transport sweeps differ in the two cases, but this is likely due to differences in the problem parameters, not the efficiency of the CMFD schemes and solvers.

[3] notes that, because the spatial grid is significantly coarsened in the lowest level of the two-level CMFD scheme, the runtime required by the lowest level is negligible. The fine-grid spatial dependence of the solution is only resolved at the multigroup CMFD level in [3]. It is hard to compare this aspect to MSED, as the burden of resolving the fine-grid spatial dependence of the problem is shared between the grey and multigroup systems in MSED. Since [3] shows good results with a simultaneous coarsening in energy and space between just two levels, it may be possible that the efficiency of MSED may be improved by not fully resolving the fine-grid fission source distribution in the grey equation. However, the reader should note that the coarsest level of the multigrid-in-space linear solver in MSED is most likely resolving the same error frequencies as those resolved in the coarsest level of [3]. Structuring the collapse in space using a multigrid approach rather than a CMFD-like approach may also be beneficial, as it is less likely to excite any intermediate frequency error modes during the prolongation step. This aspect has been a general concern in traditional CMFD when updating fine grid fluxes (and has led to the development of stabilization techniques such as those in [59]).

Although it is difficult to do a proper comparison of the overall efficiency of the two methods without actually implementing them, we feel that our multigrid-in-space linear solver is more efficient than the GS solver used in [3]. Our reasoning for this stems from two factors: (1) the symbiotic relationship between multilevel-in-energy and multigrid-in-space seen in Section 7.2.1, and (2) the numerical results of Section 6.1.2, which indicate that a “one-group sweep” approach is far from optimal for typical 51-group problems. Moreover, while multigrid has nearly optimal scaling properties to larger (3-D, full-core) problems, our hypothesis is that the same cannot be said for the GS solver used in [3].

Finally, we recall from Section 6.4.2 that using two groups on the coarse energy grid does not provide much benefit in MSED. However, this may not be the case for the two-level CMFD scheme. There, the use of two coarse energy groups may be mitigating losses in the convergence rate of the fine-grid CMFD solver that stem from the simultaneous, aggressive coarsening in space between the two levels of CMFD.

In short, although both methods provide significant improvement over single-level CMFD methods/solvers such as the default method in MPACT, we feel that the use of a multigrid linear solver should allow MSED to scale better to 3-D, full-core problems than the two-level CMFD scheme in [3]. However, we cannot draw any confident conclusions without actually implementing the methods in the same context and comparing them.

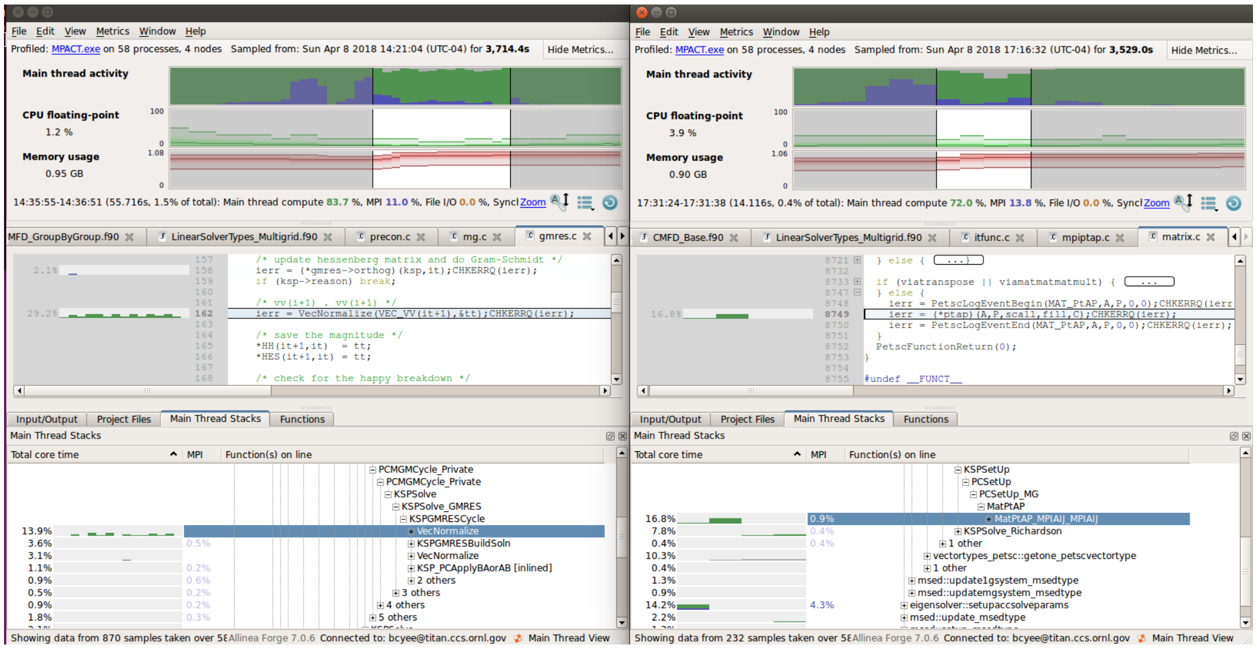
8.2 Performance Inconsistencies in MPACT

In this section, we discuss communication issues that we have alluded to several times in the previous two chapters. We have observed three phenomenon that are unusual and potentially related.

First, we have found that, for most problems, the reduced multigroup approach (Algorithm 10) does not result in a speedup over the standard, full multigroup approach for the multigrid-in-space linear solver in MPACT. This is unusual because, as discussed in Section 6.1.2, the number of computations required by the reduced approach should be significantly less than the full multigroup approach.

To better understand this phenomenon, we used the MAP profiler from the Arm Forge package [117] (previously known as Allinea) to study the performance of the MSED method using the reduced and full multigroup approaches. For the full multigroup approach, the most notable computational cost for multigrid is the computation of the coarse-grid operators on the multigroup fixed-source problem, comprising $\sim$10% of the total CMFD runtime for Problem 4a on Titan. On the other hand, the reduced multigroup approach has an entirely different bottleneck. Surprisingly, calls to the `VecNormalize` function in PETSc on the coarsest spatial grid are the most notable component from the MAP profile; these calls constitute $\sim$14% of the total CMFD runtime. A screenshot from the MAP profiler is shown for these two cases in Figure 8.1.

The results of this profile reinforce our intuition that the issue is communication-related. Although vector normalization is not normally a significant computational burden, it does require a synchronization of all processors. Moreover, although the reduced multigroup approach requires fewer computations than the full multigroup approach, it does require *more synchronization* because many more linear systems are solved. Each time a V-cycle reaches the coarsest grid, all the processors must synchronize to solve the coarse grid system. In our implementation, this means performing 15 communication-dominated GMRES iterations. (They are communication-dominated because the number of unknowns is relatively small relative to the number of processors.) The reduced multigroup method results in far more V-cycles performed since it breaks up the multigroup linear system into $\sim G/2$ linear systems. This translates into a significant increase in the number of MPI messages that need to be sent, and, if the MPI communication is not implemented well in the code or the system, the runtime can be hindered by a few processors waiting for their messages



Profiled: MPACT.exe on 58 processes, 4 nodes Sampled from: Sun Apr 8 2018 14:21:04 (UTC-04) for **3,714.4s**

CPU floating-point 1.2 %

Memory usage 0.95 GB

14:35:55-14:36:51 (55.716s, 1.5% of total): Main thread compute 83.7 %, MPI 11.0 %, File I/O 0.0 %, Sync

MFD_GroupByGroup.f90 | LinearSolverTypes_Multigrid.f90 | precon.c | mg.c | gmres.c

```
        /* update hessenberg matrix and do Gram-Schmidt */
2.1%    ierr = (*gmres->orthog)(ksp,it);CHKERRQ(ierr);
        if (ksp->reason) break;

        /* vv(i+1) . vv(i+1) */
29.2%   ierr = VecNormalize(VEC_VV(it+1),&tt);CHKERRQ(ierr);

        /* save the magnitude */
        *HH(it+1,it)  = tt;
        *HES(it+1,it) = tt;

        /* check for the happy breakdown */
```

Total core time	MPI	Function(s) on line
		PCMGMCycle_Private
		PCMGMCycle_Private
		KSPSolve
		KSPSolve_GMRES
		KSPGMRESCycle
13.9%		VecNormalize
3.6%	0.5%	KSPGMRESBuildSoln
3.1%		VecNormalize
1.1%	0.2%	KSP_PCApplyBAorAB [inlined]
0.9%	0.6%	2 others
0.5%	0.2%	3 others
0.9%	0.2%	4 others
1.8%	0.3%	5 others

Showing data from 870 samples taken over 58 Allinea Forge 7.0.6 Connected to: bcyee@titan.ccs.ornl.gov

Profiled: MPACT.exe on 58 processes, 4 nodes Sampled from: Sun Apr 8 2018 17:16:32 (UTC-04) for **3,529.0s**

CPU floating-point 3.9 %

Memory usage 0.90 GB

17:31:24-17:31:38 (14.116s, 0.4% of total): Main thread compute 72.0 %, MPI 13.8 %, File I/O 0.0 %, Sync

CMFD_Base.f90 | LinearSolverTypes_Multigrid.f90 | itfunc.c | mpiptap.c | matrix.c

```
        } else { ...}

        if (viatranspose || viamatmatmatmult) { ...
        } else {
          ierr = PetscLogEventBegin(MAT_PtAP,A,P,0,0);CHKERRQ(ierr
16.8%     ierr = (*ptap)(A,P,scall,fill,C);CHKERRQ(ierr);
          ierr = PetscLogEventEnd(MAT_PtAP,A,P,0,0);CHKERRQ(ierr);
        }
        PetscFunctionReturn(0);
      }

      #undef __FUNCT__
```

Total core time	MPI	Function(s) on line
		KSPSetUp
		PCSetUp
		PCSetUp_MG
		MatPtAP
16.8%	0.9%	MatPtAP_MPIAIJ_MPIAIJ
7.8%	0.4%	KSPSolve_Richardson
0.4%	0.4%	1 other
10.3%		vectortypes_petsc::getone_petscvectortype
0.4%		1 other
1.3%		msed::update1gsystem_msedtype
0.9%		msed::updatemgsystem_msedtype
14.2%	4.3%	eigensolver::setupaccsolveparams
2.2%		msed::update_msedtype

Showing data from 232 samples taken over 58 Allinea Forge 7.0.6 Connected to: bcyee@titan.ccs.ornl.gov

Figure 8.1: Screenshot of MAP profiles of Problem 4a using the reduced multigroup (left) and full multigroup (right) formulations in MSED. The profile has been truncated to only consider (approximately) the timing statistics during the first CMFD solve. The left figure indicates that 14% of the total MSED runtime is spent in the `VecNormalize` function, while the right figure indicates that ∼17% of the total MSED runtime is spent generating the coarse-grid operators.

to be sent or received. This is especially true of highly parallel supercomputers such as Titan, in which the permitted volume of MPI messages is lower than in smaller computing clusters. Future work will include efforts to mitigate or eliminate these communication inefficiencies so that we can realize the benefits of using the reduced multigroup approach for the multigrid linear solver.

Second, we have also observed that the difference between the CMFD runtimes does not account for all of the difference between the total runtimes. For example, in Table 7.9, the difference in CMFD runtime between the MED and MSED cases is only ∼200 seconds, but the difference in total runtime is nearly 1000. Similar examples can also be found in the results of Tables 7.10 and 7.11. In the timings reported by the MPACT code, we see that this difference stems from differences in the communication time required by the transport (MOC) solver. This indicates that the communication performed in the CMFD solver can negatively impact the communication performed in the transport solver, and that this observation may be related to the previous observation.

Our third observation is that there are likely inefficiencies and/or bugs in the MPI communication in MPACT that are unrelated to its CMFD solver. Often, even when MSED and MED are not used, a processor will fail to receive a message it is waiting for, resulting in the code stalling indefinitely. For example, this occurs if we try to run problem 5a-2D with 252 groups using MSED on Titan. Moreover, the performance of MPACT on Titan exhibits stochastic behavior, though this may have more to do with the file system than MPI. For example, we have seen that the setup of the transport mesh for the AP1000-BOC problem in Tables 6.14 and 6.15 can sometimes take an extra $\sim$5 minutes. In such cases, we are forced to rerun the problem so that our timing results are as independent as possible of this stochastic behavior.

Despite the impact of stochastic machine behavior on the total runtime, we have found the CMFD runtime to be fairly deterministic. For a given problem and CMFD solver method, variations in the CMFD runtime from simulation to simulation are usually less than $\sim 1\%$. Nonetheless, we acknowledge that the results in this thesis may be affected by the issues discussed in this section. That is, although we still believe that MSED is an optimal method for solving multigroup diffusion and CMFD eigenvalue problems, the magnitude of the relative speedups provided by MSED over alternative methods may change if the MPI implementation in MPACT were improved.

Overall, we have found that (1) these issues are more common on Titan than on Eos, indicating that the problems may be partially caused by or at least exacerbated by hardware limitations, and (2) these issues are more common with larger problems, indicating that we may be overloading the network channels with too many messages. The issues that we have discussed here provide a strong motivation to examine and improve upon MPACT's MPI implementation in future work.

8.3 Impact of Spatial Decomposition on the Performance of MSED

In this section, we experiment with the parallel spatial decomposition to see if there is any impact on the speedups provided by MSED. In Tables 8.1 and 8.2, we compare the performance of MSED (and other CMFD solvers in MPACT) using two different spatial decompositions for the 3-D, quarter-core problem 5a-0. For both cases, 464 cores over 58 nodes are used, and the problem is simulated using MPACT's 51-group library. Moreover, in each case, the assemblies are divided into 16 sets, and each processor is assigned one plane from one of the 16 sets of assemblies. (464 spatial divisions = 16 radial divisions per plane $\times$ 58 planes.) The only difference between the two decompositions is in the radial decomposition of the assemblies. In Table 8.1, an explicit, user-defined radial partition of the assemblies is used [118]. In Table 8.2, an automatic radial partition, developed by Fitzgerald et al. [119], is used. The acronyms in these two tables are the same as those in Sections 6.4 and 7.4.

From these results, we see that the the relative performances of the CMFD solvers are similar for the two spatial decompositions. The automated partition results in a slightly increased or decreased CMFD runtime for all of the methods. In the standard partition, MSED provides a $\sim$12x reduction in the CMFD runtime over the default CMFD solver. In the automated decomposition, the same

Table 8.1: Performance of MSED and other CMFD solvers for problem 5a-0 on Titan with the standard, manually-specified "explicit radial" spatial partition. Iteration counts, runtimes, and memory costs are shown for each option; runtimes are given in seconds while memory is given in GB. We note that the results here differ from the results in Tables 6.11 and 7.7, which were generated on Eos instead of Titan.

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	21	96	9600	–	–	2.74	–	**3936**	7499
PI+PWS	12	52	8002	–	–	2.74	–	**3382**	5890
RBBJ	15	300	45000	–	–	2.56	–	**5884**	8679
GD	12	749	–	–	–	3.02	–	**637**	2876
MED	12	37	1226	160	21794	2.7	247	**629**	2918
MSED	12	36	72	173	865	3.14	53	**324**	2663

Table 8.2: Performance of MSED and other CMFD solvers for problem 5a-0 on Titan with the automated spatial decomposition described in [119].

Solver	TSs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	21	96	9600	–	–	2.71	–	**3974**	7475
PI+PWS	12	50	7740	–	–	2.71	–	**3286**	5719
RBBJ	15	300	45000	–	–	2.54	–	**5698**	8599
GD	12	772	–	–	–	2.94	–	**689**	2942
MED	12	37	1257	161	22407	2.67	251	**661**	3035
MSED	12	36	72	180	900	3.05	60	**360**	2726

speedup figure is $\sim$11x. It appears that the MED and MSED methods have a slightly larger total runtime when the automated decomposition is used rather than the standard decomposition. This may be a result of some of the communication issues discussed in the previous section. However, these differences are insignificant compared to the savings in CMFD and total runtime from using MED or MSED instead of the default CMFD solver.

In short, the performance of MSED appears to have some dependence on the spatial decomposition, but the dependence is relatively small, especially compared to the savings in total runtime from using MSED instead of the MPACT's default CMFD solver.

8.4 Problems with Feedback

Here, we consider two problems with thermohydraulic and xenon feedback: problem 7 and problem AP1000-EOC-HFP. Problem 7, which was used to generate results in Chapter 4, is a critical boron search problem for a 3-D, quarter-core model of the Watts Bar Unit 1 reactor. Problem AP1000-

EOC-HFP is the same as the AP1000-EOC problem used in the previous two chapters, except it is at hot full power (HFP) rather than hot zero power (HZP). The results for these two problems are presented in Tables 8.3 and 8.4. Results for many of the alternative solver options are unavailable for the AP1000-EOC-HFP problem due to time and computational resource constraints. Except for two new acronyms, the acronyms used in these tables are the same as those used in the tables in Sections 6.4 and 7.4. Descriptions of the old acronyms can be found in these two sections, as well as in the list of acronyms at the beginning of the thesis. The two new nomenclatures are EXSCs, which stands for equivalence cross-section calculations, and the "-L," which stands for "limited," that is appended to some solvers. The relevance of these two new nomenclatures is explained in the following paragraphs.

Table 8.3: Comparison of MSED to other MPACT CMFD solvers for problem 7 with feedback. Iteration counts, runtimes, and memory costs are shown for each option; runtimes are given in seconds while memory is given in GB. The second entry for the default solver, differentiated by an asterisk ("Default*"), corresponds to a case with a tightened CMFD convergence criteria.

Solver	TSs	EXSCs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	15	6	107	10700	–	–	3.15	–	**4469**	10230
Default*	19	9	271	27100	–	–	3.17	–	**11351**	19371
PI+PWS	14	11	75	12551	–	–	3.17	–	**5380**	13408
RBBJ	11	6	159	23850	–	–	2.89	–	**3122**	8303
GD	18	11	1528	–	–	–	3.33	–	**1322**	9782
MED-L	13	7	29	1269	135	12212	3.1	155	**555**	7020
MSED	20	11	58	116	513	2565	3.37	154	**624**	9749
MSED-L	12	7	27	54	133	665	3.39	37	**300**	6057

Table 8.4: Comparison of MSED to other MPACT CMFD solvers for the AP1000-EOC-HFP problem with feedback

Solver	TSs	EXSCs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	21	8	167	16700	–	–	2.98	–	**971**	9309
MSED-L	13	9	34	68	169	845	3.52	75	**189**	7833

As described in Chapter 3, MPACT accounts for feedback by calculating updated temperatures (via a thermohydraulics code) and xenon concentrations after each transport sweep, using these quantities to generate updated cross section values, and then, if these updates are significant, performing an EXSC to generate new self-shielding corrected "equivalence" cross sections. (EXSCs are described in [4, 120].) This simple Picard iteration scheme, however, can be slow or unstable. One of the primary weaknesses of MPACT is that the CMFD system has no knowledge of the feedback mechanisms. The temperature-dependent cross sections in the CMFD problem are "lagged" due to the Picard iteration scheme, and "fully converging" the CMFD system is not desirable in the early iterations since these cross sections may not be close to their converged values. In fact, as we have seen in the results of Chapter 4, it may actually *increase* the number of Picard iterations (i.e., the number of transport sweeps and cross section updates) required to converge. This is especially undesirable because transport sweeps and EXSCs are the typically the most time-consuming components of MPACT.

Paradoxically, the poor convergence of the default solver in MPACT can actually act as a pseudo-relaxation mechanism that improves the convergence rate of the outer Picard iteration scheme, reducing the number of transport sweeps required. This is seen in the results of Table 8.3. For the default CMFD solver, the number of transport sweeps required and and the number of EXSCs required are only 15 and 6, respectively. However, GD and MSED, which typically produce a more well-converged CMFD system than the default CMFD solver, require 18 and 20 transport sweeps, respectively, and 11 EXSCs. When the CMFD system is better converged, the Picard iteration scheme tends to be more "oscillatory"; large corrections to the temperatures occur more frequently, resulting in more EXSCs.

We see the same phenomenon when we force the default CMFD solver to perform more iterations to better converge the CMFD system; this case is denoted by "Default*" in Table 8.3. In the "Default*" case, the CMFD convergence tolerances are all tightened by an extra factor of 10 compared to the "default" case, and the maximum number of MGPIs permitted is increased from 20 to 50. The total and CMFD runtimes increase when the "default*" solver is used, but this is expected since we are deviating from a previously empirically determined convergnece criteria for CMFD that optimizes the balance between CMFD and non-CMFD runtime. What is surprising, however, is that the total number of TSs and EXSCs increases when the CMFD system is better converged using the default CMFD solver. This verifies that the phenomenon we are observing exists regardless of the CMFD solver (though the impact of this phenomenon is solver-dependent). We note that this paradoxical behavior is consistent with the results from [103], which indicates that the basic Picard iteration scheme often requires relaxation for stability when flux-dependent cross sections are present.

To mitigate these issues, we have decided to "limit" the convergence of MSED for feedback problems by

1. loosening the relative residual tolerance for CMFD from 0.01 to 0.05,

2. loosening the eigenvalue tolerance for the CMFD solver[1] from 10^{-6} to 10^{-4},

3. reducing the limit on the number of grey PIs (GPIs) per MSED iteration from 20 to 5, and

[1]The *outer* transport eigenvalue tolerance is still 10^{-6}.

4. using a fixed $2/3$ Wielandt shift on the grey eigenvalue problem instead of the more aggressive and iteration-dependent PARCS WS.

This "limited" solver is denoted in Tables 8.3 and 8.4 by the "-L" appended to the end of the solver name. The idea behind this limited solver is that we should be able to achieve a poorly converged CMFD system using MSED in less time than the default CMFD solver. That is, instead of generating a poorly converged CMFD solution using the default CMFD solver, we can efficiently obtain a similarly poorly converged system by using MSED with loosened convergence criteria. When this is done, we see in Tables 8.3 and 8.4 that we can achieve similar or lower TSs and EXSCs counts with significantly reduced CMFD runtimes. For problems 7 and AP1000-EOC-HZP, we see reductions in the CMFD runtime of $\sim$15x and $\sim$5x, respectively, over the default CMFD solver when using MSED-L. Unlike the GD and MSED methods, the MSED-L method does not result in an increase in the non-CMFD runtime (i.e., the difference between TR and CR in the tables).

The results here show that MSED-L can significantly improve upon the default CMFD solver for feedback problems. However, we believe that "limiting" MSED is not an optimal solution and should only be a temporary approach. Future work should consist of improving the outer feedback iteration scheme in MPACT so that the number of TSs and EXSCs is not increased by the improved convergence of the inner CMFD system. In particular, including some of the feedback physics in the CMFD system, even in a crude or simple manner, could lead to significant speedups in the overall iteration scheme.

8.5 Depletion Problem

In this section, we examine the performance of the MSED-L method described in the previous section for problem 9, a 32-statepoint depletion problem with thermohydraulic and xenon feedback. This problem is run on 464 cores over 58 nodes on Titan. Multiple restarts of the simulation are necessary due to the 28-hour walltime limit for all computing jobs on Titan. The final results for this simulation are shown in Table 8.5.

Table 8.5: Comparison of MSED to other MPACT CMFD solvers for problem 9, a 32-statepoint depletion problem with feedback. Iteration counts, runtimes, and memory costs are shown for each option; runtimes are given in **hours** while memory is given in GB. We note that the CR and GR are estimates extrapolated from the output of the final restart file.

Solver	TSs	EXSCs	MGPIs	MGLSIs	GPIs	GLSIs	Mem.	GR	**CR**	TR
Default	650	192	2925	2.92e5	–	–	3.63	–	**42.2**	123.5
MSED-L	552	186	1133	2.27e3	5599	2.80e4	3.93	1.6	**6.7**	91.0

In the results, we see that the CMFD runtime has been reduced by a factor of $\sim$6 while the total runtime has been reduced by $\sim$26%. We note that runtimes are given in hours rather than seconds here. The importance of the speedups seen for this problem are amplified by the large magnitude of the core-hours required by this problem. The default method required an extra restart because of the longer runtime; MSED finished $\sim$9 hours into its fourth 28-hour job, while the constant method finished $\sim$22 hours into its fifth 28-hour job. We also see that the difference in total runtime is smaller than the difference in CMFD runtime, despite the fact that MSED-L requires fewer TSs and EXSCs. As in previous cases, this is due to increases in the communication time in the transport solver. It may be possible to see more improvements in the total runtime if the communication in MPACT is improved. Nonetheless, the present total runtime savings provided by MSED-L is significant in both absolute and relative terms.

8.6 Final Discussion on Performance of MSED

In this final discussion section, we summarize our findings on the performance of MSED as a CMFD solver in the MPACT code.

For single statepoint calculations without feedback on 3-D, full-core problems, MSED provides a $\sim$8-12x speedup in the CMFD runtime over the default CMFD solver in MPACT. The same figure for the MED method, which is just MSED with GMRES instead of multigrid-in-space, is $\sim$4-8x for 3-D, full-core problems. However, the use of MSED can result in a $\sim$20% increase in the memory usage over MED, so MED may be a better option if memory is limited. Moreover, as discussed in Section 8.2, there are still some unresolved MPI issues that may be limiting the speedups provided by the MSED method.

For single statepoint calculations without feedback on smaller (2-D and/or non-full-core) problems, we do not see a significant speedup (if any) for MSED over MED. With either MED or MSED, however, we still see an order of magnitude speedup in the CMFD solver over the default CMFD method. For geometrically smaller problems with a large number of groups (e.g., problem 5a-2D with 252 groups), the MED method with the one-group sweep approach described by Algorithm 9 appears to be the best CMFD solver we have considered.

For all single statepoint calculations without feedback, the benefits provided by the use of MSED (or MED) over the default CMFD solver typically go beyond the savings in CMFD runtime. The default CMFD solver is often unable to sufficiently converge the CMFD problem, and extra transport sweeps are often required as a result. These transport sweeps are expensive and lead to significant increases in the total runtime required.

For calculations with feedback, a "limited" MSED method, described in Section 8.4, allows us to achieve an order of magnitude speedup in the CMFD runtime required, without any drawbacks in the non-CMFD runtime. This efficiency translates well to the 32-statepoint problem considered in Section 8.5, where we see significant speedups in both the CMFD and total runtime required when MSED-L is used instead of the default CMFD solver.

Finally, we return to the bigger picture discussed in the motivation section of Chapter 1. With the default CMFD solver, CMFD was a bottleneck, often taking up more than 50% of the total MPACT runtime. *With MSED, the resulting CMFD runtime is now less than 10% in most problems, and no more than 16% of the total runtime in any of the problems we have considered.* Thus, CMFD is no longer a computational bottleneck in MPACT. In many problems, the total improvement is not fully captured by the improvement in the CMFD runtime, because the number of sweeps required by the outer transport solver is also significantly reduced by the use of MSED.

CHAPTER 9

Future Work

In this chapter, we briefly outline potential avenues of future work. Most of these topics have been mentioned in earlier chapters of the thesis, particularly at places where suboptimal performance results are discussed.

With regards to the MSED method specifically, there is still considerable room for improvement for the multigrid solver. In particular, reducing the memory burden of the multigrid solver would be very beneficial. Because the trend in high-performance computing is toward systems that are more memory-limited than FLOP-limited, speedups in runtime may not be beneficial if they are accompanied by significant increases in memory usage. The bulk of the memory burden (and also computational overhead) in multigrid is a result of the computation of the coarse-grid operators via the Galerkin triple matrix product (Equation (2.30)). In Appendix A of [15], Stüben suggests the use of an "aggressive coarsening" in which one skips the second coarsest grid. The primary drawback of this is that the error modes that would have been the highest frequency error modes on the second coarsest grid would no longer be converged efficiently. This drawback can be mitigated by the use of more complex interpolation operators and/or polynomial smoothing operators, but this can result in significant additional computations and communication in the interpolation and/or smoothing steps. Direct truncation of the entries of the coarse-grid operators (i.e., zeroing out matrix elements that fall below a threshold magnitude) may help with the memory cost as well. However, Stüben warns in Remark A.2.4 of [15] that this would lead to a violation of the variational property satisfied normally by Galerkin coarse-grid operators. The consequences of this could be poor convergence or even divergence. Alternatively, because the grid in MPACT is relatively simple, a more geometric approach to defining coarse-grid operators, which does not rely on the Galerkin definition of coarse-grid operators, may be more appropriate. Again, careful analysis would have to be done to ensure stability and good convergence rates.

Memory issues aside, we note that it may also be possible to improve the smoother used in the multigrid-in-space linear solver, especially for multigroup problems. Our current implementation in MPACT uses a simple GS-like smoother, which often requires extra smoother iterations to prevent the appearance of negative fluxes, rather than RBBJ, which results in an optimal spectral radius for multigrid but does not scale well with the number of groups G. It may be possible to find a smoother that "falls in the middle" – i.e., more robust than the current GS-like smoother, but having better scaling with G than RBBJ.

For problems larger and "more parallelized" (lower spatial cells to cores ratio) than those considered in this thesis, we noted that inter-processor interpolation and restriction may be necessary in order to reach a sufficiently small coarsest grid. In such cases, it may be possible to optimize the interpolation/restriction with the specific machine on which the code is run. For example, one could design interpolation/restriction operators that only require intra-node communications. That is, only processors residing on the same node would have to communicate in the interpolation/restriction step.

For improving the multilevel-in-energy component of MSED, a natural next step would be to consider multiple levels in energy as Cornejo has done in the multilevel LONDA method [66]. This would improve the scaling of MSED's efficiency with the number of groups G. The potential benefits of using multiple levels in energy were discussed in Chapter 6 and can be seen in the results of [66].

With regards to the one-group sweep approach defined in Algorithm 11, it may be possible to optimize the upscatter iterations by iterating more frequently on groups that have more difficulty converging. The result would be an "adaptive" Gauss-Seidel algorithm that tracks the convergence of each group and iterates on slowly-converging groups more often than on fast-converging groups. The motivation for this is that, among all of the groups with upscattering sources, the relative size of the upscattering component varies significantly. The first (fastest) group with upscattering has a significantly weaker dependence on upscattering than the thermal groups. This adaptive Gauss-Seidel scheme could be beneficial if many upscattering iterations are needed to converge in the one-group sweep approach.

Lastly, as discussed in Chapter 8, improvements to the MPI communication and feedback iteration scheme could significantly reduce the runtime and memory costs of MPACT. Inefficiencies in the communication between processors have prevented us from using the reduced multigroup approach for the multigrid-in-space linear solver. Using the reduced multigroup approach would allow us to mitigate some of the memory burden associated with the current multigrid linear solver. Improvements to the feedback iteration scheme would allow us to avoid using the "limited" MSED method described in Section 8.4. Although this limited MSED method has resulted in significant improvements over the default CMFD solver, the overall iteration scheme is still not ideal. For example, the convergence criteria for "limited" MSED must be carefully and empirically tuned for optimal use in the current feedback iteration scheme. The incorporation of some feedback physics into the CMFD system in MPACT, even on a simple or crude scale, should lead to an overall iteration scheme that is both more robust and faster converging.

CHAPTER 10

Conclusions

In this final chapter, we summarize the content that has been presented in this thesis.

The primary focus of this thesis was the development of the Multilevel-in-Space-and-Energy Diffusion (MSED) method for solving multigroup neutron diffusion and CMFD eigenvalue problems. Although diffusion problems do not provide the full fidelity found in the corresponding transport problems, they play an important role as lower-order (computationally less expensive) but reasonably accurate approximations to transport problems. Even when the full fidelity of the multigroup neutron transport eigenvalue problem is needed, the solution of diffusion eigenvalue problems can be used to accelerate the convergence of transport solvers via techniques such as CMFD. The motivation for our work was the inefficiency of the CMFD solver in MPACT, which resulted in CMFD taking 50% or more of MPACT's runtime and suboptimal convergence rates for the outer transport solver in certain problems of interest.

Whereas CMFD is a technique for solving transport problems by leveraging a smaller multigroup diffusion-like problem, MSED is a technique for solving multigroup diffusion and CMFD eigenvalue problems by leveraging even smaller problems with no energy dependence and/or coarsened spatial grids. MSED combines a multilevel-in-energy approach with a multigrid-in-space linear solver; a visualization of MSED was presented in Figures 5.1 and 5.2. In Chapter 1, we presented a history of multilevel techniques from reactor physics and multigrid techniques from the general mathematics community, and we compared the MSED method to these historical methods. Chapter 2 provided some of the theory and background on reactor physics, linear solvers, and eigenvalue solvers needed to understand and implement the work in this thesis.

In Chapter 4, our earlier work on space-dependent Wielandt shifts (SDWSs) was presented. In this work, we attempted to improve upon existing Wielandt shifted PI schemes by using a physics-based, space-dependent shift rather than the standard empirically-derived, space-independent shifts. Although we saw modest improvements in 1-D diffusion eigenvalue problems, these results did not translate well to higher dimensional CMFD problems. We found that the performance of SDWSs (as well as other WSs) was limited by the tradeoff between the spectral radius of PI and the condition number of the diffusion operator that needed to be solved at each PI. A detailed study of this tradeoff was presented in Section 4.3. The shortcomings of SDWSs ultimately motivated our development of the MSED method.

The theory and development of the MSED method was divided into two chapters, 6 and 7, each corresponding to one of its two primary components – the grey diffusion acceleration scheme (multilevel-in-energy) and the multigrid-in-space linear solver. In these two chapters, Fourier analyses were presented alongside results from the 1-D diffusion code and MPACT. Our Fourier analyses indicated that the grey diffusion acceleration scheme has a spectral radius of $\sim$0.16, while a two-grid multigrid-in-space linear solver has a spectral radius of $\sim$0.125 for both multigroup and grey fixed-source diffusion problems. A combined analysis of these two components indicated that MSED could be optimized by restricting Q (the number of multigrid V-cycles on the multigroup system per MSED iteration) to 2. This was an especially desirable optimization, because the work performed on the multigroup system constitutes the bulk of the runtime required by MSED. In our 1-D results, we verified some of the convergence rates predicted by the Fourier analyses, as well as the benefits of restricting Q to 2.

At the end of Chapters 6 and 7, results were presented to verify and quantify the speedups from the grey diffusion acceleration and the multigrid-in-space linear solver, respectively. We found that the use of a grey diffusion eigenvalue problem (i.e., the MED method described in Chapter 6) resulted in speedups in the CMFD solver ranging from $\sim$4-20x. For single statepoint 3-D, full-core problems specifically, the speedups ranged from $\sim$4-6x. We found that the multigrid-in-space linear solver did not provide significant speedups over GMRES for smaller (not 3-D and/or not full-core) problems. However, one of the defining attributes of multigrid is the insensitivity of its convergence rate to problem size. For 3-D, full-core problems, the incorporation of the multigrid-in-space linear solver into MSED resulted in a further $\sim$2x speedup in the CMFD runtime compared to just using GMRES as the linear solver.

In Chapter 8, additional problems from MPACT were considered. We verified the relative insensitivity of MSED to the radial spatial decomposition (i.e., how the domain is divided among processors radially). We also showed that a modified, "limited" MSED method can extend the speedups seen in Chapter 7 to problems with feedback, despite lingering convergence and robustness issues with the outer feedback iteration mechanism in MPACT. Moreover, a discussion of some unusual MPI behavior was presented; from our results, we see evidence that the implementation of MPI communication in MPACT may require repair and improvement. A final discussion of the performance of MSED in MPACT is presented in the final section of Chapter 8. In Chapter 9, several avenues of future work are briefly discussed.

Overall, the MSED method, with both its grey diffusion acceleration and multigrid-in-space linear solver, provided a total speedup of $\sim$6-12x over the de facto default CMFD solver in MPACT.

The benefits of using MSED goes beyond the reduction in CMFD runtime. The improved efficiency from using MSED allows for better convergence of the CMFD problem, which in turn allows the outer CMFD-accelerated transport scheme to achieve the optimal convergence rates predicted by [5]. With MSED, the CMFD runtime is now no more than 16% of the total runtime in any of the problems we have considered, and, for most problems, it is less than 10% of the total runtime. Thus, the goal of removing the computational bottleneck in the CMFD component of MPACT has been accomplished. Figure 10.1 provides a visualization of the overall speedup provided by MSED for *3-D, full-core* problems in MPACT.

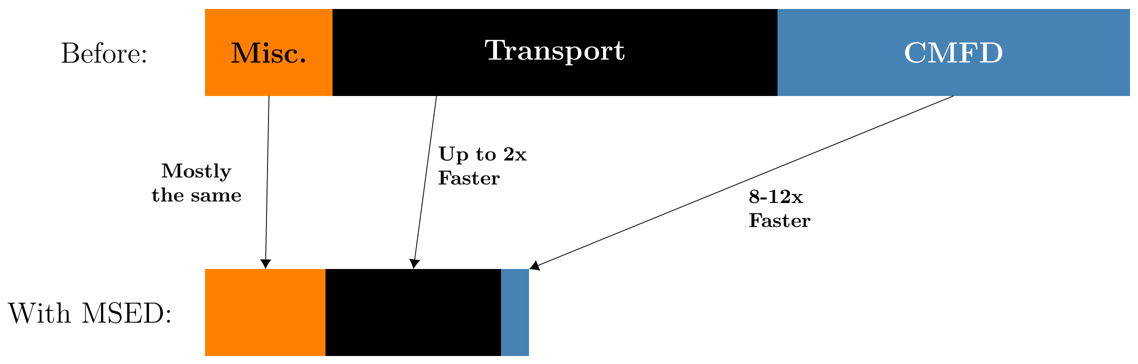


Figure 10.1: A visualization of the overall speedup provided by MSED for single statepoint calculations on *3-D, full-core* reactor models. The relative sizes of the boxes are chosen based on the results for Problem AP1000-BOC in Table 7.10.

In recent decades, reactor physics codes have become increasingly reliant on general, “black-box” solvers for linear systems and eigenvalue problems. These solvers are easy to implement because of their wide availability in linear algebra libraries such as PETSc, but they are also typically far from optimal because of their generality. Unless one carefully constructs a well-informed preconditioner, these “black-box” solvers are agnostic to the geometry and underlying physics of the problem. Understandably, there is a tradeoff dictated by real-world constraints. Though these “black-box” solvers may not be optimal, their performance is often “good enough,” and it is not always worth the time and labor to develop a customized solver. Nonetheless, the work in this thesis shows that, by carefully developing solution methods that are customized to the important geometric and physical features of the specific problem of interest, it is possible to significantly reduce the cost of solving the problem (sometimes by an order of magnitude). For codes such as MPACT that require significant computational resources, the development, implementation, and use of a customized solver like MSED is well worth the effort.

APPENDIX A

A Summary of PETSc Options Used in MPACT

PETSc is a library developed by Argonne National Laboratory that provides efficient data structures and routines for highly parallel linear algebra applications [7]. Both the existing MPACT code and our implementation of MSED in MPACT rely heavily on the tools available in this library. For all problems run on the Eos supercomputer at the Oak Ridge Leadership Computing Facility (OLCF), PETSc version 3.6.3 was used. For the problems run on the Titan supercomputer at the OLCF, PETSc version 3.7.6 was used to generate the results in Chapter 4, and PETSc version 3.6.4 was used to generate the remaining results (Chapters 6, 7, and 8). All problems used the default convergence criteria in PETSc; these convergence criteria vary from solver to solver and are described in [7].

In this subsection, we provide a brief list (with explanations and details where necessary) of the options from PETSc that are used in this thesis.

Linear Solvers:

- KSPGMRES – Standard GMRES solver. A restart length of 100 iterations is used in this thesis.
- KSPBiCGSTAB – Standard BiCGSTAB solver.
- Multigrid – For MSED's spatial multigrid solver, we use the PETSc framework to facilitate the V-cycle. PETSc is provided with the fine-grid matrix, number of levels, smoother options on each level, and all of the interpolation operators needed. More details are provided in Chapter 7. [1]

Smoother:

- PCSOR – This is the smoother option used in conjunction with the multigrid solver. Although its name indicates it is an SOR solver, we set the relaxation parameter to 1. In parallel, it is an iteration scheme that lies somewhere between GS and Jacobi. Information from unknowns

[1]Multigrid is actually listed in PETSc as a preconditioner (denoted PCMG); to use it as a linear solver, it must be paired with the KSPRICHARDSON solver.

residing on other processors are lagged as in Jacobi, but, within each processor, the matrix rows are iterated through sequentially as in GS.

Preconditioners:

- PCBJACOBI – Block Jacobi preconditioner.
- PCILU – ILU preconditioner.

For the Krylov solvers GMRES and BiCGSTAB, we use a block Jacobi preconditioner, with the blocks corresponding to the domain size on each processor. Each block is approximately solved using the approximate LU factorization from PCILU.

More information about these solver options can be found in the PETSc user manual [7].

APPENDIX B

Description of 1-D Diffusion Problems

In this appendix, we describe the three problems used to generate results for the 1-D multigroup diffusion test code described in Section 3.1.

B.1 1-D 1-Group Joo-Lee Problem

This is a 1-D 1-group problem consisting of 216 moderator and fuel pins, with uranium dioxide (UOX), mixed oxide (MOX), and burnable poisons (Gd, B). Each pin is divided into three equally-sized spatial cells, each of which has width $\Delta x = 0.425$ cm. The eigenvalue λ corresponding to the solution of this diffusion problem for this particular spatial discretization is 0.827208.

Although this problem was formulated by Lee and Joo, the full details of this problem are found in a manuscript by Wolters et al. [121]

B.2 1-D Watts Bar Problems

We now describe the WB-1D-1 and WB-1D-2 problems, which are 1-D reactor models derived from the 2D Watts Bar model described in [112] (Problem #5). The descriptions, tables, and figures in this section are taken from [107]. Cross-sections from the MPACT 47-group library (version 3.1, dated November 11th, 2014) [4, 79] are used, and the temperature is assumed to be 600 K. WB-1D-1 is a representation of the center row of pins in the 2D Watts Bar core, while WB-1D-2 is a representation of the row immediately below the center row. We note that they are not exact replicas of the rows, as certain pins and regions were adjusted to create a uniform spatial grid.

Material and isotopic compositions of each pin cell can be found in [112]. All of the pin cells have a pitch of 1.26 cm, and a topical description of the pin cells is provided in Table B.1. Figures B.1 and B.2 describe the WB-1D-1 and WB-1D-2 half-core geometries in terms of modules. These modules are arbitrary constructs used to condense the description of the cores, and Table B.2 provides the composition of each module in terms of the pins listed in Table B.1. A reflective boundary condition is imposed on the left boundary, while a vacuum boundary is imposed on the right boundary.

The WB-1D-1 problem has more nonfissile regions than the WB-1D-2 model has a higher true λ as a result. For $\Delta x = 1.26$ cm (each spatial cell corresponds to a pin), λ is 1.13864 and 0.889732 for WB-1D-1 and WB-1D-2 respectively. For $\Delta x = 6.3$ cm, the same two values are 1.16470 and 0.895094.

Table B.1: Description of pins used in the WB-1D-1 and WB-1D-2 models.

Fuel Pin #	Description
A	2.1%-enriched UO2 fuel pin
B	2.6%-enriched UO2 fuel pin
C	3.1%-enriched UO2 fuel pin
D	Guide tube
E	Pyrex rod
F	Empty instrument tube
G	Water
H	Baffle
I	12%-88% Water-Baffle Mix
J	62%-32% Water-Baffle Mix

1	2	3	4	1	2	3	4	1	2	3	4	1	5	6	7	8

Figure B.1: Module layout in WB-1D-1.

11	12	13	14	11	12	13	14	11	12	13	14	11	15	16	17	18

Figure B.2: Module layout in WB-1D-2.

Table B.2: Pin layouts of each module in the WB-1D-1 and WB-1D-2 models.

Module #	Fuel Pin Composition
1	FAADAADAA
2	GBBEBBEBB
3	FBBEBBEBB
4	GAADAADA
5	GCCDCCECC
6	FCCECCDCC
7	GIHJGGGGG
8	GGGGGGGGG
11	CAAAAAAAA
12	GBBBBBBBB
13	BBBBBBBBB
14	GAAAAAAA
15	GCCCCCCCC
16	CCCCCCCCC
17	GIHJGGGGG
18	GGGGGGGGG

APPENDIX C

Description of MPACT Problems

In this Appendix, we name and describe the problems simulated by MPACT code for the results throughout this thesis. All of the problems except for the peach bottom problem and the AP1000 problems are taken from the VERA progression problems documented in [112]. The model for the peach bottom problem is described in [122], while the AP1000 models are described by [123]. Except for Problem 5a-2D, all problems are simulated using the 51-group MPACT library. The problem 5a-2D problem is simulated using the 252-group library.

Tables C.1, C.2, and C.3 provide a tabulated summary of the features of each problem. Included in these tables are references to the tables in this thesis that contain the corresponding performance results.

- **Problem 2a**: This is a single 17×17 fuel assembly. Its geometry was primarily used in this thesis as a means of generating relevant cross sections for the Fourier analyses in Chapters 6 and 7.

- **Problem 4a**: This is a criticality problem from the VERA Benchmark Progression Problems [112] consists of a 3-D model with nine 17×17 Westinghouse fuel assemblies.

- **Problem 5a-0**: This is a criticality problem from the VERA Benchmark Progression Problems [112] that consists of a quarter-core 3-D model of the Watts Bar Unit 1 reactor. It is listed as case 0 in [112].

- **Problem 5a-2D**: This is the mid-plane of problem 5a-0.

- **Problem 5a-ARO**: This is a criticality problem using a quarter-core 3-D model of the Watts Bar Unit 1 reactor with all rods ejected (ARO stands for all rods out). It is listed as case 11 in [112].

- **Problem 5a-0-FC**: This is a full-core model of problem 5a-0.

- **Problem 7**: This is a critical boron search problem from the VERA Benchmark Progression Problems [112] on a quarter-core 3-D model of the Watts Bar Unit 1 reactor. Thermohydraulic and xenon feedback is present in this problem. The critical boron concentration for this problem is 827.6 ppm when the 47-group library is used and 829.4 ppm when the 51-group library is used.

- **Problem 9**: This is a 32-state depletion problem from the VERA Benchmark Progression Problems [112] on a quarter-core 3-D model of the Watts Bar Unit 1 reactor. Thermohydraulic and xenon feedback is present in this problem.

- **AP1000-BOC**: This is a 3-D full-core model of the AP1000 reactor at hot-zero-power, with beginning of cycle (BOC) fuel and material compositions. No feedback is considered in this problem.

- **AP1000-EOC**: This is a 3-D full-core model of the AP1000 reactor at hot-zero-power, with end of cycle (EOC) fuel and material compositions. Feedback is present in this problem, but plays an insignificant role because of the near-zero power level.

- **AP1000-EOC-HFP**: This is a 3-D full-core model of the AP1000 reactor at hot-zero-power, with end of cycle (EOC) fuel and material compositions. Feedback is present in this problem and plays a significant role in its solution.

- **PB**: This is a criticality problem using a 2-D full core model of the Peach Bottom (PB) boiling water reactor (BWR) at the beginning of life [122].

The following list describes (in no particular order) all of the alterations made to the input files used to generate all of the MPACT results in this thesis. These tweaks do not detract or change the conclusions of this thesis; they are presented here to improve the reproducibility of the work in this thesis.

- When the one-group approach (Algorithm 11) is used with MED, the number of GMRES iterations required is reduced from 100 to 25. This is because an unshifted one-group diffusion problem is very easy to solve.

- For the "reduced" approach (Algorithm 10), the relative residual tolerance for the CMFD solver in MPACT is loosened from 0.01 to 0.05. Without this loosening, it converges to a much more tightly converged CMFD problem since the linear solver iterations performed on a reduced system are much more effective than those performed on a full multigroup system.

- In the AP1000 problems and the PB problem, “one-group” data passing is used instead of “multigroup” data passing in the MPACT input. More information about what this means can be found in the MPACT user manual [4].

- For problems AP1000-EOC, AP1000-EOC-HFP, 4a, 5a-0 (the runs on Titan only), 5a-2D, 7, and 9, only 10 GMRES iterations were used on the coarsest spatial grid for the multigrid solver on the grey problem in MSED. (We were experimenting with this, and it turns out that we can use 10 instead of 15 without any drawbacks in the overall convergence rate of multigrid. The other results, which use 15 instead of 10, have marginally higher GR runtimes than optimal.)

- In MPACT, c_0 is 0.02 instead of 0.01. The tradeoff between the number of PIs and the conditioning of the shifted diffusion operator (described in Chapter 4) is worse for large problems such as those in MPACT.

- In Problem 5a-0 and 5a-0-FC, the outer (transport) convergence criterion on the eigenvalue is increased slightly from 10^{-6} to $1.1 \cdot 10^{-6}$. The relative error in the eigenvalue is slightly above 10^{-6} after 12 transport sweeps, and we feel that it would be a better representation to just stop there instead of doing another transport sweep.

- The first 28 hours of problem 9 (i.e., up until the first restart) was run using ORIGEN as the depletion solver. However, this solver crashed in during the second 28 hours of problem 9 for the default method. For consistency, both methods switched to an internal depletion solver in MPACT after the first restart.

- In the AP1000-EOC-HFP problem and problem 9, 2 smoother iterations are performed instead of 1 for the multigrid solver on the multigroup problem. Because we are using GS (Algorithm 13) instead of RBBJ (Algorithm 12) as the smoother, the multigrid method in MPACT is not as robust as the one in the 1-D diffusion code. Sometimes, extra smoother iterations are needed to account for this.

Table C.1: Runtime details for various MPACT problems. The machines used are provided, along with the number of processors and nodes and the relevant SHA-1 number(s) that can be used to identify the version of MPACT used to generate the results. If two SHA-1 numbers are provided, the first is the SHA-1 number of the commit on the master branch from which the current branch is forked, and the latter is the SHA-1 number corresponding to the most recent commit of the forked branch. The table numbers of the corresponding performance results are provided in the final column.

Problem	Machine	Nodes	Cores	SHA-1 Number(s)	Table(s) in Thesis
4a	Eos	4	58	`5e1111`, `a81bec`	6.9, 7.5
5a-0	Eos	29	464	`5e1111`, `a81bec`	6.5, 6.7, 6.11, 7.7
5a-0	Titan	58	464	`5e1111`, `a81bec`	8.1, 8.2
5a-0-2D	Eos	4	32	`5e1111`, `a81bec`	6.6, 6.8, 6.12, 7.8
5a-0-ARO	Titan	58	464	`03ec0e`	4.6, 4.8
5a-0-FC	Titan	464	3712	`5e1111`, `a81bec`	6.13, 7.9
7	Titan	58	464	`03ec0e`	4.7
7	Titan	58	464	`5e1111`, `a81bec`	8.3
9	Titan	58	464	`5e1111`, `a81bec`	8.5
AP1000-BOC	Titan	448	3584	`5e1111`, `a81bec`	6.14, 7.10
AP1000-EOC	Titan	448	3584	`5e1111`, `a81bec`	6.15, 7.11
AP1000-EOC-HFP	Titan	448	3584	`5e1111`, `a81bec`	8.4
PB	Eos	15	232	`5e1111`, `a81bec`	6.10, 7.6

Table C.2: Physics-related details for various MPACT problems. Here, we tabulate the type of the problems, whether or not feedback is present, the number of groups, and the final k_{eff}. The table numbers of the corresponding performance results are provided in the final column. The quotes on the AP1000-EOC line indicate that, although feedback is present for this problem, it plays an insignificant role.

Problem	Problem Type	Feedback?	G	k_{eff}	Table(s) in Thesis
4a	Criticality	No	51	0.998177	6.9, 7.5
5a-0	Criticality	No	51	0.999259	6.5, 6.7, 6.11, 7.7, 8.1, 8.2
5a-0-2D	Criticality	No	252	0.998142	6.6, 6.8, 6.12, 7.8
5a-0-ARO	Criticality	No	47	1.012136	4.6, 4.8
5a-0-FC	Criticality	No	51	0.999259	6.13, 7.9
7	Critical Boron	Yes	47	1.000000	4.7
7	Critical Boron	Yes	51	1.000000	8.3
9	Depletion, 32 States	Yes	51	N/A	8.5
AP1000-BOC	Criticality	No	51	1.123087	6.14, 7.10
AP1000-EOC	Criticality	"Yes"	51	0.989179	6.15, 7.11
AP1000-EOC-HFP	Criticality	Yes	51	0.953181	8.4
PB	Criticality	No	51	1.042273	6.10, 7.6

Table C.3: Geometry-related details for various MPACT problems. The table numbers of the corresponding performance results are provided in the final column.

Problem	# Dims.	Assembly Layout	Table(s) in Thesis
4a	3-D	3×3	6.9, 7.5
5a-0	3-D	Quarter-Core	6.5, 6.7, 6.11, 7.7, 8.1, 8.2
5a-0-2D	2-D	Quarter-Core	6.6, 6.8, 6.12, 7.8
5a-0-ARO	3-D	Quarter-Core	4.6, 4.8
5a-0-FC	3-D	Full-Core	6.13, 7.9
7	3-D	Quarter-Core	4.7, 8.3
9	3-D	Quarter-Core	8.5
AP1000-BOC	3-D	Full-Core	6.14, 7.10
AP1000-EOC	3-D	Full-Core	6.15, 7.11
AP1000-EOC-HFP	3-D	Full-Core	8.4
PB	2-D	Full-Core	6.10, 7.6

APPENDIX D

Alternate Grey Diffusion Equation

D.1 Formulation of the Alternate Grey Diffusion Equation

In Chapter 6, we discussed the development of a grey diffusion equation, obtained by summing the multigroup diffusion equations over the groups g. In this section, we consider an alternative collapse in which the multigroup diffusion equations are first multiplied by a space-and-energy independent function $f_{j,g}$ before the summation over g.

For a multigroup CMFD eigenvalue problem of the form

$$-\frac{1}{\Delta x_j}\left[D_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}-\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}}-D_{j-\frac{1}{2},g}\frac{\phi_{j,g}-\phi_{j-1,g}}{\Delta x_{j-\frac{1}{2}}}\right]$$
$$+\frac{1}{\Delta x_j}\left[\hat{D}_{j+\frac{1}{2},g}\frac{\phi_{j+1,g}+\phi_{j,g}}{\Delta x_{j+\frac{1}{2}}}+\hat{D}_{j-\frac{1}{2},g}\frac{\phi_{j,g}+\phi_{j-1,g}}{\Delta x_{j+\frac{1}{2}}}\right]$$
$$+\Sigma_{t,j,g}\phi_{j,g}-\sum_{g'=1}^{G}\Sigma_{s0,j,g'\to g}\phi_{j,g'}=\lambda\chi_{j,g}\sum_{g'=1}^{G}\nu\Sigma_{f,j,g'}\phi_{j,g'}\,, \quad \text{(6.1 revisited)}$$

a collapse in energy weighted by $f_{j,g}$ yields

$$-\left[\frac{\left\langle fD_{j+\frac{1}{2},j+1}\right\rangle}{\Delta x_j\Delta x_{j+\frac{1}{2}}}\right]\Phi_{j+1}-\left[\frac{\left\langle fD_{j-\frac{1}{2},j-1}\right\rangle}{\Delta x_j\Delta x_{j-\frac{1}{2}}}\right]\Phi_{j-1}+$$
$$\left[\frac{\left\langle fD_{j+\frac{1}{2},j}\right\rangle}{\Delta x_j\Delta x_{j+\frac{1}{2}}}+\frac{\left\langle fD_{j-\frac{1}{2},j}\right\rangle}{\Delta x_j\Delta x_{j-\frac{1}{2}}}+\langle f\Sigma_{a,j}\rangle\right]\Phi_j=\lambda\left\langle\nu\Sigma_{f,j}\right\rangle\Phi_j\,. \quad \text{(D.1)}$$

Here,

$$\Phi_j\equiv\sum_{g=1}^{G}\phi_{j,g}\,, \quad \text{(D.2a)}$$

$$\langle f\Sigma_{t,j}\rangle \equiv \frac{1}{\Phi_j}\sum_{g=1}^{G} f_{j,g}\Sigma_{t,j,g}\phi_{j,g}\,, \tag{D.2b}$$

$$\langle f\Sigma_{s0,j}\rangle \equiv \frac{1}{\Phi_j}\sum_{g'=1}^{G}\sum_{g=1}^{G} f_{j,g}\Sigma_{s0,j,g'\to g}\phi_{j,g'}\,, \tag{D.2c}$$

$$\langle \nu\Sigma_{f,j}\rangle \equiv \frac{1}{\Phi_j}\sum_{g=1}^{G} \nu\Sigma_{f,j,g}\phi_{j,g}\,, \tag{D.2d}$$

$$\langle fD_{j,j_1,j_2}\rangle \equiv \begin{cases} \frac{1}{\Phi_{j_2}}\sum\limits_{g=1}^{G} f_{j,g}\left(D_{j_1,g}-\hat{D}_{j_1,g}\right)\phi_{j_2,g}\,, & \text{if } j_2 = j \text{ (if } j_2 \text{ is the current cell)}\,, \\ \frac{1}{\Phi_{j_2}}\sum\limits_{g=1}^{G} f_{j,g}\left(D_{j_1,g}+\hat{D}_{j_1,g}\right)\phi_{j_2,g}\,, & \text{if } j_2 \neq j \text{ (if } j_2 \text{ is a neighbor)}\,. \end{cases} \tag{D.2e}$$

The definitions of Φ and $\langle\nu\Sigma_f\rangle$ for the weighted collapse here are the same as those for the standard collapse in Equations (6.3). In the above collapse, we have assumed that $f_{j,g}$ is normalized at each fissile spatial cell so that

$$\sum_{g=1}^{G}\chi_{j,g}f_{j,g} = 1\,. \tag{D.3}$$

We have not yet specified $f_{j,g}$, but our analysis below shows that it may be beneficial to define $f_{j,g}$ as the solution to the infinite-medium adjoint eigenvalue problem at that spatial cell. It is not clear what should be done at non-fissile spatial cells, but the simple option would be to choose $f_{j,g} = 1$ so that it is the same as the standard collapse.

D.2 Fourier Analysis

To gain a sense of how $f_{j,g}$ should or could be defined, we perform a Fourier analysis with arbitrary (but space-independent) weights f_g. This analysis is very similar to the Fourier analysis performed in Section 6.2 with $C = 1$ for the MED method (C is the number of groups on the coarsest energy grid), and much of the algebra will be omitted here for brevity.

As usual, we begin with our Fourier ansatz:

$$\phi_{j,g}^{(l)} = \varphi_g + \epsilon a_g\omega^l e^{i\xi j\Delta x}\,, \tag{D.4a}$$

$$\Phi_j^{(l)} = 1 + \epsilon A\omega^l e^{i\xi j\Delta x}\,, \tag{D.4b}$$

$$\lambda^{(l)} = \lambda_0 + \epsilon\lambda_1\omega^l\,, \tag{D.4c}$$

$$\phi_{j,g}^{(l')} = \varphi_g + \epsilon a'_g\omega^l e^{i\xi j\Delta x}\,, \tag{D.4d}$$

$$\Phi_j^{(l')} = 1 + \epsilon A'\omega^l e^{i\xi j\Delta x}\,, \tag{D.4e}$$

$$\lambda^{(l')} = \lambda_0 + \epsilon\lambda'_1\omega^l\,, \tag{D.4f}$$

$$\phi_{j,g}^{(l'')} = \varphi_g + \epsilon a_g'' \omega^{l+1} e^{i\xi j \Delta x} \,. \tag{D.4g}$$

The assumptions and variable definitions here are the same as those in Section 6.2. For the reasons described in Sections 4.2 and 6.2, we have

$$\lambda_1 = 0 \,, \tag{D.5}$$

$$\lambda_1' = 0 \,, \tag{D.6}$$

$$\phi_{j,g}^{(l+1)} = \phi_{j,g}^{(l'')} \,, \tag{D.7}$$

$$\omega a_g = a_g'' \,, \tag{D.8}$$

$$A = \sum_{g=1}^{G} a_g \,. \tag{D.9}$$

Following the same procedure used to derive Equation (6.29) in Section 6.2, we arrive at the following analogous result:

$$\left[\frac{4\overline{fD}}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)\right](A'-A) + \left[\overline{f\Sigma}_a - \lambda_0\overline{\nu\Sigma}_f\right](A'-A) = -\frac{4}{\Delta x^2}\sin^2\left(\frac{\xi\Delta x}{2}\right)\sum_{g=1}^{G} D_g a_g$$
$$- \sum_{g=1}^{G} f_g \Sigma_{t,g} a_{g'} + \sum_{g'=1}^{G}\sum_{g=1}^{G}\left[\lambda_0 f_g \chi_g \nu\Sigma_{f,g'} + f_g \Sigma_{s0,j,g'\to g}\right] a_{g'} \,. \tag{D.10}$$

Here, overbar-ed quantities of the form $\overline{f\Sigma}$ are obtained by summation with weights $f_g\varphi_g$, while overbar-ed quantities of the form $\overline{\Sigma}$ are obtained by summation with weights φ_g. For example,

$$\overline{f\Sigma}_t \equiv \sum_{g=1}^{G} f_g D_g \varphi_g \,, \tag{D.11}$$

$$\overline{\nu\Sigma}_f \equiv \sum_{g=1}^{G} \nu\Sigma_{f,g}\varphi_g \,. \tag{D.12}$$

Next, we consider the infinite-medium diffusion equation with its true solution φ_g:

$$\Sigma_{t,g}\varphi_g - \sum_{g'=1}^{G} \Sigma_{s0,g'\to g}\varphi_{g'} = \lambda\chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'}\varphi_{g'} \,. \tag{D.13}$$

When we multiply the above equation by f_g, sum it over all groups g, and use the normalization condition in Equation (D.3), we obtain

$$\overline{f\Sigma}_a = \lambda_0 \overline{\nu\Sigma}_f \,. \tag{D.14}$$

Finally, we note that

$$\begin{aligned}
&\sum_{g=1}^{G} f_g \Sigma_{t,g} - \sum_{g'=1}^{G}\sum_{g=1}^{G} \left[\lambda_0 f_g \chi_g \nu\Sigma_{f,g'} + f_g \Sigma_{s0,j,g'\to g}\right] a_{g'} \\
&\quad = \sum_{g=1}^{G} f_g \Sigma_{t,g} a_g - \sum_{g=1}^{G} \lambda_0 f_g \chi_g \sum_{g'=1}^{G} \nu\Sigma_{f,g'} a_{g'} + \sum_{g=1}^{G}\sum_{g'=1}^{G} f_g \Sigma_{s0,j,g'\to g} a_{g'} \\
&\quad = \sum_{g=1}^{G} f_g \Sigma_{t,g} a_g - \sum_{g'=1}^{G} \lambda_0 f_{g'} \chi_{g'} \sum_{g=1}^{G} \nu\Sigma_{f,g} a_g + \sum_{g=1}^{G}\sum_{g'=1}^{G} f_{g'} \Sigma_{s0,j,g\to g'} a_g \\
&\quad = \sum_{g=1}^{G} \left\{ f_g \Sigma_{t,g} - \sum_{g'=1}^{G} \lambda_0 \nu\Sigma_{f,g} f_{g'} \chi_{g'} + \sum_{g'=1}^{G} f_{g'} \Sigma_{s0,j,g\to g'} \right\} a_g \,.
\end{aligned} \tag{D.15}$$

The statement in the curly braces is an expression of the infinite-medium adjoint eigenvalue problem. If we choose f_g to be an adjoint eigenvector, then the entire expression equals zero. By doing this, and then making use of use of Equation (D.14), Equation (D.10) simplifies to

$$\overline{fD}\,(A' - A) = \sum_{g=1}^{G} D_g a_g \,. \tag{D.16}$$

A similar result can be obtained for the $C > 1$ case if $\langle\chi_1\rangle = 1$ and $\langle\chi_c\rangle = 0$ for all other c. However, even in that case, the result is not as simple as Equation (D.16) because Equation (D.14) cannot be used to simplify the system for $C > 1$. It is also unclear whether there would be any significant reduction in the spectral radius when an adjoint-weighted collapse is used for $C > 1$.[1]

From this point, the analysis can be continued in the same manner as in Section 6.2. The only difference would be in the definitions of M_1 and M_2 in Equation (6.30). For brevity, we omit the details and proceed to the numerical results.

[1]We have not looked into adjoint-weighted collapses for $C > 1$ extensively; this is just our speculation.

D.3 Numerical Results and Discussion

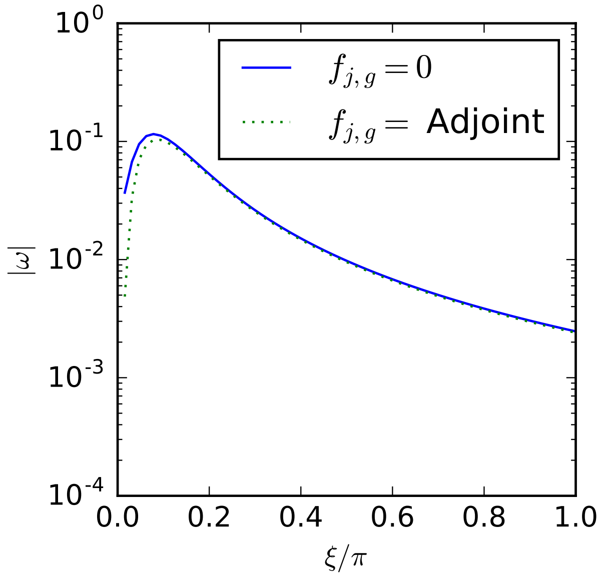
Here, we repeat the computations of Section 6.2 for the alternate (adjoint-weighted) grey diffusion equation to obtain decay rates and spectral radii. Plots of the decay rates are shown in Figure D.1, while the spectral radii are reported for each cross section set in Table D.1.

From these results, we see that the difference between using the standard grey equation and the adjoint-weighted grey equation is very small. To better contextualize the difference, we consider a problem in which the convergence tolerance is 10^{-8}. In this case, it would take 11 iterations to converge with the standard collapse and 10 iterations with the adjoint-weighted collapse. Moreover, any reductions in the number of iterations required is partially offset by the cost of computing $f_{j,g}$ (the infinite-medium adjoint spectrum) in each spatial cell.

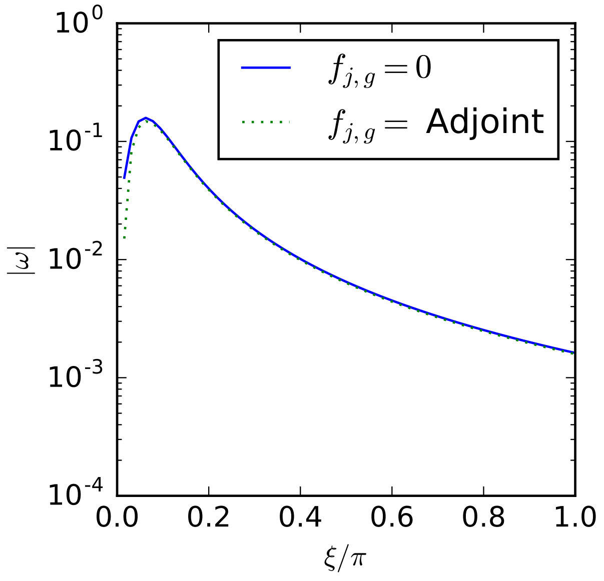
Table D.1: Spectral radii computed from Fourier analysis of MED with the alternate grey diffusion equation described by Appendix D. These spectral radii are computed by taking the maximum magnitude of the decay factors $|\omega|$ shown in Figure 6.1.

Library	$f_{j,g} = 1$	$f_{j,g} =$ Infinite Medium Adjoint
MPACT 8G	0.115	0.103
MPACT 47G	0.158	0.148
MPACT 51G	0.162	0.152
MPACT 252G	0.164	0.152

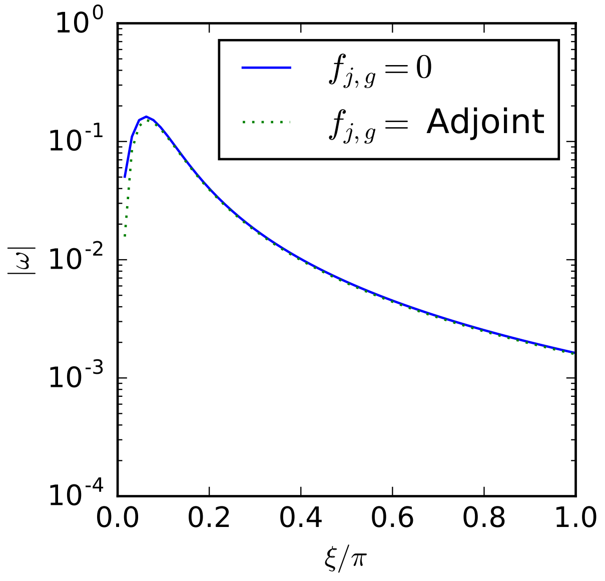
One feature of the adjoint-weighted collapse is that the resulting MED spectral radius goes to 0 as $\xi \to 0$. This is not true of the standard collapse. In this limit, it is possible to formulate toy problems in which the spectral radius of the standard collapse exceeds 1. However, we have not found any that are remotely representative of actual reactor problems. For example, the standard collapse results is unstable ($\rho > 1$) in a 2-group ($G = 2$) problem with a large fast-group absorption cross section, a low fast-group fission cross section, and a relatively small down-scatter cross section. (This would not be a good reactor design; k_{eff} would be much too low since many neutrons are captured before they have a chance to reach thermal energies.) As we have noted earlier, we have not found any practical problems for which this alternate, adjoint-weighted collapse has tangible benefits.



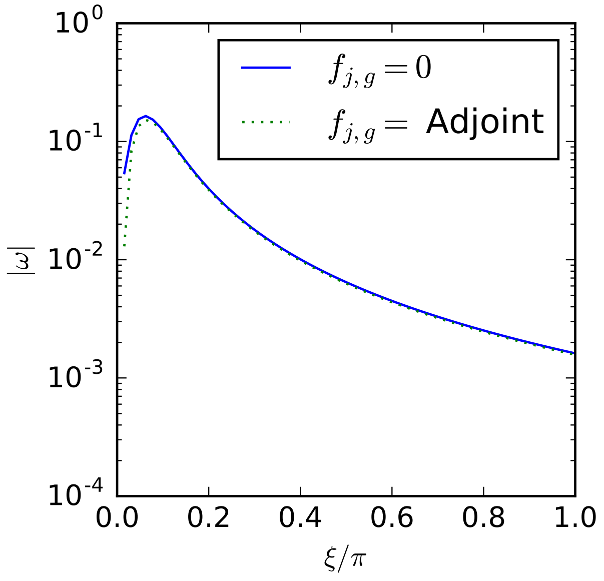
(a) MPACT 8G library.



(b) MPACT 47G library.



(c) MPACT 51G library.



(d) MPACT 252G library.

Figure D.1: Decay factors from Fourier analysis of MED with the alternate grey diffusion equation described by Appendix D. The parameters used to generate these plots are the same as those used to generate Figure 6.1.

BIBLIOGRAPHY

[1] K.S. Smith, "Assembly Homogenization Techniques for Light Water Reactor Analysis," *Progress in Nuclear Energy*, Vol. 17, No. 3, 1986, pp. 303–335.

[2] R.E. Alcouffe, "Diffusion Synthetic Acceleration Methods for the Diamond-Differenced Discrete-Ordinates Equations," *Nuclear Science and Engineering*, Vol. 64, No. 2, 1977, pp. 344–355.

[3] K.S. Smith and J.D. Rhodes III, "Full-Core, 2-D, LWR Core Calculations With CASMO-4E," *PHYSOR*, October 7–10, Seoul, 2002.

[4] MPACT Team, "MPACT Theory Manual Version 2.0.0," Tech. Rep. CASL-U-2015-0078-000, Consortium for Advanced Simulation of LWRs, 2015.

[5] A. Zhu, M. Jarrett, Y. Xu, B. Kochunas, E. Larsen, and T. Downar, "An Optimally Diffusive Coarse Mesh Finite Difference Method to Accelerate Neutron Transport Calculations," *Annals of Nuclear Energy*, Vol. 95, 2016, pp. 116–124.

[6] B. Kochunas, S. Stimpson, Y. Liu, B.C. Yee, A. Zhu, A. Fitzgerald, A. Graham, M. Jarrett, B. Collins, K.S. Kim, and K. Clarno, "MPACT Performance Improvements in VERA-CS," Tech. Rep. CASL-U-2016-1115-000, Oak Ridge National Laboratory, 2016.

[7] S. Balay, S. Abhyankar, M.F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, K. Rupp, B.F. Smith, S. Zampini, H. Zhang, and H. Zhang, "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 3.7, Argonne National Laboratory, 2016.

[8] M. Heroux, R. Bartlett, V.H.R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams, "An Overview of Trilinos," Tech. Rep. SAND2003-2927, Sandia National Laboratories, 2003.

[9] W.L. Briggs, V.E. Henson, and S.F. McCormick, *A Multigrid Tutorial*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2000.

[10] B.C. Yee, B. Kochunas, and E.W. Larsen, "A Multilevel in Space and Energy Solver for Multigroup Diffusion Eigenvalue Problems," *Nuclear Engineering and Technology*, Vol. 49, No. 6, 2017, pp. 1125–1134.

[11] B.C. Yee, B. Kochunas, and E.W. Larsen, "A Multilevel Diffusion Solver for Multi-Dimensional Transport Problems With CMFD," *PHYSOR*, April 22–26, Cancun, Mexico, 2018.

[12] W.A. Strauss, *Partial Differential Equations*, John Wiley & Sons New York, NY, USA, 1992.

[13] Y. Saad, *Iterative Methods for Sparse Linear Systems*, Vol. 82, SIAM, 2003.

[14] S.F. McCormick, *Multigrid Methods*, SIAM, 1987.

[15] U. Trottenberg, C.W. Oosterlee, and A. Schuller, *Multigrid*, Academic Press, 2000.

[16] I. Yavneh, "Why Multigrid Methods Are So Efficient," *Computing in science & engineering*, Vol. 8, No. 6, 2006, pp. 12–22.

[17] J.W. Ruge and K. Stüben, "Algebraic Multigrid," *Multigrid Methods*, SIAM, 1987, pp. 73–130.

[18] A. Brandt, S. McCoruick, and J. Huge, "Algebraic Multigrid (AMG) for Sparse Matrix Equations," *Sparsity and its Applications*, Vol. 257, 1985.

[19] K. Stüben, "A Review of Algebraic Multigrid," *Partial Differential Equations*, Elsevier, 2001, pp. 281–309.

[20] U.M. Yang et al., "BoomerAMG: A Parallel Algebraic Multigrid Solver and Preconditioner," *Applied Numerical Mathematics*, Vol. 41, No. 1, 2002, pp. 155–177.

[21] A. Brandt, "Multigrid Solvers on Parallel Computers," *Elliptic Problem Solvers*, Elsevier, 1981, pp. 39–83.

[22] A. Brandt, "Barriers to Achieving Textbook Multigrid Efficiency (TME) in CFD," Tech. Rep. NASA/CR- 1998-207647, ICASE Interim Report No. 32, NASA, 1998.

[23] B. Gmeiner, U. Rüde, H. Stengel, C. Waluga, and B. Wohlmuth, "Towards Textbook Efficiency for Parallel Multigrid," *Numerical Mathematics: Theory, Methods and Applications*, Vol. 8, No. 1, 2015, pp. 22–46.

[24] P. Bastian, G. Wittum, and W. Hackbusch, "Additive and Multiplicative Multi-Grid – A Comparison," *Computing*, Vol. 60, No. 4, 1998, pp. 345–364.

[25] M. Adams, M. Brezina, J. Hu, and R. Tuminaro, "Parallel Multigrid Smoothing: Polynomial Versus Gauss–Seidel," *Journal of Computational Physics*, Vol. 188, No. 2, 2003, pp. 593–610.

[26] A.H. Baker, R.D. Falgout, T.V. Kolev, and U.M. Yang, "Scaling Hypre's Multigrid Solvers to 100,000 Cores," *High-Performance Scientific Computing*, Springer, 2012, pp. 261–279.

[27] E.H. Müller, R. Scheichl, B. Muite, and E. Vainikko, "Petascale Elliptic Solvers for Anisotropic PDEs on GPU Clusters," *CoRR*, Vol. abs/1402.3545, 2014.

[28] C.C. Douglas, G. Haase, and U. Langer, "Multigrid Methods," *A Tutorial on Elliptic PDE Solvers and Their Parallelization*, Vol. 16, Chap. 7, SIAM, 2003, pp. 111–118.

[29] E. Chow and A. Patel, “Fine-Grained Parallel Incomplete LU Factorization,” *SIAM Journal on Scientific Computing*, Vol. 37, No. 2, 2015, pp. C169–C193.

[30] T.J. Downar and H.G. Joo, “A Preconditioned Krylov Method for Solution of the Multi-Dimensional, Two Fluid Hydrodynamics Equations,” *Annals of Nuclear Energy*, Vol. 28, No. 12, 2001, pp. 1251–1267.

[31] D. Knoll, H. Park, and K. Smith, “Application of the Jacobian-Free Newton-Krylov Method in Computational Reactor Physics,” *American Nuclear Society 2009 International Conference on Advances in Mathematics, Computational Methods, and Reactor Physics, Saratoga Springs, NY*, 2009.

[32] B. Collins, S. Hamilton, and S. Stimpson, “Use of Generalized Davidson Eigenvalue Solver for Coarse Mesh Finite Difference Acceleration,” *M&C 2017 - International Conference on Mathematics & Computational Methods Applied to Nuclear Science & Engineering*, April 16 – 20, Jeju, Korea, 2017.

[33] R.E. Alcouffe, “The Multigrid Method for Solving the Two-Dimensional Multigroup Diffusion Equation,” *Advances in Reactor Computations, Salt Lake City*, 1983, pp. 340–351.

[34] R.E. Alcouffe, A. Brandt, J.E. Dendy, Jr, and J.W. Painter, “The Multi-Grid Method for the Diffusion Equation With Strongly Discontinuous Coefficients,” *SIAM Journal on Scientific and Statistical Computing*, Vol. 2, No. 4, 1981, pp. 430–454.

[35] J.E. Dendy, “Black Box Multigrid,” *Journal of Computational Physics*, Vol. 48, No. 3, 1982, pp. 366–386.

[36] M.H. Protter and H.F. Weinberger, *Maximum Principles in Differential Equations*, Springer Science & Business Media, 2012.

[37] E.L. Wachspress, *Iterative Solution of Elliptic Systems*, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1966.

[38] R. FROEHLICH, “A Theoretical Foundation for Coarse Mesh Variation Techniques,” Tech. Rep. GA-7870, General Atomic, 1967.

[39] K.D. Lathrop and F.W. Brinkley, “TWOTRAN-II: An Interfaced, Exportable Version of the TWOTRAN Code for Two-Dimensional Transport,” Tech. rep., Los Alamos Scientific Lab., N. Mex.(USA), 1973.

[40] G.R. Cefus and E.W. Larsen, “Stability Analysis of Coarse-Mesh Rebalance,” *Nuclear Science and Engineering*, Vol. 105, No. 1, 1990, pp. 31–39.

[41] N.Z. Cho and C.J. Park, “A Comparison of Coarse Mesh Rebalance and Coarse Mesh Finite Difference Accelerations for the Neutron Transport Calculations,” *Proc. M&C 2003*, 2003, pp. 6–11.

[42] R. Van Geemert, “Synergies of Acceleration Methodologies for Whole-Core N/TH-coupled Steady-State and Transient Computations,” *Proceedings on the International Conference on the Physics of Reactors (PHYSOR 2006)*, 2006.

[43] R. van Geemert, “A Multi-Level Surface Rebalancing Approach for Efficient Convergence Acceleration of 3D Full Core Multi-Group Fine Grid Nodal Diffusion Iterations,” *Annals of Nuclear Energy*, Vol. 63, 2014, pp. 22–37.

[44] H. Finnemann, R. Boeer, R. Mueller, and Y.I. Kim, “Adaptive Multi-Level Techniques for the Solution of Nodal Transport Equations,” *Proceedings of the Third European Conference on Multigrid Methods*, Bonn, Germany, 2006.

[45] A. Yamamoto, “Generalized Coarse-Mesh Rebalance Method for Acceleration of Neutron Transport Calculations,” *Nuclear science and engineering*, Vol. 151, No. 3, 2005, pp. 274–282.

[46] M.L. Adams and W.R. Martin, “Diffusion Synthetic Acceleration of Discontinuous Finite Element Transport Iterations,” *Nuclear science and engineering*, Vol. 111, No. 2, 1992, pp. 145–167.

[47] Y. Azmy and E.W. Larsen, “Fourier Analysis of the Diffusion Synthetic Acceleration Method for Weighted Diamond-Differencing Schemes in Cartesian Geometries,” *Nuclear Science and Engineering*, Vol. 95, No. 2, 1987, pp. 106–115.

[48] E.W. Larsen, “Diffusion-Synthetic Acceleration Methods for Discrete-Ordinates Problems,” *Transport Theory and Statistical Physics*, Vol. 13, No. 1-2, 1984, pp. 107–126.

[49] R.E. Alcouffe, “Stable Diffusion Synthetic Acceleration Method for Neutron Transport Iterations,” *Transactions of the American Nuclear Society*, Vol. 23, 1976.

[50] E.W. Larsen, “Unconditionally Stable Diffusion-Synthetic Acceleration Methods for the Slab Geometry Discrete Ordinates Equations. Part I: Theory,” *Nuclear Science and Engineering*, Vol. 82, No. 1, 1982, pp. 47–63.

[51] D.R. McCoy and E.W. Larsen, “Unconditionally Stable Diffusion-Synthetic Acceleration Methods for the Slab Geometry Discrete Ordinates Equations. Part II: Numerical Results,” *Nuclear Science and Engineering*, Vol. 82, No. 1, 1982, pp. 64–70.

[52] M.L. Adams and E.W. Larsen, “Fast Iterative Methods for Discrete-Ordinates Particle Transport Calculations,” *Progress in nuclear energy*, Vol. 40, No. 1, 2002, pp. 3–159.

[53] N.Z. Cho, “Fundamentals and Recent Developments of Reactor Physics Methods,” *Nuclear Engineering and Technology*, Vol. 37, No. 1, 2005, pp. 25–78.

[54] T.M. Sutton, “NODEX: A High Order NEM-Based Multigroup Nodal Code,” *Proc. Topical Meeting on Advances in Nuclear Engineering Computations and Radiation Shielding*, Vol. 38:1-3:11, April 9 – 13, Santa Fe, New Mexico, 1989.

[55] J.Y. Cho, H.G. Joo, and K.S. Kim, “Cell Based CMFD Formulation for Acceleration of Whole-Core Method of Characteristics Calculation,” *Journal of the Korean Nuclear Society*, Vol. 34, No. 2, 2002, pp. 250 – 258.

[56] M.J. Lee, H.G. Joo, D. Lee, and K. Smith, "Coarse mesh finite difference formulation for accelerated Monte Carlo eigenvalue calculation," *Annals of Nuclear Energy*, Vol. 65, 2014, pp. 101 – 113.

[57] E.W. Larsen and B.W. Kelley, "The Relationship Between the Coarse-Mesh Finite Difference and the Coarse-Mesh Diffusion Synthetic Acceleration Methods," *Nuclear Science and Engineering*, Vol. 178, No. 1, 2014, pp. 1–15.

[58] N.Z. Cho, G.S. Lee, and C.J. Park, "Partial Current-Based CMFD Acceleration of the 2D/1D Fusion Method for 3D Whole-Core Transport Calculations," *TRANSACTIONS-AMERICAN NUCLEAR SOCIETY*, 2002, pp. 594–596.

[59] M. Jarrett, B. Kelley, B. Kochunas, T. Downar, and E. Larsen, "Stabilization Methods for CMFD Acceleration," *Proc. M&C*, American Nuclear Society, Nashville, Tennessee, 2015.

[60] M. Jarrett, B. Kochunas, A. Zhu, and T. Downar, "Analysis of Stabilization Techniques for CMFD Acceleration of Neutron Transport Problems," *Nuclear Science and Engineering*, Vol. 184, No. 2, 2016, pp. 208–227.

[61] A. Zhu, B. Kochunas, Y. Xu, M. Jarrett, E. Larsen, and T. Downar, "Theoretical Convergence Rate Lower Bounds for Variants of Coarse Mesh Finite Difference to Accelerate Neutron Transport Calculations," *Nuclear Science and Engineering*, Vol. 186, No. 3, 2017, pp. 224–238.

[62] J.I. Yoon and H.G. Joo, "Two-Level Coarse Mesh Finite Difference Formulation With Multigroup Source Expansion Nodal Kernels," *Journal of Nuclear Science and Technology*, Vol. 45, No. 7, 2008, pp. 668–682.

[63] Z. Zhong, T.J. Downar, Y. Xu, M.D. DeHart, and K.T. Clarno, "Implementation of Two-Level Coarse-Mesh Finite Difference Acceleration in an Arbitrary Geometry, Two-Dimensional Discrete Ordinates Transport Method," *Nuclear Science and Engineering*, Vol. 158, No. 3, 2008, pp. 289–298.

[64] B.T. Adams and J.E. Morel, "A Two-Grid Acceleration Scheme for the Multigroup SN Equations With Neutron Upscattering," *Nuclear Science and Engineering*, Vol. 115, No. 3, 1993, pp. 253–264.

[65] R.N. Slaybaugh, T.M. Evans, G.G. Davidson, and P.P.H. Wilson, "Multigrid in Energy Preconditioner for Krylov Solvers," *Journal of Computational Physics*, Vol. 242, 2013, pp. 405–419.

[66] L.R. Cornejo and D.Y. Anistratov, "Nonlinear Diffusion Acceleration Method With Multigrid in Energy for K-Eigenvalue Neutron Transport Problems," *Nuclear Science and Engineering*, Vol. 184, No. 4, 2016, pp. 514–526.

[67] E.W. Larsen, "A Grey Transport Acceleration Method Far Time-Dependent Radiative Transfer Problems," *Journal of Computational Physics*, Vol. 78, No. 2, 1988, pp. 459–480.

[68] E.W. Larsen, "Transport Acceleration Methods as Two-Level Multigrid Algorithms," *Modern Mathematical Methods in Transport Theory*, Springer, 1991, pp. 34–47.

[69] P.F. Nowak, "A Grey Diffusion Acceleration Method for Time-Dependent Radiative Transfer Calculation: Analysis and Application," *Proc. ANS Topical Meeting, Mathematical Methods and Supercomputing in Nuclear Applications, M&C+ SNA'93, April 19–23, 1993, Karlsruhe, Germany*, Vol. 1, 1993.

[70] H. Park, D. Knoll, R.M. Rauenzahn, A.B. Wollaber, and J.D. Densmore, "A Consistent, Moment-Based, Multiscale Solution Approach for Thermal Radiative Transfer Problems," *Transport Theory and Statistical Physics*, Vol. 41, No. 3-4, 2012, pp. 284–303.

[71] B.C. Yee, A.B. Wollaber, T.S. Haut, and H. Park, "A Stable 1D Multigroup High-Order Low-Order Method," *Journal of Computational and Theoretical Transport*, Vol. 46, No. 1, 2017, pp. 46–76.

[72] A. Brandt, "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Mathematics of computation*, Vol. 31, No. 138, 1977, pp. 333–390.

[73] C. Hao, Y. Xu, and T.J. Downar, "Multi-Level Coarse Mesh Finite Difference Acceleration With Local Two-Node Nodal Expansion Method," *Annals of Nuclear Energy*, Vol. 116, 2018, pp. 105–113.

[74] D.F. Gill, Y.Y. Azmy, J.S. Warsa, and J.D. Densmore, "Newton's Method for the Computation of k-Eigenvalues in SN Transport Applications," *Nuclear Science and Engineering*, Vol. 168, No. 1, 2011, pp. 37–58.

[75] J.J. Duderstadt and L.J. Hamilton, *Nuclear Reactor Analysis*, Vol. 1, Wiley New York, 1976.

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

[77] E.W. Larsen, *Course Notes for NERS 543: Nuclear Reactor Theory II*, University of Michigan, 2013.

[78] R.J. Stamm'ler and M.J. Abbate, *Methods of Steady-State Reactor Physics in Nuclear Design*, Vol. 111, Academic Press London, 1983.

[79] K.S. Kim, M.L. Williams, D. Wiarda, and A.T. Godfrey, "Development of a New 47-Group Library for the CASL Neutronics Simulators," Tech. rep., Oak Ridge National Laboratory (ORNL); Consortium for Advanced Simulation of LWRs (CASL), 2015.

[80] B.C. Yee, E.W. Larsen, and B. Kochunas, "An Analytical Derivation of Transport-Corrected P0 Cross Sections and Diffusion Coefficients," *Proceedings of the 2016 PHYSOR Conference*, ANS, 2016, Sun Valley, Idaho.

[81] L.N. Trefethen and D. Bau III, *Numerical Linear Algebra*, Vol. 50, Siam, 1997.

[82] D. Serre, *Iterative Methods for Linear Problems*, Chap. 9, Springer, 2002, pp. 149–167.

[83] D. Young, "Iterative Methods for Solving Partial Difference Equations of Elliptic Type," *Transactions of the American Mathematical Society*, Vol. 76, No. 1, 1954, pp. 92–111.

[84] S. Yang and K.G. Matthias, "The Optimal Relaxation Parameter for the SOR Method Applied to a Classical Model Problem," Tech. rep., Technical Report number TR2007-6, Department of Mathematics and Statistics, University of Maryland, Baltimore County, 2007.

[85] A. Vidal, A. Kassab, D. Mota, and D. Dechev, "A Multithreaded Solver for the 2D Poisson Equation," *Proceedings of the 2012 Symposium on High Performance Computing*, Society for Computer Simulation International, 2012.

[86] R.S. Sampath, P. Barai, and P.K. Nukala, "A Parallel Multigrid Preconditioner for the Simulation of Large Fracture Networks," *Journal of Statistical Mechanics: Theory and Experiment*, Vol. 2010, No. 03, 2010, pp. P03029.

[87] R.J. LeVeque, *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*, Vol. 98, Siam, 2007.

[88] J. Liesen and P. Tichỳ, "The Field of Values Bound on Ideal GMRES," Vol. 1211.5969, 2012.

[89] A. Ghai, C. Lu, and X. Jiao, "A Comparison of Preconditioned Krylov Subspace Methods for Nonsymmetric Linear Systems," *arXiv preprint arXiv:1607.00351*, 2016.

[90] T.J. Downar et al., "PARCS: Purdue Advanced Reactor Core Simulator," *PHYSOR 2002*, October 7–10, Seoul, 2002.

[91] S.P. Hamilton and T.M. Evans, "Efficient Solution of the Simplified PN Equations," *Journal of Computational Physics*, Vol. 284, 2015, pp. 155–170.

[92] R.B. Morgan, "Generalizations of Davidson's Method for Computing Eigenvalues of Large Nonsymmetric Matrices," *Journal of Computational Physics*, Vol. 101, No. 2, 1992, pp. 287–291.

[93] E.R. Davidson, "The Iterative Calculation of a Few of the Lowest Eigenvalues and Corresponding Eigenvectors of Large Real-Symmetric Matrices," *Journal of Computational Physics*, Vol. 17, No. 1, 1975, pp. 87–94.

[94] R.B. Morgan, "Davidson's Method and Preconditioning for Generalized Eigenvalue Problems," *Journal of Computational Physics*, Vol. 89, No. 1, 1990, pp. 241–245.

[95] P.F. Dubois, K. Hinsen, and J. Hugunin, "Numerical Python," *Computers in Physics*, Vol. 10, No. 3, May/June 1996.

[96] J. Millman and T. Vaught, "The State of SciPy," *Proceedings of the 7th Python in Science Conference*, edited by G. Varoquaux, T. Vaught, and J. Millman, Pasadena, CA USA, 2008.

[97] J.A. Turner, K. Clarno, M. Sieger, R. Bartlett, B. Collins, R. Pawlowski, R. Schmidt, and R. Summers, "The Virtual Environment for Reactor Applications (VERA): Design and Architecture," *Journal of Computational Physics*, Vol. 326, 2016, pp. 544–568.

[98] B. Kochunas, B. Collins, D. Jabaay, T.J. Downar, and W.R. Martin, "Overview of Development and Design of MPACT: Michigan Parallel Characteristics Transport Code," Tech. rep., American Nuclear Society, 555 North Kensington Avenue, La Grange Park, IL 60526 (United States), 2013.

[99] S. Palmtag, "MPACT Library Verification by Comparison of Pincell Calculations to Monte Carlo Results," Tech. rep., CASL-U-2016-0281-002, Rev. 2, 2017.

[100] M. Snir, *MPI–The Complete Reference*, Vol. 1, MIT press, 1998.

[101] A. Zhu, Y. Xu, and T. Downar, "A Multilevel Quasi-Static Kinetics Method for Pin-Resolved Transport Transient Reactor Analysis," *Nuclear Science and Engineering*, Vol. 182, No. 4, 2016, pp. 435–451.

[102] G. Teschl, *Ordinary Differential Equations and Dynamical Systems*, Vol. 140, American Mathematical Society Providence, 2012.

[103] B. Kochunas, A. Fitzgerald, and E. Larsen, "Fourier Analysis of Iteration Schemes for k-Eigenvalue Transport Problems with Flux-Dependent Cross Sections," *Journal of Computational Physics*, Vol. 345, 2017, pp. 294–307.

[104] B.W. Kelley and E.W. Larsen, "A consistent 2D/1D approximation to the 3D neutron transport equation," *Nuclear Engineering and Design*, Vol. 295, 2015, pp. 598 – 614.

[105] B. Collins, S. Stimpson, A. Godfrey, B. Kochunas, and T. Downar, "Assessment of the 2D/1D Implementation in MPACT," *PHYSOR*, Kyoto, Japan, 2014.

[106] J.J. Hu and A. Prokopenko, "MueLu: Multigrid Framework for Advanced Architectures," Tech. rep., Sandia National Laboratories (SNL-CA), Livermore, CA (United States); Sandia National Laboratories, Albuquerque, NM, 2015.

[107] B.C. Yee, B. Kochunas, E.W. Larsen, and Y. Xu, "Space-Dependent Wielandt Shifts for Multigroup Diffusion Eigenvalue Problems," *Nuclear Science and Engineering*, Vol. 188, No. 2, 2017, pp. 140–159.

[108] D. Gilbarg and N.S. Trudinger, *Elliptic Partial Differential Equations of Second Order*, springer, 2015.

[109] M.H. Protter and H.F. Weinberger, *Maximum Principles in Differential Equations*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1967.

[110] A. Till, *Finite Elements With Discontiguous Support for Energy Discretization in Particle Transport*, Ph.D. thesis, 2015.

[111] B. Collins and S. Stimpson, "Acceleration Methods for Whole Core Reactor Simulations using VERA," *2018 ANS Annual Meeting*, June 17–21, Philadelphia, PA, 2018.

[112] A.T. Godfrey, "VERA Core Physics Benchmark Progression Problem Specifications," Tech. Rep. CASL-U-2012-0131-004, Oak Ridge National Laboratory, 2014.

[113] M. Khalil and P. Wesseling, "Vertex-Centered and Cell-Centered Multigrid for Interface Problems," *Journal of Computational Physics*, Vol. 98, No. 1, 1992, pp. 1–10.

[114] R. Swanson and R. Radespiel, "Cell Centered and Cell Vertex Multigrid Schemes for the Navier-Stokes Equations," *AIAA journal*, Vol. 29, No. 5, 1991, pp. 697–703.

[115] X.S. Li, "An Overview of SuperLU: Algorithms, Implementation, and User Interface," *ACM Transactions on Mathematical Software*, Vol. 31, No. 3, September 2005, pp. 302–325.

[116] H. De Sterck, U.M. Yang, and J.J. Heys, "Reducing Complexity in Parallel Algebraic Multigrid Preconditioners," *SIAM Journal on Matrix Analysis and Applications*, Vol. 27, No. 4, 2006, pp. 1019–1039.

[117] "HPC Tools – Arm," https://www.arm.com/products/development-tools/hpc-tools, Accessed: 2018-04-15.

[118] S. Stimpson and B. Collins, "Flexible Spatial Partitions in MPACT Through Module-Based Data Passing," *Transactions of the American Nuclear Society, 2017 ANS Annual Meeting*, Vol. 116, San Francisco, CA, 2017.

[119] A. Fitzgerald, Z. Dodson, S. Stimpson, and B. Kochunas, "Automated Decomposition of a Structured Grid," *2017 ANS Winter Meeting*, October 29 – November 2, Washington D.C., 2017.

[120] Y. Liu, W. Martin, M. Williams, and K.S. Kim, "A Full-Core Resonance Self-shielding Method Using a Continuous-Energy Quasi–One-Dimensional Slowing-Down Solution that Accounts for Temperature-Dependent Fuel Subregions and Resonance Interference," *Nuclear Science and Engineering*, Vol. 180, No. 3, 2015, pp. 247–272.

[121] E.R. Wolters, E.W. Larsen, and W.R. Martin, "Hybrid Monte Carlo-CMFD Methods for Accelerating Fission Source Convergence," *Nuclear Science and Engineering*, Vol. 174, 2013, pp. 286–299.

[122] N. Larsen, "Core Design and Operating Data for Cycles 1 and 2 of Peach Bottom 2," Tech. Rep. NP-563, EPRI, 1978.

[123] F. Franceschini, A. Godfrey, J. Kulesza, and R. Oelrich, "Westinghouse VERA Test Stand: Zero Power Physics Test Simulations for the AP1000 PWR," Tech. Rep. CASL-U-2014-0012-000, Consortium for Advanced Simulation of LWRs, 2014.