ACCELERATION METHODS FOR MASSIVELY PARALLEL DETERMINISTIC TRANSPORT

by

Rachel N. Slaybaugh

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

Doctor of Philosophy

(Engineering Physics)

at the

UNIVERSITY OF WISCONSIN–MADISON

6 October 2011

ACKNOWLEDGMENTS

First and foremost I would like to thank my thesis advisor Paul Wilson, my project mentor Tom Evans, and my fellowship mentor Tom Sutton for their instruction and guidance. I would also like to thank Steve Wilson, my mentor from my summer practicum at Bettis. Many other individuals such as Yousry Azmy and Bernadette Kirk, among others, have been instrumental to my professional development as well. Greg Davidson was extremely helpful throughout my work on Denovo. My colleagues at Karlsruhe were wonderful. Milad Fatenejad and Katy Huff made my life easier by teaching me about software carpentry while feeding me brownies and making me laugh.

This research was performed under appointment to the Rickover Fellowship Program in Nuclear Engineering sponsored by Naval Reactors Division of the U.S. Department of Energy. I would also like to thank Oak Ridge National Laboratory and their computing facilities (OLCF) for computer time and project support.

Finally, I would like to thank my friends and family without whom I could not do much of anything. My mom has always been my hero; my Dad is good at telling me he's proud of me; I've had fun watching my brother become a rock star in his own right; Jenna and Sam have long served as rocks; Julie and Megan have kept me laughing since I met them; all of my roommates in grad school have put up with my crazy kitchen adventures; the hash has kept me running; and all the folks I love who I haven't called out individually have made things interesting. Thank you all for inspiring me and making my life fun, fulfilling, and wonderful.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Chapter 1

Introduction

Nuclear technology plays an important role in society, particularly within the field of energy generation. More nuclear reactors are being constructed, existing designs are being refined, and new plants are being developed. For progress to continue in these areas, the modeling of nuclear systems must also progress.

The neutron transport equation describes “where all the neutrons are” in a nuclear system. The more accurately this is known, the more accurately new systems can be developed. This means that solutions to the transport equation are needed in high-fidelity in all parts of phase space. Very large computers are now available to perform such high-fidelity calculations, but most existing solution methods are not able to take full advantage of new computer architectures. This work aims to accelerate the transport solution with new methods that use new machines fully, facilitating the design of better nuclear systems.

The goal of this research is to accelerate transport calculations with methods that use new computers fully, facilitating the design of better nuclear systems. Three complimentary methods have been implemented to accomplish this goal. In the chapters that follow this, background information, past work, mathematics and implementation, and results will be discussed for each method in turn.

This introductory chapter is intended to provide the foundation of this document. The types of problems that have been, are, and could be solved by neutron transport are discussed first, including why such problems matter. This is followed by a presentation of the transport equation and its discretization. Finally, the goals of this research are outlined. The remainder of this report provides the details about how and why these goals have been met.

1.1 Motivation

The problems typically of interest in the nuclear engineering community are of large scale, with many independent variables representing the pertinent nuclear physics. Some of the most important applications are finding reactor core power distributions for cooling and safety needs, determining the criticality state of the reactor, and predicting isotope depletion. All of these applications require high-resolution neutron flux spectra, where flux is the number of particles per cm^2 per second in some portion of phase of space. Commercial light water reactors can have core heights of three to four meters and contain anywhere from 700 to 1200 fuel bundles [23]. Geometrically large shielding applications are commonly of interest as well. To obtain the quantities of interest with sufficient precision and resolution, the size of required calculations can become quite large.

Solving the full steady-state transport equation, which depends on location, energy, and solid angle, is computationally intensive. For a reasonable discretization of reactor-type calculations, 10^8 coupled algebraic equations could easily be required. In the past, such large calculations were generally intractable because of computer hardware limitations: lack of memory and processing capability. There was simply not enough space to store all of the required data, and it would take too long to conduct the number of floating point operations (flops) needed. Calculations for the problems of interest were impossible to perform using the transport equation with fine discretizations. [17].

Accordingly, simplifying approximations were used to solve problems in practice. Approximations include modeling reactor geometries in one or two dimensions rather than three; approximating neutron sources as isotropic; approximating neutron scattering as isotropic; eliminating the angular component of the solution through the diffusion approximation (discussed in A); physically truncating geometries by taking advantage of geometric symmetries or near symmetries, particularly for reactor cores with repeated lattice structures; and suppressing energy dependence or using very few energy groups [17].

While all such approximations can be appropriate and give good results in some situations, they are generally not as accurate as solving the full, finely discretized transport equation. The way the nuclear industry has compensated for approximate answers is by building conservative margins into designs by using thicker shields, lower operating powers, larger safety margins, and so on. All of this costs money, which is of crucial import since economic competitiveness may be the largest barrier to the construction of new nuclear plants and, correspondingly, provision of emissions-free energy. Having higher-fidelity neutron fluxes could influence design bases and have a meaningful impact on current reactor operations and new reactor designs.

1.1.1 Problems of Interest

To get more accurate fluxes, typical transport problems today are three-dimensional, have up to thousands $\times$ thousands $\times$ thousands of mesh points, use up to $\sim$150 energy groups, include accurate expansions of scattering terms, and are solved over many directions. Some examples of problems solved recently using discrete ordinates codes on parallel machines are shown next. The exact meaning of the expansions will become clear in the next section.

- In 1998, 30 million unknowns[1]: a 3-D time dependent shielding problem where a 10 $\times$ 10 $\times$ 10 cm innermost region containing water and a uniform source is surrounded by 10 cm of iron, which is surrounded by 30 more cm of water. Three energy groups, a 50 $\times$ 50 $\times$ 50 Cartesian mesh, S_8 scattering quadrature, P_0 scattering expansion, and the adaptive weighted diamond difference method were used [5].

- In 2004, 1.62 million unknowns: a cylinder with a 3.5 cm radius and 9 cm length containing layers of boron-10, water, and highly enriched uranium was solved with 13,500 cells, S_4 Chebyshev-Legendre quadrature, and five energy groups [70].

- In 2010, 78.5 billion unknowns: a Pressurized Water Reactor (PWR)-900 with two groups, a 578 $\times$ 578 $\times$ 700 mesh, using S_{16} level-symmetric quadrature, and with P_0 scattering was solved [16].

[1]In each example the number of unknowns assumes one unknown per cell.

The types of problems of interest have been increasing in size and scope as larger computer resources have made it possible to solve such problems.

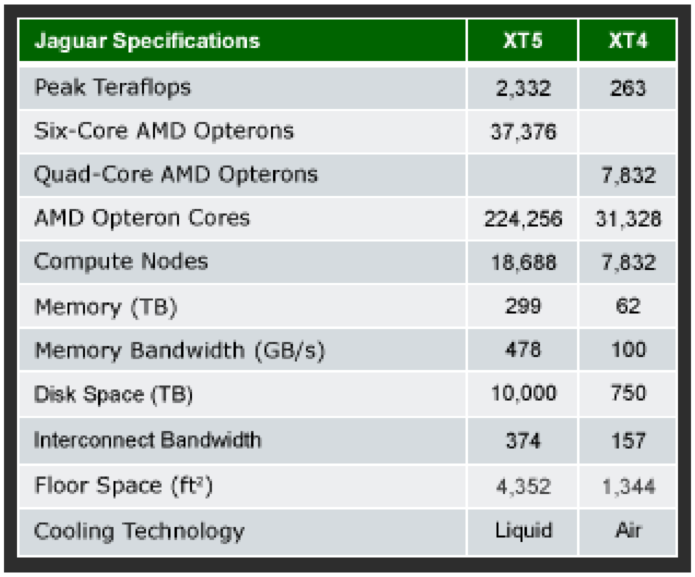
The next phase of challenging problems are even more highly refined. High-fidelity, coupled, multi-physics calculations are the new “grand challenge” problems for reactor analysis. Of particular interest in nuclear systems is feedback between neutronics and thermal-hydraulics. A recent INCITE grant was awarded to spatially quantify data uncertainties in 3-D Boiling Water Reactor assembly calculations. When neutron transport is coupled to a fluid dynamics code it is important to know whether or not homogenizing the materials in the subchannels being analyzed will have a large impact. Such uncertainty studies will require spatial meshes exceeding 500 million elements [19]. Calculations of this detail present the next generation of challenges for computational neutronics.

1.1.2 Enabling Technology

Nuclear transport computation began in the age of run tapes and punch cards with widely used codes being written in advanced programming languages such as FORTRAN IV [24]. Computer technology has evolved quickly and radically since that time. Now there are machines like Jaguar, which consists of two partitioned machines, the Cray XT4 and the Cray XT5. The XT4 has 84 cabinets of quad-core cores with eight gigabytes of memory per node. The XT5 has 200 cabinets of six-core cores with 16 gigabytes of memory per node, giving a total of 362 terabytes of high-speed memory [57]. A table of machine parameters can be seen in Figure 1.1.

While the fastest computer in the world in 2010 may not be the best example of what is standard, machines on which parallel codes can be run are widely available and are becoming faster over time. Access to such machines has changed the landscape of the types of neutron transport problems that can be solved practically.

Whenever an equation is being mathematically approximated, errors can be introduced that are inherent in the approximation. These errors are present regardless of machine roundoff, phase space discretization, etc. and cannot be changed without changing the approximation. Accepting this mathematical limit as a given, the factor that can be changed to improve calculations is machine



Jaguar Specifications	XT5	XT4
Peak Teraflops	2,332	263
Six-Core AMD Opterons	37,376	
Quad-Core AMD Opterons		7,832
AMD Opteron Cores	224,256	31,328
Compute Nodes	18,688	7,832
Memory (TB)	299	62
Memory Bandwidth (GB/s)	478	100
Disk Space (TB)	10,000	750
Interconnect Bandwidth	374	157
Floor Space (ft^2)	4,352	1,344
Cooling Technology	Liquid	Air

Figure 1.1 Jaguar Machine Specifications [57]

architecture. This means that the quality of calculations for a given method is ultimately limited by computers. The research presented here is intended to provide methods that can take full advantage of these new computers, pushing back the frontier of limiting calculations.

1.2 The Transport Equation

To understand the methods that were developed for this work and how they will enable the use of leadership-class hardware, the mathematical details of the transport equation must be discussed first. This subsection discusses those details for neutrons in steady-state systems.

Neutrons can have many different kinds of interactions that influence a system's behavior. These interactions are described by cross sections, which reflect the likelihood of a particular interaction occurring and are given in units of inverse length. Of particular interest is the total cross section, Σ, which includes all possible interactions; the scattering cross section, Σ_s, where scattering can change the momentum and kinetic energy of a neutron; and the fission cross section, Σ_f.

Fission is the process through which a nucleus splits into (typically two) smaller atoms. Through this process a relatively large amount of energy is released along with several neutrons. These neutron are available to go on to cause other fissions, creating a chain reaction. The quantity ν is the average number of neutrons released per fission. Most of the energy released from fission is kinetic, which creates the heat used to make electricity [43].

If fission is occurring, it is often of interest to know the asymptotic behavior of the system. A reactor is called "critical" if the chain reaction is self-sustaining and time-independent. If the system is not in equilibrium then the asymptotic neutron distribution, or the fundamental mode, will grow or decay exponentially over time. A convenient way to capture this behavior is to assume ν can be adjusted to obtain a time-independent solution by replacing it with $\frac{\nu}{k}$, where k is the parameter expressing the deviation from critical. This substitution changes the transport equation into an eigenvalue problem. A spectrum of eigenvalues can be found, but at long times only the non-negative solution corresponding to the largest real eigenvalue will dominate. The eigenproblem can be written as:

$$\begin{aligned}[\hat{\Omega} \cdot \nabla + \Sigma(\vec{r}, E)]\psi(\vec{r}, \hat{\Omega}, E) = & \int dE' \int d\hat{\Omega}' \, \Sigma_s(\vec{r}, E' \to E, \hat{\Omega}' \cdot \hat{\Omega})\psi(\vec{r}, \hat{\Omega}', E') \\ & + \frac{\chi(E)}{k} \int dE' \, \nu\Sigma_f(\vec{r}, E') \int d\hat{\Omega}' \, \psi(\vec{r}, \hat{\Omega}', E') \,, \end{aligned} \tag{1.1}$$

where the quantities are at location $\vec{r}$, traveling in directions $\hat{\Omega}$, and at energy E and are defined as:

$\psi(\vec{r}, \hat{\Omega}, E)$ is the angular neutron flux in neutrons per unit length squared per steradian and expresses where all the neutrons are in phase space,

$\chi(E)$ is the fission spectrum and specifies the energy distribution of neutrons born from fission,

k is the eigenvalue, which can be thought of as the asymptotic ratio of the number of neutrons in one generation to the number in the next [43].

If fission is not present the transport equation becomes a fixed source rather than eigenvalue problem. The term containing k is replaced by an external source, $q_{ex}(\vec{r}, \hat{\Omega}, E)$, and the equation becomes a standard linear system.

A brief aside about some properties of the transport equation will aid in understanding the challenge of finding solution techniques that work for all problems of interest. In a void, the transport equation is like a hyperbolic wave equation. For highly-scattering regions where Σ_s is close to Σ, the equation becomes elliptic for the steady-state case. If the scattering is forward-peaked then the equation is parabolic. All of these classes of equations have different solution strategies. The equation is linear, though non-linearities can be introduced if temperature-dependence of cross sections and other similar physics are considered [3]. The non-linear considerations are beyond the scope of this work and only the linear case is considered here.

To numerically solve Equation (1.1) it is discretized in space, angle, and energy. This work uses the multigroup approximation, the scattering term is expanded into Legendre polynomials, and discrete ordinates to treat direction of neutron travel. There are many spatial differencing methods available, the discussion of which are beyond the scope of this document as the proposed work is not dependent upon the spatial discretization employed. To ensure the new methods apply to the most general cases it will be assumed that the matrices resulting from discretization are be non-symmetric.

1.2.1 Multigroup Approximation

The first variable to discretize is energy. In the multigroup approximation, the energy range of interest is broken into G groups. The neutron flux is constant in energy over each group. The groups are ordered such that the highest energy group bound corresponds with $g = 0$, meaning group 1 is defined over E_1 to E_0, and the lowest energy bound corresponds with $g = G$. On this grid the group angular flux is defined as:

$$\psi_g(\vec{r}, \hat{\Omega}) = \int_g dE\, \psi(\vec{r}, \hat{\Omega}, E) = \int_{E_g}^{E_{g-1}} dE\, \psi(\vec{r}, \hat{\Omega}, E)\,. \tag{1.2}$$

Justification of this definition comes from assuming the angular flux within each energy group can be approximated by the product of the group flux and a known function: $\psi(\vec{r}, \hat{\Omega}, E) \approx f(E)\psi_g(\vec{r}, \hat{\Omega})$ for $E_g < E \leq E_{g-1}$. The function is normalized for a specific group g such that $\int_{g'=g} dE\, f(E) = 1$ and $\int_{g' \neq g} dE\, f(E) = 0$. The details of f are isotope-dependent and can be garnered from nuclear data. With this notation, group quantities are defined as:

$\Sigma_g(\vec{r}) = \int_g dE\, \Sigma(\vec{r}, E) f(E)$,
$\nu\Sigma_{fg}(\vec{r}) = \int_g dE\, \nu\Sigma_f(\vec{r}, E) f(E)$,
$\Sigma_s^{gg'}(\vec{r}, \hat{\Omega}' \cdot \hat{\Omega}) = \int_g dE\, \int_{g'} dE'\, \Sigma_s(\vec{r}, E' \to E, \hat{\Omega}' \cdot \hat{\Omega}) f(E')$ is the scattering from g' into g,
$\chi_g = \int_g dE\, \chi(E)$, and
$q_g^e(\vec{r}, \hat{\Omega}) = \int_g dE\, q_{ex}(\vec{r}, \hat{\Omega}, E)$.

Using these terms, the eigenvalue form of Equation (1.1) can be rewritten for each group g as seen in (1.3) [43]. Using more energy groups represents the physics in the cross sections more accurately, but also increases the cost of a calculation in terms of both operations and storage.

$$\begin{aligned}[\hat{\Omega} \cdot \nabla + \Sigma_g(\vec{r})]\psi_g(\vec{r}, \hat{\Omega}) = &\sum_{g'=1}^{G} \int d\hat{\Omega}'\, \Sigma_s^{gg'}(\vec{r}, \hat{\Omega}' \cdot \hat{\Omega}) \psi_{g'}(\vec{r}, \hat{\Omega}') \\ &+ \frac{\chi_g}{k} \sum_{g'=1}^{G} \nu\Sigma_{fg'}(\vec{r}) \int d\hat{\Omega}'\, \psi_{g'}(\vec{r}, \hat{\Omega}') \end{aligned} \tag{1.3}$$

1.2.2 Scattering Discretization

In Equation (1.3) the scattering cross section is a complicated function of angle. Simplifying this term will make the system easier to solve. Because of rotational symmetry of nuclear collisions, Σ_s is only a function of the cosine between the incoming and outgoing angles. This allows the scattering cross section to be written as a series expansion of Legendre polynomials as follows, where the the spatial variable has been supressed:

$$\Sigma_s^{gg'}(\hat{\Omega}' \cdot \hat{\Omega}) = \sum_{l=0}^{N} \frac{2l+1}{4\pi} P_l(\hat{\Omega}' \cdot \hat{\Omega}) \Sigma_{s\,l}^{gg'}\ . \tag{1.4}$$

The addition theorem of spherical harmonics can be used to evaluate the Legendre function, $P_l(\hat{\Omega}' \cdot \hat{\Omega})$ with spherical harmonic terms, Y_{lm}. These can then be expanded into real and imaginary components. The imaginary components must be zero since scattering must be real. All of this gives:

$$P_l(\hat{\Omega}' \cdot \hat{\Omega}) = \frac{4\pi}{2l+1} \Big[Y_{l0}^e(\hat{\Omega}) Y_{l0}^e(\hat{\Omega}') + \sum_{m=1}^{l} \big(Y_{lm}^e(\hat{\Omega}) Y_{lm}^e(\hat{\Omega}') + Y_{lm}^o(\hat{\Omega}) Y_{lm}^o(\hat{\Omega}') \big) \Big]\ . \tag{1.5}$$

The Y^e and Y^o terms obey the following orthogonality relationships: $\int d\hat{\Omega}\; Y^e_{lm}(\hat{\Omega})Y^e_{l'm'}(\hat{\Omega}) = \frac{1}{2}(1+\delta_{m0})\delta_{ll'}\delta_{mm'}$ and $\int d\hat{\Omega}\; Y^o_{lm}(\hat{\Omega})Y^o_{l'm'}(\hat{\Omega}) = \frac{1}{2}(1-\delta_{m0})\delta_{ll'}\delta_{mm'}$. The spherical harmonics therefore form an orthonormal basis in which the Legendre polynomials are expressed [18], [43].

By putting all of this information together, the scattering source can be expressed as:

$$q^g_s(\vec{r},\hat{\Omega}) = \sum_{g'=1}^{G}\sum_{l=0}^{N}\Sigma^{gg'}_{s\,l}(\vec{r})\Big[Y^e_{l0}(\hat{\Omega})\phi^{g'}_{l0}(\vec{r}) + \sum_{m=1}^{l}\big(Y^e_{lm}(\hat{\Omega})\phi^{g'}_{lm}(\vec{r}) + Y^o_{lm}(\hat{\Omega})\vartheta^{g'}_{lm}(\vec{r})\big)\Big]\,. \tag{1.6}$$

In Equation (1.6) two terms, called the even and odd flux moments, have been used:

$$\phi^g_{lm} = \int_{4\pi} d\hat{\Omega}'\; Y^e_{lm}(\hat{\Omega}')\psi^g(\hat{\Omega}')\,,\quad m \geq 0\,,\qquad \text{even}\,, \tag{1.7}$$

$$\vartheta^g_{lm} = \int_{4\pi} d\hat{\Omega}'\; Y^o_{lm}(\hat{\Omega}')\psi^g(\hat{\Omega}')\,,\quad m > 0\,,\qquad \text{odd}\,. \tag{1.8}$$

The external source can be similarly discretized if desired, making the two sources consistent with one another. The full details of these expansions and bases can be found in [18].

The multigroup, anisotropic scattering source found in Equation (1.6) is characterized by the order of Legendre scattering, P_N. For a given Legendre expansion there are $(N+1)^2$ flux moments. Using more moments represents scattering more accurately, but increases the cost of a calculation [18].

1.2.3 Discrete Ordinates Approximation

The next area of phase space to discretize is direction. The discrete ordinates or S_N approximation is a collocation method that is used to express the transport equation on a discrete set of n ordinates, where the ordinates are described by angles and represent direction of neutron travel. A collocation method represents a continuous function on a finite set of collocation points using a linear combination of basis functions. The approximate solution must satisfy the differential equation at those collocation points [27].

For the transport equation, the basis functions are the angular fluxes along specific directions and the collocation points are the angle sets. The linear combination is done by weighting the basis functions using a quadrature set.

To implement this, the transport equation is written for neutrons traveling in $d\hat{\Omega}$ about direction $\hat{\Omega}_a$. Now ψ_a^g can be defined as $\psi^g(\hat{\Omega}_a)$, and the angular flux can be related to the flux moments as $\phi^g = \sum_{a=1}^{n} \psi_a^g w_a$. Using these terms, including the discretized scattering source, and suppressing spatial dependence, Equation (1.3) becomes:

$$\begin{aligned}\big(\hat{\Omega}_a \cdot \nabla + \Sigma_g\big)\psi_a^g = \sum_{g'=1}^{G}\sum_{l=0}^{N} \Sigma_{s\,l}^{gg'} \Big[Y_{l0}^e(\hat{\Omega}_a)\phi_{l0}^{g'} + \sum_{m=1}^{l}\big(Y_{lm}^e(\hat{\Omega}_a)\phi_{lm}^{g'} + Y_{lm}^o(\hat{\Omega}_a)\vartheta_{lm}^{g'}\big)\Big] \\ + \frac{\chi^g}{k}\sum_{g'=1}^{G} \nu\Sigma_f^{g'}\phi^{g'}\ , \qquad (1.9)\end{aligned}$$

The quadrature weights, w_a, are defined such that $\int_{4\pi} d\hat{\Omega} = \sum_{a=1}^{n} w_a = 4\pi$. One of the choices in the S_N approximation is what quadrature set to use and the number of unknowns contributed by discretization of direction is determined by the quadrature set. For example, level-symmetric quadrature gives $n = N(N+2)$ unknowns for an S_N approximation. The flux moments are also collocated on the ordinates using the desired quadrature set to linearly combine the angular flux basis functions [18]:

$$\phi_{lm}^g = \sum_{a=1}^{n} Y_{lm}^e(\hat{\Omega}'_a)\psi_a^g w_a\ , \qquad (1.10)$$

$$\vartheta_{lm}^g = \sum_{a=1}^{n} Y_{lm}^o(\hat{\Omega}'_a)\psi_a^g w_a\ . \qquad (1.11)$$

1.2.4 Operator Form

Now that the transport equation has been discretized, it can be expressed in operator notation. Using the operator form of the transport equation will facilitate the presentation of the solution techniques discussed in the remainder of this document. In general, uppercase bolded letters will indicate matrices and lowercase italicized letters will indicate vectors and scalars. The following operators are used to express the transport equation:

$\mathbf{L} = \hat{\Omega} \cdot \nabla + \Sigma$ is the transport operator,

M is the operator that converts harmonic moments into discrete angles,

S is the scattering matrix,

f contains the fission source, $\nu\Sigma_f$; $\mathbf{F} = \chi f^T$,

$\mathbf{D} = \mathbf{M^T W} = \sum_{a=1}^{n} Y_{lm}^{e/o} w_a$ is the discrete-to-moment operator.

With this notation Equation (1.9) can be written as (1.12); it can be formulated as a fixed source problem by replacing the fission term with $\mathbf{M}q_e$. This has two unknowns, the angular flux and the moments, which are related by the discrete-to-moment operator as seen in Equation (1.13).

$$\mathbf{L}\psi = \mathbf{MS}\phi + \frac{1}{k}\mathbf{MF}\phi \tag{1.12}$$

$$\phi = \mathbf{D}\psi \tag{1.13}$$

The typical strategy for solving Equation (1.12) is to combine it with (1.13) and form one equation involving only the moments, where Q is comprised of the fixed or fission source:

$$(\mathbf{I} - \mathbf{DL}^{-1}\mathbf{MS})\phi = Q\,. \tag{1.14}$$

This is solved for ϕ and ψ can be backed out at the end of the calculation.

The size of the operators can be defined in terms of the granularity of discretization:

G = number of energy groups,

t = number of moments,

n = number of angular unknowns,

c = number of cells,

u = number of unknowns per cell, which is determined by spatial discretization.

These can be combined to define $a = G \times n \times c \times u$ and $f = G \times t \times c \times u$. Using a and f, Equation (1.12) can be presented in terms of operator size: $(a \times a)(a \times 1) = (a \times f)(f \times f)(f \times 1) + (a \times f)(f \times 1)$. The index variables, their meaning, and their ranges are shown in Table 1.1.

The structures of the vectors and matrices are shown in the next few equations as this will make some of the proposed methods easier to understand and visualize [18]. The angular flux vector is explicitly written first, where each discrete set of angular fluxes, ψ_a^g, includes all spatial unknowns, and the matrices follow:

$$\psi = \Big([\psi]_1 \quad [\psi]_2 \quad \cdots \quad [\psi]_g \quad \cdots [\psi]_G\Big)^T , \qquad \text{and}$$

$$[\psi]_g = \Big(\psi_1^g \quad \psi_2^g \quad \cdots \quad \psi_a^g \quad \cdots \psi_n^g\Big)^T .$$

Table 1.1 Meaning and Range of Indices Used in Transport Discretization

Variable	Symbol	First	Last
Energy	g	1	G
Solid Angle	a	1	n
Space	suppressed	n/a	n/a
Legendre moment (P_N)	l	0	N
Spherical harmonic moment (Y)	m	0	l

$$\mathbf{M} = \begin{pmatrix} [\mathbf{M}]_{11} & 0 & 0 & \cdots & 0 \\ 0 & [\mathbf{M}]_{22} & 0 & \cdots & 0 \\ 0 & 0 & [\mathbf{M}]_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & [\mathbf{M}]_{GG} \end{pmatrix}, \qquad \mathbf{S} = \begin{pmatrix} [\mathbf{S}]_{11} & [\mathbf{S}]_{12} & [\mathbf{S}]_{13} & \cdots & [\mathbf{S}]_{1G} \\ [\mathbf{S}]_{21} & [\mathbf{S}]_{22} & [\mathbf{S}]_{23} & \cdots & [\mathbf{S}]_{2G} \\ [\mathbf{S}]_{31} & [\mathbf{S}]_{32} & [\mathbf{S}]_{33} & \cdots & [\mathbf{S}]_{3G} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ [\mathbf{S}]_{G1} & [\mathbf{S}]_{G2} & [\mathbf{S}]_{G3} & \cdots & [\mathbf{S}]_{GG} \end{pmatrix},$$

$$\mathbf{F} = \begin{pmatrix} \chi^1\nu\Sigma_f^1 & \chi^1\nu\Sigma_f^2 & \cdots & \chi^1\nu\Sigma_f^G \\ \chi^2\nu\Sigma_f^1 & \chi^2\nu\Sigma_f^2 & \cdots & \chi^2\nu\Sigma_f^G \\ \vdots & \vdots & \ddots & \vdots \\ \chi^G\nu\Sigma_f^1 & \chi^G\nu\Sigma_f^2 & \cdots & \chi^G\nu\Sigma_f^G \end{pmatrix}, \qquad [\mathbf{S}]_{gg'} = \begin{pmatrix} \Sigma_{s\,0}^{gg'} & 0 & \cdots & 0 \\ 0 & \Sigma_{s\,1}^{gg'} & \cdots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \Sigma_{s\,N}^{gg'} \end{pmatrix},$$

$$[\mathbf{M}]_{gg} = \begin{pmatrix} Y_{00}^e(\hat{\Omega}_1) & Y_{10}^e(\hat{\Omega}_1) & Y_{11}^o(\hat{\Omega}_1) & Y_{11}^e(\hat{\Omega}_1) & Y_{20}^e(\hat{\Omega}_1) & \cdots & Y_{NN}^o(\hat{\Omega}_1) & Y_{NN}^e(\hat{\Omega}_1) \\ Y_{00}^e(\hat{\Omega}_2) & Y_{10}^e(\hat{\Omega}_2) & Y_{11}^o(\hat{\Omega}_2) & Y_{11}^e(\hat{\Omega}_2) & Y_{20}^e(\hat{\Omega}_2) & \cdots & Y_{NN}^o(\hat{\Omega}_2) & Y_{NN}^e(\hat{\Omega}_2) \\ Y_{00}^e(\hat{\Omega}_3) & Y_{10}^e(\hat{\Omega}_3) & Y_{11}^o(\hat{\Omega}_3) & Y_{11}^e(\hat{\Omega}_3) & Y_{20}^e(\hat{\Omega}_3) & \cdots & Y_{NN}^o(\hat{\Omega}_3) & Y_{NN}^e(\hat{\Omega}_3) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ Y_{00}^e(\hat{\Omega}_n) & Y_{10}^e(\hat{\Omega}_n) & Y_{11}^o(\hat{\Omega}_n) & Y_{11}^e(\hat{\Omega}_n) & Y_{20}^e(\hat{\Omega}_n) & \cdots & Y_{NN}^o(\hat{\Omega}_n) & Y_{NN}^e(\hat{\Omega}_n) \end{pmatrix}.$$

Note that $[\mathbf{M}]_{11} = [\mathbf{M}]_{22} = \ldots = [\mathbf{M}]_{GG} = [\mathbf{M}]$. These are written with subscripts to simplify the visualization of which blocks correspond to which equations and multiply which other blocks.

1.2.5 Solution Procedure

Once the matrices are multiplied together, a series of single-group equations that are each only a function of space and angle result:

$$\begin{aligned}
\mathbf{L}[\psi]_1 &= [\mathbf{M}]\big([\mathbf{S}]_{11}[\phi]_1 + [\mathbf{S}]_{12}[\phi]_2 + \ldots + [\mathbf{S}]_{1G}[\phi]_G\big) + \frac{1}{k}[\mathbf{M}][\mathbf{F}]_1[\phi]_1 \,, \\
\mathbf{L}[\psi]_2 &= [\mathbf{M}]\big([\mathbf{S}]_{21}[\phi]_1 + [\mathbf{S}]_{22}[\phi]_2 + \ldots + [\mathbf{S}]_{2G}[\phi]_G\big) + \frac{1}{k}[\mathbf{M}][\mathbf{F}]_2[\phi]_2 \,, \\
&\vdots \\
\mathbf{L}[\psi]_G &= [\mathbf{M}]\big([\mathbf{S}]_{G1}[\phi]_1 + [\mathbf{S}]_{G2}[\phi]_2 + \ldots + [\mathbf{S}]_{GG}[\phi]_G\big) + \frac{1}{k}[\mathbf{M}][\mathbf{F}]_G[\phi]_G \,.
\end{aligned} \tag{1.15}$$

Each "within-group" equation is solved for that group's flux. If the groups are coupled together, as they often are, then multiple "multigroup" solves over the coupled portion of the energy range may be required. If the eigenvalue is desired an additional "eigenvalue" solve is needed as well [18]. The details of these steps will be discussed later as they relate to the new methods.

1.3 Meeting the Goal

The fully discretized steady-state transport equation can become very large when trying to solve "grand challenge" types of problems. The continued improvement of computational resources has enabled the invention of massively parallel codes designed to work on leadership-class computers. The goal of this work is to develop and implement new methods that will accelerate transport calculations by efficiently taking advantage of cutting-edge computer architectures.

The code being used in this work is Denovo [18], a massively parallel discrete ordinates code being developed at Oak Ridge National Laboratory by Tom Evans, Greg Davidson, Josh Jarrell, and others. The code allows multiple combinations of spatial discretizations, quadrature sets, and solution methodologies. Denovo is three-dimensional, uses a non-uniform cartesian grid, has a flexible front end, and is capable of writing and reading input parameters rapidly for high performance computing.

At the outset of this work, Denovo could be decomposed for parallel computation in five of the six dimensions of phase space over which the steady-state transport equation is solved. The nature of the decomposition dictated how many cores Denovo could use efficiently. Studies to be discussed in Chapter 2 illustrate that the number of cores was not sufficient to solve the real problems of interest.

The original suite of solvers in Denovo included Source Iteration (SI) and Krylov as within group solvers, Gauss-Seidel (GS) as a multigroup solver, and Power Iteration (PI) as an eigenvalue solver. These solvers have some significant limitations in many cases of interest. Each method and its associated challenges will be described in detail in following chapters.

Significant acceleration of Denovo has been accomplished by implementing three complimentary solution strategies. The first is a multigroup solution option that decouples the energy groups for fixed source problems. This allows Denovo to be decomposed in energy such that multigroup solves can be parallelized in the energy dimension. This addition allows Denovo to take advantage of many more cores without significantly degrading the scaling. The *multigroup Krylov solver* also converges more quickly than Gauss Seidel. This work is discussed in Chapter 2.

The second strategy was to add an eigenvalue method with convergence properties that can be superior to Power Iteration, particularly for problems that are difficult to solve with PI. *Rayleigh Quotient Iteration* (RQI) applies a shift to the equations that effectively couples all of the energy groups together. The multigroup Krylov solver enables RQI because it solves coupled groups efficiently and decomposes coupled groups for energy parallelization. The details of this work are discussed in Chapter 3.

The third step was to add a new physics-based preconditioner that is a multigrid method in energy. This capitalizes on the energy decomposition added by the first method. Since energy groups can be treated independently, they can be combined and re-separated in a multi-grid like fashion very easily. The preconditioner is applicable to both fixed source problems and eigenvalue calculations, and it reduces the number of iterations required for convergence in all cases. The *multigrid in energy preconditioner* is discussed in Chapter 4.

Fast and scalable codes are necessary for solving truly large and challenging neutron transport problems. This work developed methods that accelerated an existing transport code by allowing it to scale to hundreds of thousands of cores and converge more quickly than the previous methods, thus enabling the development of new and useful nuclear energy systems. These methods may also be beneficial for the larger computational community.

+

Chapter 2

Upscattering and Energy Decomposition

The first piece that was added to Denovo to accelerate the code was the multigroup Krylov solver. The second two methods rely on the energy decomposition and improved convergence added by this solver, so its functionality is essential to all subsequent work. The multigroup Krylov solver works very well and is a key element in accomplishing the goals of this work.

First, why the existing parallel decomposition in Denovo is insufficient for solving challenging problems is explained. Next, the background section discusses scattering and some solver basics, including an overview of Krylov subspace methods and why they are of interest for solving the transport equation. The next section focuses on existing methods used to solve the fixed source transport equation and relevant past work. Finally, a description of the new method and associated results are presented.

2.1 Denovo and Parallelism

To be able to solve the very large problems of interest on machines the size of Jaguar, codes that have good scaling properties are required. Recall that Denovo can be parallelized in space and angle. This is done with the Koch-Baker-Alcouffe (KBA) [7] parallel sweep algorithm on a Cartesian grid. The KBA algorithm governs Denovo's parallel behavior, including how many cores it can efficiently use. To see why parallelization in energy is needed, the KBA algorithm is explained and some illustrative Denovo calculations are presented

Angular parallelization is done by algorithmically "stacking" the calculation directions (ordinates) into a "pipe" and following more than one at one time. The transport sweeps begin with the first direction in the first quadrant and spatially move through the geometry along that ordinate in a hyper-plane fashion. Denovo uses a pipelining approach in direction such that the next angle set in an octant is started as soon as the current sweep reaches the opposite end of the geometry. The quadrants are ordered such that the next quadrant begins at the earliest possible time [21].

The spatial decomposition is done by assigning groups of spatial cells to different cores. A given problem will contain (I, J, K) total global *cells*. These are decomposed in x and y across P_I cores in the x-direction and P_J cores in the y-direction for a total of $P_I \times P_J$ cores. Each core gets a *domain* that is made up of (I_b, J_b, K) cells. Here $I_b = I/P_I$ and $J_b = J/P_J$. Each domain is algorithmically split into B_K computational blocks in the z-direction (but not split in memory, all K cells in the z-direction are on all cores). This creates *computational blocks* of size (I_b, J_b, K_b), where $K_b = K/B_K$. The computational blocks are used in the angular pipelining process. Specifying the number of cores dictates P_I and P_J, while the user is free to select B_K. See Figure 2.1 for an example of a decomposition.

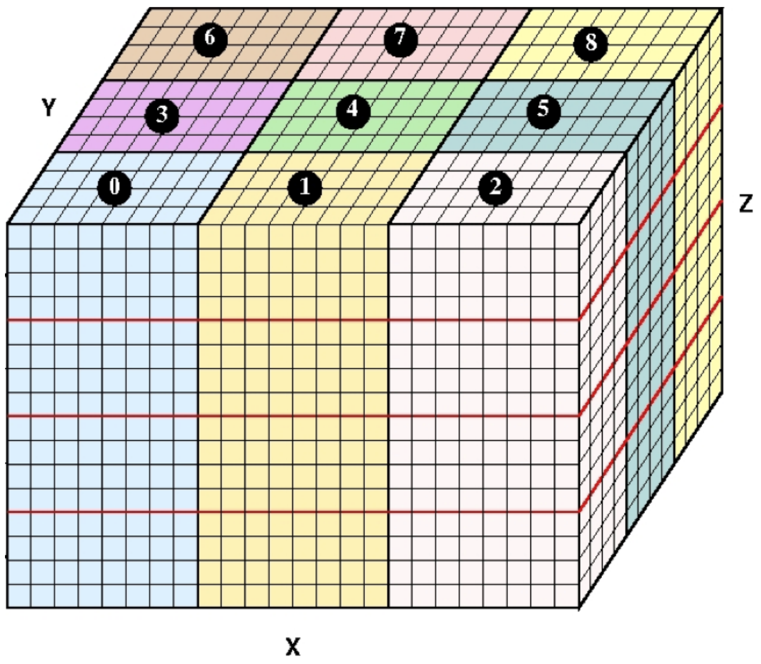


Figure 2.1 Decomposition of 3-D mesh for KBA. In this example, the grid is decomposed on nine processors. The red lines indicated computational blocks in the z-direction. Each processor has $I_b \times J_b \times K$ cells, and each computational block has size $I_b \times J_b \times K_b$.

The decomposition parameters can be used to express the theoretical efficiency of the algorithm, which is the ratio of useful computations to total computations [21], [7]:

$$\epsilon_{max} = \frac{2MB_K}{2MB_K + P_I + P_J - 2}\,, \tag{2.1}$$

where M is the number of directions in an octant. Note that for a set M, P_I, and P_J, the theoretical efficiency will be much higher when there is a larger B_k, meaning more computational blocks. Having more computational blocks creates blocks that are smaller, so work can be passed to subsequent blocks more rapidly and thus more work should be able to be done at once. See [7] for more details about the KBA algorithm.

The quality of parallelization can be measured in several ways. Strong scaling measures how the time to solution varies with the number of cores for a fixed problem size. Weak scaling measures how the time to solution varies with the addition of more cores when the problem size per core is fixed [12]. Another factor is the total number of cores that can be used efficiently.

A weak scaling study was performed by Evans et. al. on the Jaguar machine using a full-facility PWR model. The base-case time was generated using 4,096 cores with 103,716,288 unknowns, or $\sim$25,300 unknowns/core. When 40,000 cores and 1,046,879,390 unknowns ($\sim$26,200 unknowns/core or about a 3% increase in problem size/core) were used, there was a 43% increase in time to solution. The time increase for perfect weak scaling would have been about 3%. The significant increase in time can be attributed to load-balancing latencies associated with KBA [21]. Adding parallelization over energy might improve the weak scaling properties of denovo by making up for some of the latencies that come from KBA.

The author conducted a strong scaling study on the Jaguar machine that investigated the effect of B_K on efficiency. A variant of the Kobayashi benchmark problem 1 [37], which can be seen in Figure 2.2, was used where the total cross sections were changed as indicated in Table 2.1. The scattering cross sections were simply set to 0.5 $\times\Sigma$ in both the original and modified case. The calculation was done with S_{16}/P_0, step characteristic spatial differencing, and a 0.25 cm spatial resolution, giving a 400 $\times$ 400 $\times$ 400 mesh. The number of cores was varied between 12 and 3,600.

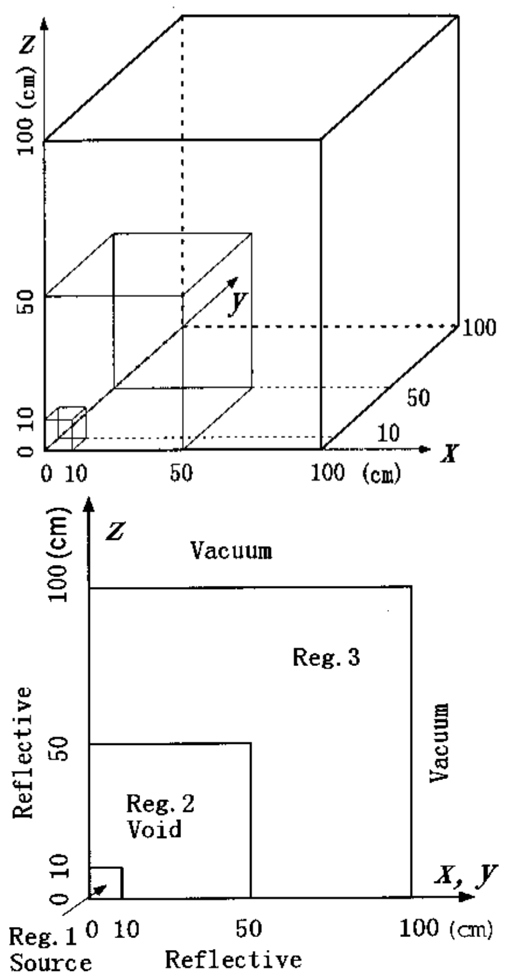


Figure 2.2 Two Views of the Kobayashi Benchmark Problem 1

The author did one set of calculations using B_K = 40, or relatively many blocks, and one set using B_K = 5, or relatively few blocks. Recall from Equation (2.1) that more computational blocks should give higher efficiency. The theoretical and actual results are plotted in Figure 2.3 as computational efficiency as a function of the number of cores used. The actual efficiency is $\frac{N_{ref}t_{ref}}{Nt}$. Here N_{ref} and t_{ref} are the reference number of cores and solve time, respectively, and N and t are for the calculation in question. The theoretical efficiency was calculated using Equation (2.1).

Table 2.1 Cross Sections and Sources for Kobayashi Benchmark Problem 1

Region (#)	S (n cm^{-3}s^{-1})	Original Σ (cm^{-1})	Modified Σ (cm^{-1})
1	1	0.1	1
2	0	10^{-4}	10^{-3}
3	0	0.1	1

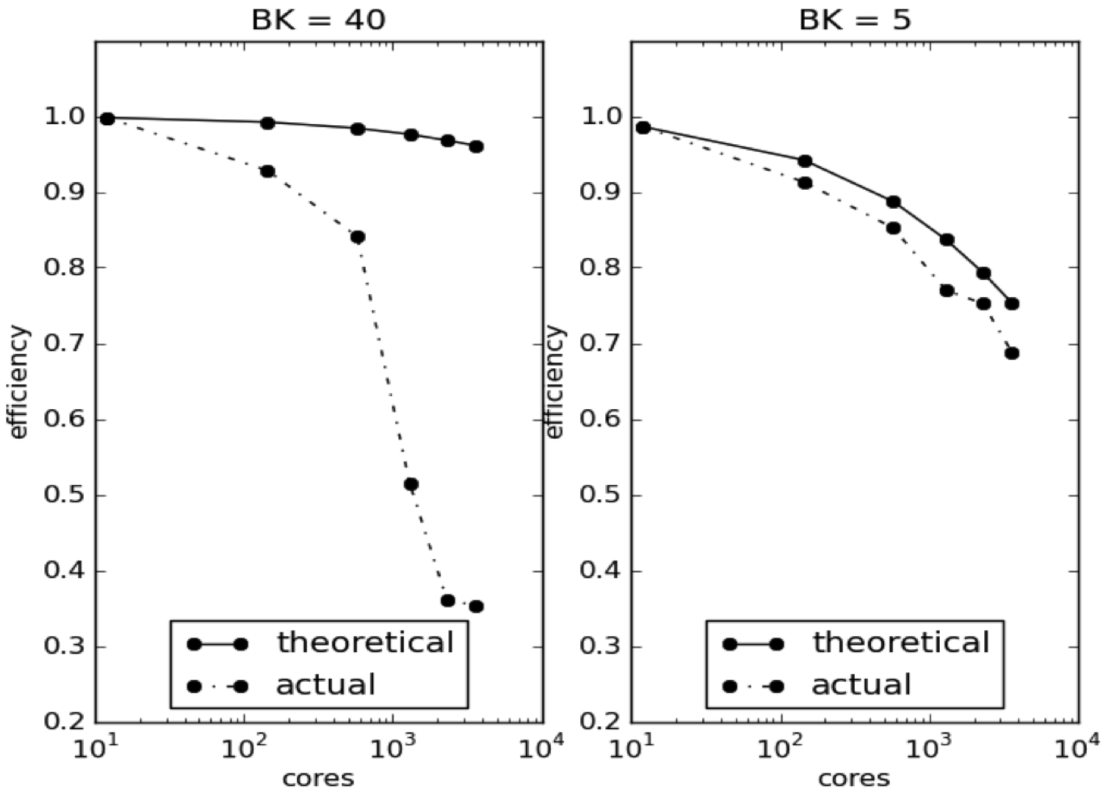


Figure 2.3 Efficiency as a Function of Number of Cores for the Modified Kobayashi 1 Problem

A few important conclusions can be drawn from the curves in Figure 2.3. One is that the increase in simultaneous work that should come from using smaller computational blocks is overwhelmed by communication latency in practice. When B_K = 40 the theoretical efficiency remains high as the number of cores increases, but the measured efficiency becomes quite low: for 3,600 cores ϵ_{max} = 0.96 and ϵ = 0.35. It is expected that more blocks would be better, but when the number of cells/block becomes too small the cores spend more time waiting to pass data than doing floating point operations (flops).

Another relevant finding is that the theoretical and actual efficiencies match when there is a minimum block size. When larger computational spaces are used (smaller B_K) communication does not become a dominant factor. This restricts the extent of spatial decomposition by limiting the number of blocks used for a given spatial mesh. For a 500M cell problem, the limitation is 15,000 - 20,000 cores.

As a result of all of this, a problem can only be decomposed so far for a given mesh before communication latency begins to dominate solution time and using more cores will be of little benefit. This is caused by the KBA algorithm. In order to take advantage of more cores for a given problem size, another area of phase space must be parallelized: energy, the remaining unparallelized dimension.

2.2 Background

To understand the new method and why it enhances Denovo's performance, some background information is required. This section contains an explanation of scattering, an overview of relevant basic solvers, and information about how we handle scattering in the transport equation. The remainder of this chapter will primarily focus on the solution of $\mathbf{A}x = b$, where $\mathbf{A}$ is $n \times n$ and the exact form of $\mathbf{A}$ and b will be specified in subsequent sections.

2.2.1 Scattering

One of the important contributions of this work is the implementation of a new way to handle upscattering in transport calculations. When neutrons scatter, several outcomes with regard to energy are possible. One outcome is that neutrons can stay within their energy group, which is called within-group scattering and is described by the $[\mathbf{S}]_{gg}$ portion of the scattering matrix. Neutrons can also move into a lower energy group, or downscatter, which is given by $[\mathbf{S}]_{gg'}$ where $g > g'$. In some cases neutrons can gain energy and upscatter, described by $[\mathbf{S}]_{gg'}$ where $g < g'$.

Material cross sections can be highly energy dependent and therefore high-resolution data in certain energy ranges may be needed to accurately capture the physics of the system. Upscattering is often important in problems where detailed information about low-energy, or thermal, neutrons is needed. In light-water reactors, thermal neutrons cause the bulk of the fission so these are systems that often need cross sections with upscattering terms.

As mentioned in Chapter 1, the transport problem is broken into outer iterations and inner iterations. The inner iterations correspond to within-group the equations. Each inner iteration converges the flux for just one group by solving a fixed source problem for that group. The fixed source contains the appropriate combination of inscattering from other groups, an external source, and a source from fission. The outer iterations are over the entire set of energy groups - all of Equations (1.15). If a problem does not have upscattering, the equations can be quickly solved using one outer iteration.

2.2.2 Solver Basics

There are two categories of methods for solving $\mathbf{A}x = b$, direct and iterative. Direct methods solve the problem by inverting $\mathbf{A}$ and setting $x = \mathbf{A}^{-1}b$. If $\mathbf{A}$ is invertible this can be done explicitly. $\mathbf{A}$ is often not invertible, so most direct methods are based on factoring the coefficient matrix $\mathbf{A}$ into matrices that are easy to invert. The problem is then solved in pieces where each factored matrix is inverted to get the final solution. An example where this is done is LU factorization. These methods are often robust and require a predictable amount of time and storage resources. However, direct methods scale poorly with problem size, becoming increasingly expensive as problems grow large [9].

Iterative methods compute a sequence of increasingly accurate approximations to the solution. They generally require less storage and take fewer operations than direct methods, though they may not be as reliable. Iterative methods are highly advantageous for large problems because direct methods become intractable for systems of the size of those of interest here. For this reason the nuclear energy industry tends to use iterative methods for transport calculations [10], [9].

2.2.2.1 Richardson Iteration

The simplest iteration scheme used by the nuclear community is source iteration (SI), also known as Richardson iteration. SI is applied to the within-group space-angle iterations. Some other method is needed to conduct outer iterations over energy. Richardson iteration can be thought of as a two-part process for the neutron transport equation, where $\bar{Q}$ includes all sources and k is the inner iteration index:

$$\mathbf{L}\psi_g^{k+1} = \mathbf{MS}\phi_g^k + \bar{Q} \, , \tag{2.2}$$

$$\phi_g^{k+1} = \mathbf{D}\psi_g^{k+1} \, . \tag{2.3}$$

The spectral radius determines the speed of convergence and is $c = \frac{\Sigma_s}{\Sigma}$. For problems that are dominated by scattering, SI will converge very slowly [21]. While this is the simplest iterative method used, there are much more sophisticated methods available.

2.2.2.2 Gauss Seidel

Gauss Seidel (GS) is known as the method of successive displacements [41]. It is commonly used as the outer iteration method over energy and can be written as follows where j is the outer iteration index:

$$\mathbf{L}[\psi]_g^{j+1} = [\mathbf{M}][\mathbf{S}]_{gg}[\phi]_g^{j+1} + [\mathbf{M}]\big(\sum_{g'=1}^{g-1}[\mathbf{S}]_{gg'}[\phi]_{g'}^{j+1} + \sum_{g'=g+1}^{G}[\mathbf{S}]_{gg'}[\phi]_{g'}^{j} + [q_e]_g\big) \, . \tag{2.4}$$

The GS method is implicit and contains terms on the right hand side at both the new and old iteration levels. Once the space-angle iterations for each upscattering group are completed for one outer iteration, the right hand side is recalculated and the within-group calculations are repeated. This goes on until the entire system converges [21].

Gauss Seidel is unconditionally stable, but may converge very slowly. The convergence of GS is governed by its spectral radius, ρ [41]. The spectral radius of Gauss Seidel has been found for various materials by solving the eigenvalue problem $(\mathbf{T} - \mathbf{S}_D)^{-1}\mathbf{S}_U\xi = \rho\xi$, where $\mathbf{T}$ is the matrix of total cross sections and ξ is the eigenvector. This eigenvalue problem comes from Fourier analysis. The spectral radii for some materials of practical interest determined using cross sections with 41 thermal upscattering groups are: graphite = 0.9984, heavy water = 0.9998, and iron = 0.6120 [3]. Problems containing graphite or heavy water would converge very slowly if GS were used.

2.2.2.3 Krylov methods

Krylov methods[1] are a powerful class of subspace methods that can be ideal for solving various types of linear and eigenvalue problems. A Krylov method solves $\mathbf{A}x = b$ by building a solution from a Krylov subspace generated by an iteration vector v_1. At iteration k, the subspace is:

$$\mathcal{K}_k(\mathbf{A}, v_1) \equiv span\{v_1, \mathbf{A}v_1, \mathbf{A}^2 v_1, ..., \mathbf{A}^{k-1} v_1\} \ . \tag{2.5}$$

The choice of v_1 varies, but $v_1 = b$ is common. When the problem is presented as $\mathbf{A}z = r_0$ where $r_0 \equiv b - \mathbf{A}x_0$ and the solution is $x_k = x_0 + z$, then v_1 is often chosen to be r_0. Note that x_0 is the initial guess and x_k is the solution approximation at step k [33].

The dimension of a Krylov space is bounded by n because Krylov methods will give the exact solution after n iterations if roundoff error in neglected. Interestingly, this technically makes Krylov methods a hybrid of direct and iterative methods because an exact answer can be obtained in a set number of steps. Krylov subspace methods are nevertheless generally classified as iterative methods [10].

Krylov methods are particularly useful in a few pertinent cases. One is when $\mathbf{A}$ is very large because fewer operations are required than traditional inversion methods like Gaussian elimination. Another is when $\mathbf{A}$ is not explicitly formed because Krylov methods only need the action of $\mathbf{A}$. Finally, Krylov methods are ideal when $\mathbf{A}$ is sparse because the number of operations are low for each matrix-vector multiplication. For deterministic transport codes, $\mathbf{A}$ is typically quite large, fairly sparse, and only its action is needed. The action of $\mathbf{A}$ is implemented through the transport sweeps [43], [33].

In the last few decades Krylov methods have been used widely to solve problems with appropriate properties for several reasons. Krylov methods are robust; the existence and uniqueness

[1]For a detailed discussion of Krylov methods and how they work see Appendix B

of the solution can be established; typically far fewer than n iterations are needed when they are used as iterative solvers; they can be preconditioned to significantly reduce time to solution; only matrix-vector products are required; explicit construction of intermediate residuals is not needed; and they have been found to be highly efficient in practice [33], [36].

There are, however, a few drawbacks. In some cases Krylov methods can be very slow to converge, causing large subspaces to be generated and thus becoming prohibitively expensive in terms of storage size and cost of computation. Some methods can be restarted after m steps[2] to alleviate this problem, keeping the maximum subspace size below $\mathcal{K}_{m+1}$. The relatively inexpensive restart techniques can reduce the storage requirements and computational costs associated with a slow-converging problem such that they are tractable. Preconditioners can also help by reducing the number of iterations needed. Development of appropriate preconditioners is an active area of research [70], [33], [36].

In the 1970s interest in Krylov methods in the wider computational community began to increase after it was demonstrated that these methods can converge quickly. At first Krylov methods were restricted to problems where $\mathbf{A}$ is symmetric positive definite (SPD). These are methods such as conjugate gradient (CG), MINRES, SYMMLQ, and others. Then, Krylov methods for non-symmetric matrices became a focus, including the Generalized Minimum Residual (GMRES) and BiConjugate Gradient Stabilized (BiCGSTAB) methods [8], [9]. The transport problem has characteristics that make it well suited for solution with Krylov methods, and these methods are used in several novel ways throughout this work.

2.2.3 Current Scattering Solution Methods

All three of the methods just discussed are used in many transport codes, including Denovo. The inner iterations are done with either SI or a Krylov method and the outer iterations are done with Gauss Seidel. In this chapter the energy block consists of the upscattering block. The operator notation developed in Chapter 1 is used throughout the balance of this work. The fixed source

[2]For information about restarted Krylov methods see Appendix B

problem will be presented here, but the discussion can be easily extended to eigenvalue calculations by using the fission source for each group in place of the external source.

A simple five group example matrix will be used for reference throughout this section. The matrix illustrates the structure of S when groups 1 and 2 have only downscattering and the upscatter block is defined over the range [g_1 = 3, g_2 = 5]:

$$\mathbf{S} = \begin{pmatrix} [\mathbf{S}]_{11} & 0 & 0 & 0 & 0 \\ [\mathbf{S}]_{21} & [\mathbf{S}]_{22} & 0 & 0 & 0 \\ [\mathbf{S}]_{31} & [\mathbf{S}]_{32} & [\mathbf{S}]_{33} & [\mathbf{S}]_{34} & [\mathbf{S}]_{35} \\ [\mathbf{S}]_{41} & [\mathbf{S}]_{42} & [\mathbf{S}]_{43} & [\mathbf{S}]_{44} & [\mathbf{S}]_{45} \\ [\mathbf{S}]_{51} & [\mathbf{S}]_{52} & [\mathbf{S}]_{53} & [\mathbf{S}]_{54} & [\mathbf{S}]_{55} \end{pmatrix} . \tag{2.6}$$

2.2.3.1 Downscattering

Groups that only contain downscattering can be solved using forward substitution. The within-group transport equations are the same as in Equation (2.4), but the upscattering term goes away. The equation is combined with Equation (2.3) and rearranged to eliminate $[\psi]_g$ such that the unknown quantity is $[\phi]_g$.

Each group equation is solved sequentially starting with group 1, so all group moments on the right hand side have already converged for the $j + 1$ outer iterate. To help clarify notation, the moments being iterated upon will be designated ϕ^*, the moments which are known at the $j + 1$ iterate will be ϕ^{new} and those from j will be ϕ^{old}. Together this looks like:

$$[\phi]^*_g = \mathbf{DL^{-1}}[\mathbf{M}][\mathbf{S}]_{gg}[\phi]^*_g + \mathbf{DL^{-1}}[\mathbf{M}]\Big(\sum_{g'=1}^{g-1}[\mathbf{S}]_{gg'}[\phi]^{new}_{g'} + [q_e]_g\Big) \; ; \tag{2.7}$$

$$\underbrace{\Big(\mathbf{I} - \mathbf{DL^{-1}}[\mathbf{M}][\mathbf{S}]_{gg}\Big)}_{\tilde{\mathbf{A}}}[\phi]^*_g = \underbrace{\mathbf{DL^{-1}}[\mathbf{M}]\Big(\sum_{g'=1}^{g-1}[\mathbf{S}]_{gg'}[\phi]^{new}_{g'} + [q_e]_g\Big)}_{\tilde{b}} \; . \tag{2.8}$$

For example matrix (2.6), the first and second groups would be solved using this strategy. That means forward substitution would be used to solve this part of the scattering matrix:

$$\mathbf{S_{down}} = \begin{pmatrix} [\mathbf{S}]_{11} & 0 & 0 & 0 & 0 \\ [\mathbf{S}]_{21} & [\mathbf{S}]_{22} & 0 & 0 & 0 \end{pmatrix} . \tag{2.9}$$

For group 2, Equation (2.8) would be:

$$\big(\mathbf{I} - \mathbf{DL^{-1}}[\mathbf{M}][\mathbf{S}]_{22}\big)[\phi]_2^* = \mathbf{DL^{-1}}[\mathbf{M}]\big([\mathbf{S}]_{21}[\phi]_1^{new} + [q_e]_1\big) . \tag{2.10}$$

Once the equations have been arranged in this way, a within-group solver is used to find the $j+1$ value for $[\phi]_g$. This iterative method could be source iteration, a Krylov method, or some other choice. When using a Krylov solver, the first step is always to calculate $\tilde{b}$.

In Denovo, Aztec [30] provides the linear within-group solver. The Aztec solver is given an operator that implements the action of $\tilde{\mathbf{A}}$, the right hand side $\tilde{b}$, and an iteration/solution vector v. The action of $\tilde{\mathbf{A}}$ is implemented by doing the following for a group g:

1. matrix-vector multiply: $y_g = [\mathbf{M}][\mathbf{S}]_{gg}v_g$,

2. sweep: $z_g = \mathbf{DL}^{-1}y_g$,

3. return: $v_g \leftarrow v_g - z_g$.

In this implementation the Aztec solver iterates on v_g, which represents $[\phi]_g$, until it is converged for that group. Once the updated moments are returned, the next group is iterated upon.

An iterative solver is used for even these simple forward substitution calculations because $\tilde{\mathbf{A}}$ is never explicitly formed and thus Equation (2.8) cannot be solved directly. Only one outer iteration is required in fixed source problems when there is no upscattering because once each group is converged it is not changed by lower energy groups.

Another way to look at this is that the right hand side of the equation is not changed by subsequent iterations, so solving $[\tilde{\mathbf{A}}]_g[\phi]_g = [\tilde{b}]_g$ will give the same answer both before and after all subsequent downscattering moments, $[\phi]_{g'}, g' \neq g$, are determined. Because this procedure is so straightforward, there is no motivation to parallelize it in energy for fixed source problems.

2.2.3.2 Upscattering

When upscattering is present, the lower energy groups do influence the higher energy groups so multiple outer iterations are needed. The within-group equations for groups $[g_1, g_2]$ now contain contributions from both higher and lower energy groups.

The Gauss-Seidel (GS) method is commonly used to handle the upscattering block and, using the same notation as the downscattering section, can be written as follows:

$$\underbrace{\left(\mathbf{I} - \mathbf{DL^{-1}}[\mathbf{M}][\mathbf{S}]_{gg}\right)}_{\tilde{\mathbf{A}}}[\phi]^*_g = \underbrace{\mathbf{DL^{-1}}[\mathbf{M}]\Big(\sum_{g'=1}^{g-1}[\mathbf{S}]_{gg'}[\phi]^{new}_{g'} + \sum_{g'=g+1}^{g2}[\mathbf{S}]_{gg'}[\phi]^{old}_{g'} + [q_e]_g\Big)}_{\tilde{b}} \; . \tag{2.11}$$

Because lower energy groups that haven't been solved yet contribute neutrons to higher energy groups that have already been solved, the upscatter block becomes its own iteration loop. The solution procedure is exactly the same as with downscattering only, except that once the $[\phi]_{g_1}$ to $[\phi]_{g_2}$ moments are converged $\tilde{b}$ is recalculated and the outer iterations over the upscatter block are repeated. Most deterministic transport codes use GS for upscattering. In GS, all upscattering groups are coupled together so parallelization over energy is not possible [21].

2.3 Past Work

Now that the basics of inner and outer iterations have been described, past solution methods can be understood.

2.3.1 Inner Iterations

Source iteration was historically the inner iteration method of choice. SI is still often used, but now it is typically accelerated to improve convergence. Unfortunately, even accelerated SI is not always fast or robust enough for new calculations of interest. The slow convergence of SI is part of what motivated interest in Krylov methods.

In 1977 CG, which requires $\mathbf{A}$ to be symmetric positive definite (SPD), was used in solving the transport equation for the first time [42]. $\mathbf{A}$ is only SPD for the discretized transport equation when a symmetric quadrature set is used for the S_N approximation and the scattering is isotropic.

Otherwise, the matrix is non-symmetric. GMRES, which works for non-symmetric systems, was applied in a transport problem with anisotropic scattering for the first time in 1991 [3]. It was not until Krylov methods for non-symmetric matrices were developed, restart methods became well known, and computer architecture could accommodate the storage required by Krylov subspaces needed in transport problems that the nuclear community began more widely using Krylov methods.

With a few exceptions, Krylov methods have only been applied to the inner iterations for neutron transport. In some cases Krylov methods have been used as stand-alone iterative schemes to perform the within-group solves. In other cases they have been used as one part of the inner iteration process. For example, diffusion or transport synthetic acceleration (D/TSA) is often chosen to precondition SI as the inner iteration solver, where CG is used to solve the D/TSA portion of the calculation [26]. Krylov methods have also been applied to the diffusion equation when a few groups have been used [64], [66].

Krylov methods have so far been largely restricted to inner iterations or few-group diffusion problems primarily because of hardware limitations and momentum of existing code structure. A Krylov subspace of size 30 can use a lot of memory when v holds many variables. Until very recently, computers were not large enough to handle subspace iteration methods for a 3-D, anisotropic transport problem with many energy groups.

Further, the inner-outer iteration structure has been successfully used in computational neutronics for quite some time, and it can take substantial effort to modify the fundamental solution processes in large codes. Often there is a time lag between when new methods are developed and when they find wide-spread incorporation, which adds to the delay in implementing new methods in existing codes.

2.3.2 Outer Iterations

When upscattering is present, Gauss Seidel is almost universally used as the outer iteration solver. As noted above, GS can converge very slowly for some systems containing materials like heavy water, which are of interest to the nuclear community. Such slow convergence is part of the motivation for this work, which develops a method to replace GS for the upscatter block.

There is one case where a Krylov method was used in a fixed source problem for something besides within-group solves. In 2004 Warsa et. al. used restarted flexible GMRES as an intermediate-level iteration over upscatter groups (what this chapter calls the outer iterations). In one upscattering case they used a Krylov solver for within-group iterations, FGMRES(m) over the upscatter block, and an implicitly restarted Arnoldi method for the eigenvalue outer iteration. They compared this to using block GS for the upscatter intermediate-level iteration. They found that using a Krylov solver rather than block Gauss Seidel for upscatter was faster. This is the only work found by the author that used a Krylov method for upscattering [70].

Choosing to use an intermediate-level Krylov solver while still doing inner iterations with a Krylov solver is a bit of a strange choice. As far as the author can tell, this choice was motivated by the desire to minimize changes to the code. By keeping the traditional inner-outer iteration method and adding a Krylov layer in between, very little code modification was needed.

The methods that have been used in the past dictate an inner-outer iteration structure and limit parallelization to space and angle. The author was unable to find transport codes that have been parallelized in energy nor ones that deal with upscattering in a way different from what has been explained above.

2.4 Multigroup Krylov Solver

The “Denovo and Parallelism” section demonstrated that Denovo is limited in the number of cores it can use effectively. In addition to enhanced parallelism, there is also a need for faster solution algorithms for all parts of the code so Denovo can be used to solve very large systems. To address these issues, a new solver was implemented that applies a Krylov method to an entire block of energy groups rather than to only one group at a time. This replaces the inner-outer iteration structure with just one level of iteration because all groups can be converged simultaneously.

2.4.1 Method

The first goal of this project is to decouple energy groups so that the energy portion of phase space can be parallelized. This goal can be achieved in a straightforward way for an upscattering block by solving the entire block as one entity with a Krylov method rather than solving one group at a time.

In the new method, the downscatter groups are treated the same way as before. For the example case, $[\phi]_1$ and $[\phi]_2$ are solved in the downscattering only calculations and are known by the time the upscatter block is reached. The portion of the scattering matrix shown in Equation (2.12), operates on these moments.

$$\mathbf{S_{up\_source}} = \begin{pmatrix} [\mathbf{S}]_{31} & [\mathbf{S}]_{32} \\ [\mathbf{S}]_{41} & [\mathbf{S}]_{42} \\ [\mathbf{S}]_{51} & [\mathbf{S}]_{52} \end{pmatrix} , \tag{2.12}$$

The remainder of the scattering matrix, shown in Equation (2.13), operates on $[\phi]_{3-5}$. The idea behind the new solver is to treat $[\phi]_{3-5}$ as one vector to be solved at one time, rather than three vectors to be solved in series.

$$\mathbf{S_{up\_block}} = \begin{pmatrix} [\mathbf{S}]_{33} & [\mathbf{S}]_{34} & [\mathbf{S}]_{35} \\ [\mathbf{S}]_{43} & [\mathbf{S}]_{44} & [\mathbf{S}]_{45} \\ [\mathbf{S}]_{53} & [\mathbf{S}]_{54} & [\mathbf{S}]_{55} \end{pmatrix} . \tag{2.13}$$

Instead of $\tilde{G} = g_2 - g_1$ separate within-group upscattering equations, the new method has *one* block upscattering equation. To do this, Equation (2.11) is modified to apply to a block instead of a series of groups. The entire upscatter block, or matrix (2.13), moves to the left because it acts on the upscattering-block-vector being iterated upon ($[\phi]_{3-5} = \phi^*$). The upscatter source, $\mathrm{S}_{\text{up\_source}}$ of matrix (2.12) remains on the right because it is only acting on the downscattering groups that have been solved at the current iteration level ($[\phi]_{1-2} = \phi^{new}$). The form of the new strategy is shown in Equation (2.14) and looks very much like Equation (2.11).

$$\underbrace{\left(\mathbf{I} - \mathbf{DL^{-1}MS}_{\text{up\_block}}\right)}_{\tilde{\mathbf{A}}} \phi^* = \underbrace{\mathbf{DL^{-1}M}\left(\mathrm{S}_{\text{up\_source}}\phi^{new} + q_e\right)}_{\tilde{\mathbf{b}}} \tag{2.14}$$

Applying the Krylov solver to this system is very similar to what was done before, but now the action of $\tilde{\mathbf{A}}$, or the matrix-vector multiply, is applied to the entire block at once instead of just one group:

1. matrix-vector multiply: $y = \mathbf{MS}_{\text{up\_block}}v$,

2. sweep: $z = \mathbf{DL}^{-1}y$,

3. return: $v \leftarrow v - z$.

All that has happened to allow for solving the entire block at once is the redefinition of the iteration vector. The scattering terms are still coupled together through physical cross sections, but the way they contribute to one another is addressed all at one time through Krylov iteration rather than serially through Gauss Seidel iteration.

This new formulation allows for parallelization of the upscatter groups in energy. The matrix vector multiply can be done at the same time for all upscattering groups for each iteration. To see how this works, refer to Figure 2.4. Each color can do its part of the matrix vector multiply at the same time as the other colors. After the separate multiplies, there is a global reduction so that every color has the updated s, or the same brown box. The rest of the application of A is straightforward and doesn't require any inter-color communication [22].

$$s_g = \mathbf{S}_{g1}v_1 + \mathbf{S}_{g2}v_{g2} + \cdots + \mathbf{S}_{g5}v_5$$

$$\begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \end{pmatrix} = \begin{pmatrix} [\mathbf{S}]_{11} & 0 & 0 & 0 & 0 \\ [\mathbf{S}]_{21} & [\mathbf{S}]_{22} & 0 & 0 & 0 \\ [\mathbf{S}]_{31} & [\mathbf{S}]_{32} & [\mathbf{S}]_{33} & [\mathbf{S}]_{34} & [\mathbf{S}]_{35} \\ [\mathbf{S}]_{41} & [\mathbf{S}]_{42} & [\mathbf{S}]_{43} & [\mathbf{S}]_{44} & [\mathbf{S}]_{45} \\ [\mathbf{S}]_{51} & [\mathbf{S}]_{52} & [\mathbf{S}]_{53} & [\mathbf{S}]_{54} & [\mathbf{S}]_{55} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{pmatrix}$$

Figure 2.4 Parallel Implementation of Upscattering Matrix-Vector Multiply

To accomplish the parallelization, the problem is broken up into *energy sets*. There can be between 1 and $\tilde{G}$ sets; the groups are distributed evenly among sets. Each energy set gets the entire spatial mesh, so the spatial decomposition does not change. Because the full mesh is on each set, KBA can be used just like before and it does not have to cross any spatial boundaries or scale to where it becomes inefficient. A picture of the decomposition can be seen in Figure 2.5.

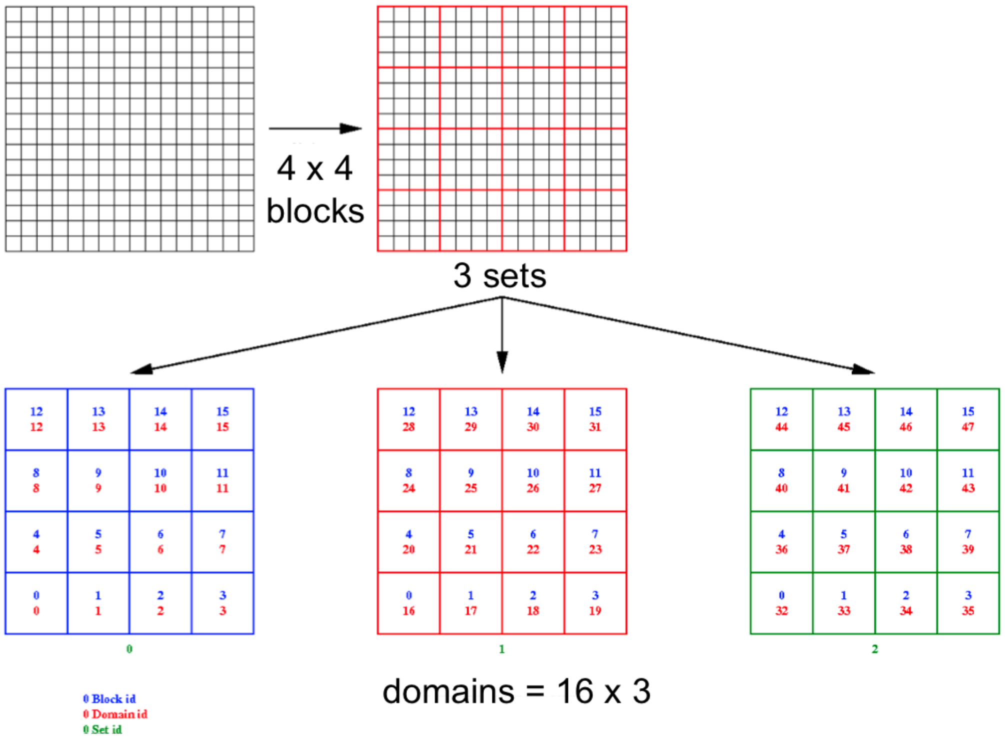


Figure 2.5 Energy Set Decomposition Added to Denovo [20]

It was demonstrated in Section 1 that KBA becomes latency dominated when block size becomes small, limiting the number of cores that can be used for a given problem. With energy decomposition, more cores can be used without requiring KBA to scale further because scaling can be done across energy instead of space-angle.

The number of cores is set by the number of domains. Previously, the number of domains was equal to the number of spatial blocks. Now it is the number of energy sets $\times$ the number of spatial blocks. With the addition of sets Denovo can use hundreds of thousands of cores. For example, with 10,000 spatial blocks (which KBA can handle) and 10 energy sets, Denovo can use 100,000 cores [20], [22]. KBA would not be able to efficiently use 100,000 spatial blocks. This dramatically increases the scalability of Denovo.

Before this work, no one had decoupled the transport equation in energy nor used one iteration level instead of two. We can now do this because we have machines large enough to store Krylov subspaces made from multiple group-sized iteration vectors, and these computers are large enough to warrant parallelization in energy .

2.4.2 Results

The new method, which will be referred to as multigroup (MG) Krylov or block Krylov, has been implemented in Denovo. A few different calculations were performed to investigate the improvement gained from the new method. The benefit of decomposing in energy was investigated from the standpoint of both strong and weak scaling. Some problems were run without energy decomposition, i.e. using one energy set, to compare using Krylov instead of GS for the upscatter groups.

To consider these issues, two test problems were chosen. One was simply a small toy problem and the other was a full scale reactor calculation. The results will be presented first from the perspective of strong scaling, then MG Krylov and GS are compared, and finally weak scaling is discussed.

2.4.2.1 Strong Scaling

The toy problem was designed to give an initial indication of whether or not the new method was effective. It does not provide a good measure of scaling, but a qualitative indication of whether the energy set decomposition works. The problem is a cube, half of which is composed of iron and the other half is graphite, discretized with a $100\times$ 100 $\times$ 100 orthogonal mesh and decomposed into four spatial blocks. It is a 26 group problem with 13 upscattering groups and vacuum boundary conditions. An isotropic source is assigned everywhere; the problem used P_0, and S_4. This composition was chosen for several reasons. Both materials are commonly found in nuclear reactors; graphite is highly scattering and creates a system with a large spectral radius for GS; and the cross sections were available with multiple upscattering groups.

It should be noted that it is not expected that this problem will scale particularly well because dividing 13 groups into energy sets will result in load imbalance. There is no way to evenly divide the number of sets among cores, meaning at least one processor always has one more group than the others. The cores with fewer groups must wait for the cores with more groups. The more sets used, the larger the number of cores waiting.

The number of energy sets was varied from one to 13. The test problems were run on the small Orthanc cluster at ORNL. The results can be seen in Figure 2.6. The downscattering groups were solved with forward substitution as before, and the upscattering was solved with the Krylov multigroup solver. The upscattering groups converged in 104 GMRES iterations.

The results indicate that decomposing over energy does reduce solution time. Going from one to two energy sets, or doubling the number of cores, reduces the time to solution by 37%. Using four energy sets reduces the solve time by 56%. Using more than a few energy sets for this problem does not give significant payback in solve time. This is not too surprising as it was not expected that this problem would scale well because of load balancing issues. The point, however, is that some speedup was seen and the decomposition and solver are working correctly.

A more realistic problem, a full pressurized water reactor (PWR) core, was tested by Evans on the Jaguar machine. This problem was used to study strong scaling in a more quantitative way than the toy problem. This PWR has 289 17×17 assemblies (157 fuel, 132 reflector). There are low, medium, and high enrichment fuel pins. A 2×2 spatial discretization per homogenized fuel pin was used giving 233,858,800 cells. S_{12} was selected as the angular discretization. For the strong scaling study, 44 energy groups were used.

Three decomposition combinations were used, detailed in Table 2.2. It should be noted that Jaguar requires decompositions to be done in a certain way, which determines the number of domains that can be used. For example, exactly doubling domain size is typically not possible so the closest available approximation is chosen instead. Here 9,024 spatial domains was the closest available approximation to 10,200, and 5,040 domains was closest to half.

Case III was compared to Case I to examine the effect of doubling the number of energy groups with the spatial decomposition fixed. Case III was compared to Case II to examine the effect of

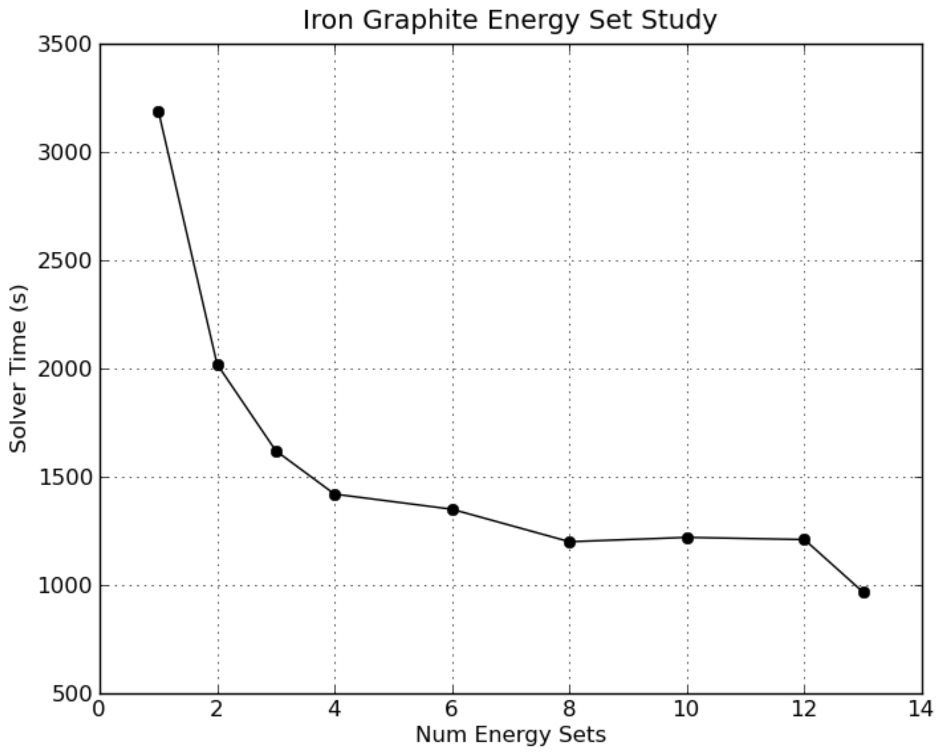


Figure 2.6 Solver Time vs. Energy Decomposition for the Iron-Graphite Toy Problem

Table 2.2 Strong Scaling for the Full PWR Problem Decomposition

Case	Blocks	Sets	Domains	Solver Time (min)
I	10,200	11	112,200	49.61
II	5,040	22	110,880	53.61
III	9,024	22	198,528	34.99

doubling the spatial decomposition with the energy sets fixed. If perfect, i.e. linear, scaling were obtained, the solve time would be $(\frac{\text{base\_domains}}{\text{used\_domains}}) \times \text{base\_time}$. The efficiency can be defined as the $\frac{\text{t\_perfect}}{\text{t\_actual}}$.

Table 2.3 Full PWR Case III Comparison

Compare To	Perfect	Actual	Efficiency
I (energy)	28.04	34.99	0.80
II (space)	29.94	34.99	0.86

The comparison results are shown in Table 2.3. The timing results, plotted with a linear scaling line, are shown in Figure 2.7. Both of the results are quite good. Doubling the energy sets resulted in an efficiency of 80%. This is slightly less than the efficiency of 86% that came from doubling the spatial decomposition. Neither case differs drastically from linear scaling.

Communication optimization was added after these initial results were obtained Case I and Case III were re-run. The optimization greatly improved the Case I results. Unfortunately, there was almost no difference in run time for Case II and the scaling became very poor. A plot of the new results and comparison to linear scaling can be seen in Figure 2.8.

There are a few reasons that the improvement did not scale to 200,000 cores. It is likely that the problem just is not large enough to demonstrate strong scaling. The block decomposition was not optimal. There were communication collisions on the torus across the full machine that significantly slow down communication. The multiset communication is also not optimal. Some changes have been identified that should reduce the cost of multiset communication by about half. If these issues are rectified then better strong scaling behavior would likely result.

Despite this, one of the most important results from these calculations is the number of cores that were used successfully. Denovo was able to use 100,00 to 200,000 cores when it was previously limited to about 20,000. These results demonstrate that the multigroup Krylov solver allows for the use of leadership-class machines.

2.4.2.2 Gauss Seidel vs. Block Krylov

The two test problems were also used to compare using GS to MG Krylov for multigroup upscattering. For this study the iron-graphite problem was run on a dual core MacBook Pro laptop.

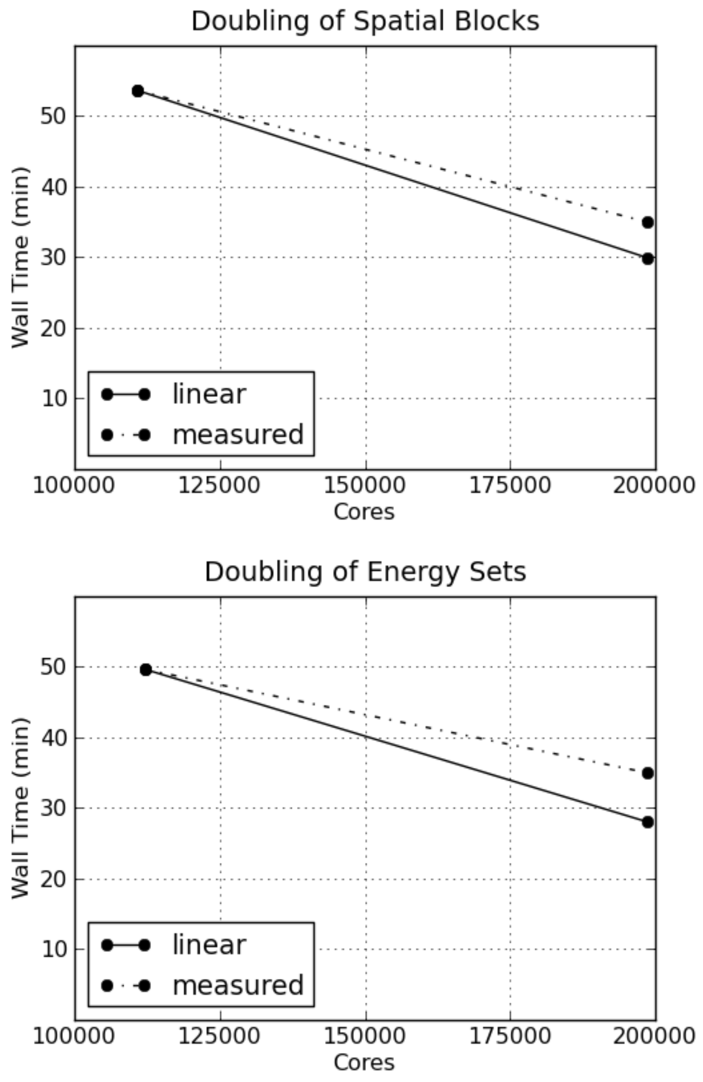


Figure 2.7 Strong Scaling Study for the Full PWR Calculation with 44 Groups

It was reduced in size to $50 \times 50 \times 50$ and decomposed into only two spatial domains to be able to run on the two cores available.

The problem took 3.51×10^3 seconds using MG Krylov with one energy set; the upscattering block converged in 43 GMRES iterations. With GS, the calculation took 2.87×10^4 seconds, or

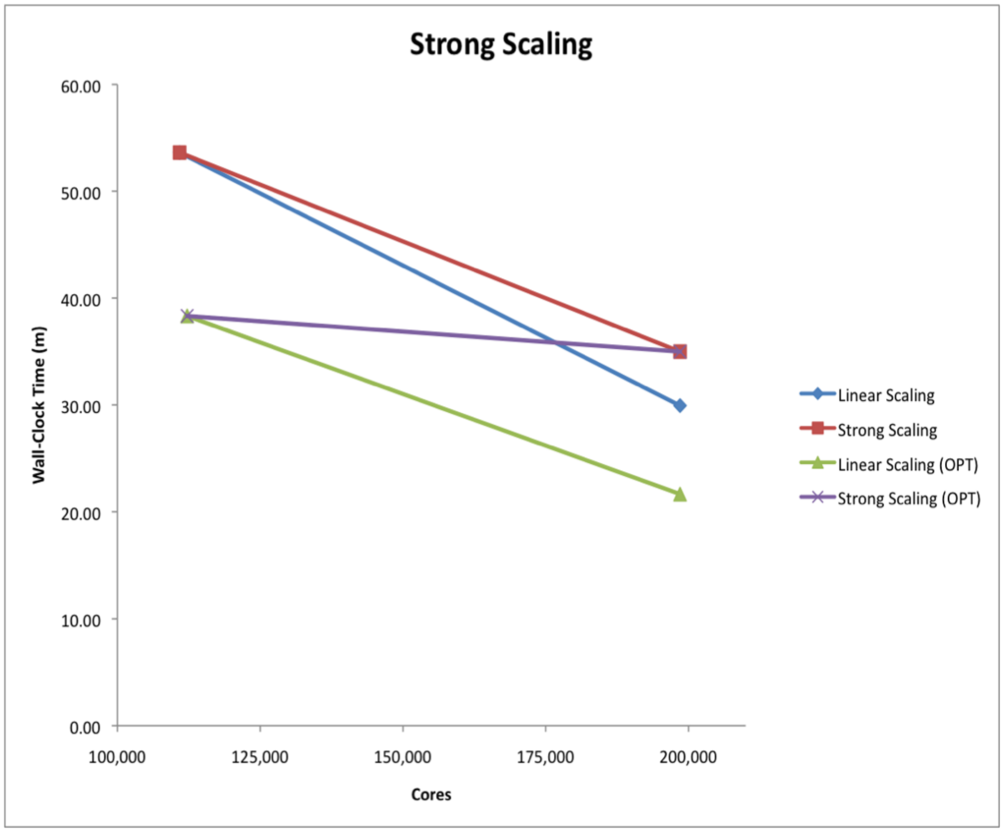


Figure 2.8 Strong Scaling Study With Communication Optimization for the Full PWR Calculation with 44 Groups

718% longer. This case converged in 44 Gauss Seidel iterations, each of which took about 124 GMRES inner iterations, or ∼5455 GMRES iterations all together. Note that the Krylov iterations in MG Krylov needed a subspace with a length dimension of 13 energy groups while the Krylov iterations in GS had a length of 1 energy group. This means the cost of the GMRES applications is not the same. Despite this, GS was very clearly more costly.

Table 2.4 shows the results for the full PWR problem. The goal was to compare MG Krylov and GS using the same number of domains, which required a different spatial decomposition. For this problem only two energy groups were used to reduce the total problem size. Power Iteration (described in Chapter 3) was used for the eigenvalue solver in both cases; GS used GMRES for the inner iterations. For this problem GS was even accelerated with a two-grid preconditioner (TTG;

Table 2.4 MG Krylov - GS Comparison for the Full PWR

Solvers	Blocks	Sets	Domains	Solver Time (min)
PI + TTG GS	17,424	1	17,242	11.00
PI + MG Krylov	10,200	2	20,400	3.03

details are given in Section 4.2.4.2). While GS had 15.5% fewer domains than MG Krylov, it took 263% longer. Clearly the MG Krylov method was much faster than GS for this problem, even when GS was preconditioned and MG Krylov was not.

In both test cases the new Krylov multigroup solver significantly outperformed Gauss Seidel. The Krylov method took much less time and many fewer iterations, even when Gauss Seidel was accelerated. This demonstrates that the new multigroup solver provides substantial convergence improvements and that will help solve large and challenging problems.

2.4.2.3 Weak Scaling

Finally, the full PWR problem was used for a weak scaling study. The two-group, 17,424-block, GS calculation from the previous subsection (PI + 2-grid preconditioned GS in Table 2.4) was compared to the 44-group, 11-set, 10,200-block, MG Krylov calculation with the optimized communication strategy. With this improvement the runtime using MG Krylov was reduced to 38.33 minutes.

To get a sense of the problem size there were there are 78,576,556,800 unknowns when using two energy groups and 17,424 cores; there were 1,728,684,249,900 unknowns when using 44 groups and 112,200 cores. Typically with weak scaling the problem size/core is kept constant. In this case the problem size/core was not constant so an adjustment factor was included. The number of energy groups were used to weight the problem size/core and that was used to get an adjusted solve time of 11.21 minutes for the MG Krylov calculation: $[(\frac{44}{2})(\frac{17{,}424}{112{,}200})]^{-1} \times 38.33 = 11.21$.

Efficiency for weak scaling is calculated by taking the ratio of the MG Krylov adjusted time to the reference time, or 11.21 over 11.00 giving an efficiency of 1.019. A plot of the weak scaling is shown in Figure 2.9. Note that perfect weak scaling would maintain an efficiency of 1.

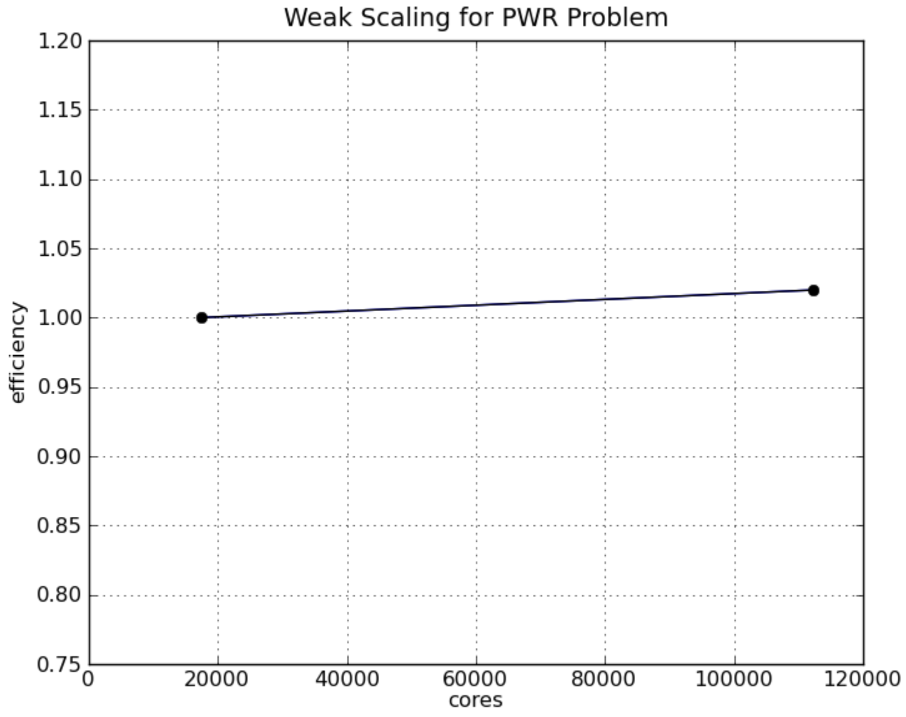


Figure 2.9 Weak Scaling Study for the Full PWR Calculation

2.4.3 Implications

In general, Krylov methods converge more quickly than GS and are extremely robust [41]. The test just discussed showed that the new method provides a time benefit both from the parallelization in energy that is enabled by this method and from using a Krylov solver for the equivalent of the outer iterations. The author was unable to find another deterministic transport code that decomposes the upscatter calculation in energy nor one that only uses one iteration level for the upscattering block. This decomposition is a new contribution to the field and helps alleviate the scaling limitations discussed earlier.

There are a few reasons this has not been pursued before, primarily based on computer architecture and existing code capability. Memory has only become large enough to accommodate the necessary Krylov subspaces recently. Only in the last several years have there been computers with enough processing capability to allow for scaling to hundreds of thousands of cores, motivating energy decomposition. In addition, the inner-outer paradigm is at the core of most transport codes and in many cases it would take substantial effort to modify that structure.

Note that an explicit block Jacobi method, which converges more slowly that GS by a factor of two, could have been used previously to decouple energy groups because it does not have the large memory footprint associated with Krylov methods. However, there was no need to scale to so many cores and so there was no motivation for this development. Now that the scaling need exists, there is sufficient memory for Krylov methods. Krylov methods are the clear choice because they often converge faster that GS.

Overall, the strong scaling studies showed that the block Krylov solver works and provides reasonably good strong scaling in both energy and space, at least out to 100,000 cores. Each test problem illustrated that the block Krylov method solves fixed source problems with upscattering more quickly than the Gauss Seidel method. The weak scaling study showed that when compared to GS, the MG Krylov method has very good weak scaling properties. This new method decisively accomplishes the goal of accelerating transport calculations by using new computers fully.

+

Chapter 3

Eigenvalue Acceleration

Nuclear engineers are often concerned with finding the criticality state of a reactor. The term critical means that the fission chain reaction is self-sustaining, i.e. every neutron produces one new neutron. In the context of Equation 1.1, this corresponds to $k = 1$. If $k < 1$, the reaction is subcritical and the total number of neutrons decreases over time. If $k > 1$, the reaction is supercritical and the total number of neutrons increases over time [17]. Steady-state reactor behavior is generally the operation mode of interest, and the typical concern is only the dominant eigenmode. The remaining eigenvalues can describe regional instabilities and may be useful for some analysis goals [68].

The ability to quickly and accurately determine the state of a multiplying nuclear system is very important in designing new nuclear systems. The standard eigenvalue solution methods can be slow for some cases of interest. To design new reactors, better solvers are needed. This chapter discusses the new eigenvalue solver that has been implemented to meet this challenge. Some background about eigenvalue solution methods is presented first. These methods are then put into the context of what has been done in the past and are used to explain how the new method works and why it is novel. Finally, results from the new solver are presented.

3.1 Background

Details about eigenvalue problems, characteristics of convergence, and eigenvalue solution methods are all needed to understand the new eigenvalue solver. The generalized eigenvalue problem takes the form $\mathbf{B}x = \mu\mathbf{C}x$ and can be transformed into an ordinary eigenvalue problem,

$\mathbf{A}x = \lambda x$. Both forms have the same right eigenvectors. If $\mathbf{C}$ is non-singular then $\mathbf{A} = \mathbf{C}^{-1}\mathbf{B}$ and then the problem $v = \mathbf{A}x$ can be solved in two steps: [61]

1. $w = \mathbf{B}x$

2. Solve the system $\mathbf{C}v = w$.

Because the generalized form can be converted to the ordinary form, this chapter will focus on the ordinary form without loss of applicability.

Some basic notation will be needed: let $\sigma(\mathbf{A}) \equiv \{\lambda \in \mathbb{C} : rank(\mathbf{A} - \lambda\mathbf{I}) \leq n\}$ be the spectrum of $\mathbf{A}$, where the elements in the set are the eigenvalues and $\mathbb{C}$ is the set of complex numbers. The eigenvalues can be characterized as the n roots of the polynomial $p_{\mathbf{A}}(\lambda) \equiv det(\lambda\mathbf{I} - \mathbf{A})$. Each distinct eigenvalue in $\sigma(\mathbf{A})$ has a corresponding nonzero vector x such that $\mathbf{A}x_i = \lambda_i x_i$ for $i = 1, ..., n$ [60]. It will be assumed that the eigenvalues are ordered as $|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| \geq 0$.

Eigenvalue problems in the nuclear transport community are typically solved with iterative rather than direct methods. A variety of iterative solvers have been used to solve eigenvalue problems. This subsection will cover the methods that have been most widely used in the nuclear field as well as methods that will aid in understanding the work being proposed.

3.1.1 Convergence Concerns

Before the methods are discussed, some terms that are used to characterize how numerical methods might behave when solving problems of interest are presented. A method's behavior can be affected by characteristics of the mathematical problem ($f : X \rightarrow Y$) as well as by characterisitics of the algorithm itself ($\tilde{f} : X \rightarrow Y$). The material in this subsection is derived from Trefethen and Bau [65]; see this reference for more detail.

Conditioning is one way to express the perturbation behavior of the mathematical problem. A *well-conditioned* problem is one in which all small perturbations of x lead to only small changes in $f(x)$. An *ill-conditioned* problem is one in which some small perturbation of x leads to a large change in $f(x)$. The condition number is a quantity used to express how well-conditioned a matrix or problem is. A small condition number corresponds to a well-conditioned problem, and vice versa.

The relative condition number is used to measure the effect of relative perturbations. This is particularly useful in numerical analysis because computers introduce relative errors. Let δx be a small perturbation of x and $\delta f = f(x+\delta x) - f(x)$. With these terms, the relative condition number is defined as

$$\kappa(x) = \lim_{\delta\to 0} \sup_{||\delta x||\leq\delta} \left(\frac{||\delta f||}{||f(x)||} / \frac{||\delta x||}{||x||} \right) , \tag{3.1}$$

where $\sup$ is the supremum, the smallest real number that is greater than or equal to every number in the set in question [71]. This definition holds for any norm. To express a specific norm, a sub number will be added, e.g. the two norm would be indicated by $\kappa_2(x)$.

The condition number of a matrix $\mathbf{A}$ is defined as

$$\kappa(\mathbf{A}) = ||\mathbf{A}||\, ||\mathbf{A}^{-1}|| \,. \tag{3.2}$$

If $\mathbf{A}$ is singular, its condition number is infinity. If the two-norm is used, then $||\mathbf{A}|| = \sigma_1$, $||\mathbf{A}^{-1}|| = \sigma_m$, and $\kappa_2(\mathbf{A}) = \frac{\sigma_1}{\sigma_m}$. Here σ_j is the jth singular value of $\mathbf{A}$, and the singular values are ordered such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n > 0$. The ratio $\frac{\sigma_1}{\sigma_m}$ can be interpreted as the eccentricity of the hyper-ellipse that is the image of an m-dimensional unit sphere under $\mathbf{A}$. When $\mathbf{A}$ has a large condition number the largest principle semiaxis is much longer than the smallest principle semiaxis.

Stability is a way to express the perturbation behavior of an algorithm used to solve a problem on a computer. A stable algorithm "gives nearly the right answer to nearly the right question." An algorithm $\tilde{f}$ for a problem f is stable if each $x \in X$ satisfies

$$\frac{||\tilde{f}(x) - f(\tilde{x})||}{||f(\tilde{x})||} = O(\epsilon_{machine}) \qquad \text{for some } \tilde{x} \text{ with } \frac{||\tilde{x} - x||}{||x||} = O(\epsilon_{machine}) \,. \tag{3.3}$$

Backwards stability has a tighter requirement than stability alone, and a backwards-stable algorithm will "give exactly the right answer to nearly the right question." An algorithm $\tilde{f}$ for a problem f is backwards stable if, for each $x \in X$,

$$\tilde{f}(x) = f(\tilde{x}) \qquad \text{for some } \tilde{x} \text{ with } \frac{||\tilde{x} - x||}{||x||} = O(\epsilon_{machine}) \,. \tag{3.4}$$

The concepts of condition and stability are not independent of one another. The accuracy of a backwards stable algorithm depends on the condition number, $\kappa(x)$ of f. The relative errors of an algorithm applied to some problem satisfy:

$$\frac{||\tilde{f}(x) - f(x)||}{||f(x)||} = O(\kappa(x)\epsilon_{machine}) \; . \tag{3.5}$$

These ideas and definitions will be needed when considering the behavior of the new eigenvalue solver in combination with the multigroup Krylov solver.

3.1.2 Power Iteration

Power iteration (PI) is an old and straightforward algorithm for finding an eigenvalue/vector pair. The basic idea is that any nonzero vector can be written as a linear combination of the eigenvectors of $\mathbf{A}$ because the eigenvectors are linearly independent, namely $v_0 = \gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_n x_n$ where x_j is the jth eigenvector and γ_j is some constant. This specific expression assumes a non-defective $\mathbf{A}$, though this assumption is not necessary for the method to work.

Another fact that is used to understand power iteration is that $\mathbf{A}^k x_i = \lambda_i^k x_i$. Thus

$$\mathbf{A}^k v_0 = \gamma_1 \lambda_1^k x_1 + \gamma_2 \lambda_2^k x_2 + \cdots + \gamma_n \lambda_n^k x_n \; . \tag{3.6}$$

Since $|\lambda_1| > |\lambda_i|, i \neq 1$, the first term in the expansion will dominate as $k \to \infty$ and $\mathbf{A}^k v_0$ therefore becomes an increasingly accurate approximation to x_1. In practice, it is desirable to avoid exponentiating a matrix and it is helpful to normalize v in order to avoid possible over or underflow. This leads to Algorithm 1 [61].

Using Equation (3.6) and Algorithm 1 it is easy to understand PI's convergence behavior. After k steps, the iteration vector will be:

$$v_k = \big(\frac{\lambda_1^k}{e_i^T \mathbf{A}^k v_0}\big)\big(\frac{1}{\lambda_1^k}\mathbf{A}^k v_0\big) \; . \tag{3.7}$$

If $\mathbf{A}$ has eigenpairs $\{(x_j, \lambda_j), 1 \leq j \leq n\}$ and v_0 has the expansion $v_0 = \sum_{j=1}^{n} x_j \gamma_j$ then

$$\frac{1}{\lambda_1^k}\mathbf{A}^k v_0 = \frac{1}{\lambda_1^k}\sum_{j=1}^{n} \mathbf{A}^k x_j \gamma_j = \sum_{j=1}^{n} x_j \big(\frac{\lambda_j}{\lambda_1}\big)\gamma_j \; . \tag{3.8}$$

Algorithm 1 Power Iteration

```
Given $\mathbf{A}$ and $v_0$, $v = \frac{v_0}{||v_0||_\infty}$.
Until convergence do:
    $w = \mathbf{A}v$
    $\lambda = \frac{w^T v}{v^T v}$
    $i = i_{max}(w)$
    $v = \frac{v}{e_i^T w}$.
```

Equation (3.8) shows that this algorithm converges at the linear rate of $|\frac{\lambda_2}{\lambda_1}|$, which is called the dominance ratio. If λ_2 is close to λ_1, then this ratio will be close to unity and the method will converge very slowly. If λ_2 is far from λ_1, then convergence will happen much more quickly. Put simply, PI is better suited for problems where $\mathbf{A}$ has eigenvalues that are well separated [60].

Power iteration is very attractive because it only requires matrix-vector products and two vectors of storage space. Because of its simplicity and low storage cost, PI has been widely used in the transport community for criticality problems for quite some time [43], [70]. Despite these beneficial characteristics many current codes use an acceleration method with PI, or have moved away from it altogether because of the slow convergence for many problems of interest. Nevertheless it is still used in some codes, has historical relevance, and is used in many studies as a base comparison case.

As an aside, it is interesting to point out the connection between Krylov methods and power iteration. The power method is a Krylov method that uses a subspace of dimension one. A Krylov subspace is built by storing the vectors created during each iteration of the power method. Krylov methods with subspaces larger than one take advantage of the information generated during each iteration that the power method discards.

3.1.3 Shifted Inverse Iteration

One way to improve power iteration is to shift the matrix $\mathbf{A}$ and then, in a power iteration-type scheme, apply the inverse of the shifted matrix rather than the regular unshifted matrix. The method is called shifted inverse iteration and the goal is to provide better convergence properties. Understanding why shifted inverse iteration is an improvement requires understanding spectral transformation.

The fundamental notion is that $\mathbf{A}$ can be shifted by a scalar without changing the eigenvectors. That is, for some shift μ, $(\mathbf{A} - \mu\mathbf{I})$ will have the same eigenvectors as $\mathbf{A}$ [60]. If $\mu \notin \sigma(\mathbf{A})$, then $(\mathbf{A} - \mu\mathbf{I})$ is invertible and $\sigma([\mathbf{A} - \mu\mathbf{I}]^{-1}) = \{1/(\lambda - \mu) : \lambda \in \sigma(\mathbf{A})\}$. Eigenvalues of $\mathbf{A}$ that are near the shift will be transformed to extremal eigenvalues that are well separated from the others. Such a spectral transformation can be added to the power method. Given an estimate, $\mu \approx \lambda_1$, the shifted inverse power method will usually converge more quickly than PI.

To see why this works consider:

$$\tau_1 = \frac{1}{\lambda_1 - \mu}\,, \tau_2 = \frac{1}{\lambda_2 - \mu}\,, \ \ldots\ , \tau_n = \frac{1}{\lambda_n - \mu}\,. \tag{3.9}$$

As $\mu \to \lambda_1$, $\tau_1 \to \infty$ and all the other eigenvalues go to finite quantities. If the τs are inserted into the convergence analysis done for PI, it can be shown that the inverse power method will converge as $\frac{|\lambda_1 - \mu|}{|\lambda_2 - \mu|}$. This is typically much faster than $|\frac{\lambda_2}{\lambda_1}|$, though the ultimate success of the method is dependent upon the quality of the shift [60], [65].

The algorithm for shifted inverse iteration is much like that for power iteration, requiring only a few simple changes. Step 1 now becomes "solve $(\mathbf{A} - \mu\mathbf{I})w = v$" and Step 3 becomes $\lambda = \mu + \frac{w^T v}{w^T w}$; all other steps are the same. After convergence, the actual eigenvalue of interest is backed out using the eigenvector [60]. Shifted inverse iteration is identical to performing power iteration on $(\mathbf{A} - \mu\mathbf{I})^{-1}$.

Wielandt's method is a flavor of shifted inverse iteration that has been used widely in the neutron transport community, though it was originally developed in 1944 in an unrelated context [73], [35], [34], [32]. In the transport equation formulation, Wielandt's method changes the eigenvalue problem from $\mathbf{A}\phi = \lambda\mathbf{F}\phi$ to $(\mathbf{A} - \lambda_e\mathbf{F})\phi = \delta\lambda\mathbf{F}\phi$ where λ_e is an estimate for λ_1 and $\delta\lambda = \lambda_1 - \lambda_e$. The power method with iteration index t is applied to this, giving $\phi^t = \delta\lambda^{t-1}(\mathbf{A} - \lambda_e\mathbf{F})^{-1}\mathbf{F}\phi^{t-1}$. The estimate, λ_e, can be improved on each iteration by starting with an initial guess of 0 and setting $\lambda_e^t = \delta\lambda^{t-1} + \lambda_e^{t-1} + \alpha$. Here α is an optional small positive constant that is designed to keep roundoff error from dominating when λ_e becomes very close to λ_1 [46].

On initial consideration it would seem shifted inverse methods might not work well when the shift is very good because the matrix becomes very ill-conditioned. If the shift is exact, i.e. when $\mu = \lambda_1$, the matrix is singular. It turns out this concern is unfounded. Peters and Wilkinson proved that ill-conditioning is not a problem for inverse iteration methods [53]. Trefethen and Bau also assert that this is the case as long as the $\mathbf{A}w = v$ portion is solved with a backwards stable algorithm [65].

Studies of reactor systems have found shifted inverse iteration to be faster than power iteration [6]. In the general computational community, shifted inverse iteration has largely taken the place of power iteration when looking for eigenvectors associated with eigenvalues that are relatively well known at the outset of the calculation since this information allows for the selection of a good shift [32].

3.1.4 Rayleigh Quotient Iteration

Rayleigh quotient iteration is a variation of shifted inverse iteration that has a changing shift like Wielandt's method sometimes does. The key difference is that this method uses a specific formula, the Rayleigh quotient (RQ), that is designed to find the optimal the shift at every iteration.

The Rayleigh quotient for the ordinary eigenvalue problem, originally proposed by the third Baron Rayleigh in the 1870s, is defined as [51]:

$$\rho(x) = \frac{x^T\mathbf{A}x}{x^T x} \, . \tag{3.10}$$

If x is an eigenvector of $\mathbf{A}$, then the RQ is the corresponding eigenvalue. If x is close to an eigenvector, then the RQ will approximate the eigenvalue [61]. The derivation of the RQ comes from asking the question “what α will minimize $||\mathbf{A}x - \alpha x||_2$?” Solving this using least squares will give $\alpha = \rho(x)$ [65].

The RQ has a similar form for the generalized eigenvalue problem, mentioned here because it applies to some of the past work discussed below and is the form that was implemented in the new solver. For the problem $\mathbf{A}x = \lambda \mathbf{B}x$, there is a right eigenpair (λ, x) for $x \neq 0$ and a left eigenpair $(\lambda, y) : y^T\mathbf{A} = \lambda y^T \mathbf{B}$ for $y \neq 0$, of the pencil $(\mathbf{A}, \mathbf{B})$. Let $\langle \alpha, \beta \rangle$ be a simple eigenvalue of pencil $(\mathbf{A}, \mathbf{B})$. If x and y are right and left eigenvectors corresponding to $\langle \alpha, \beta \rangle$, respectively, then $\langle \alpha, \beta \rangle = \langle y^T\mathbf{A}x, y^T\mathbf{B}x \rangle$. This means the ordinary form of the eigenvalue is:

$$\lambda = \frac{y^T\mathbf{A}x}{y^T\mathbf{B}x}\,, \tag{3.11}$$

which is the generalization of the RQ [61].

Recall from the previous subsection that choosing a shift close to the eigenvalue of interest controls the dominance ratio of shifted inverse iteration, and hence convergence. The RQI algorithm simply uses a strategically selected shift; see Algorithm 2. This process generates a sequence,

Algorithm 2 Rayleigh Quotient Iteration

```
Choose a starting unit vector, $v_0$, then for $k = 0, 1, 2, \ldots$
    form $\rho_k = \rho(v_k) = v_k^T \mathbf{A} v_k$,
    if $\mathbf{A} - \rho_k$ is singular, then solve $(\mathbf{A} - \rho_k)v_{k+1} = 0$ for $v_{k+1} \neq 0$ and halt. Otherwise
    solve $(\mathbf{A} - \rho_k)w_{k+1} = v_k$,
    normalize $v_{k+1} = \frac{w_{k+1}}{||w_{k+1}||}$.
```

$\{\rho_k, v_k\}$, called the Rayleigh sequence generated by v_0 on $\mathbf{A}$. To more deeply understand why this method is optimal and useful for the purposes of this work, more properties of the Rayleigh quotient must highlighted [51]:

- For $\alpha \neq 0$, $\rho(\alpha x, \mathbf{A}) = \rho(x, \mathbf{A})$, so using any multiple of x will produce the same sequence as x.

- The RQ has translational invariance, $\rho(x, \mathbf{A} - \alpha\mathbf{I}) = \rho(x, \mathbf{A}) - \alpha$, meaning the matrix $(\mathbf{A} - \alpha\mathbf{I})$ produces the same sequence as $\mathbf{A}$ [51]. This relationship can be used to relate eigenvalues and applied shifts. In fact, this is one of the ways to find the eigenvalue of interest for the standard inverse iteration method [60].

- When x is an eigenvector of $\mathbf{A}$, $\rho(x)$ is stationary at x.

- When $x \neq 0$ the RQ gives the minimal residual, with equivalence holding only when $\beta = \rho(x)$:
$$||(\mathbf{A} - \beta)x||^2 \leq ||\mathbf{A}x||^2 - ||\rho(x)x||^2 \, . \tag{3.12}$$

- A corollary of this is that x is orthogonal to $(\mathbf{A} - \rho(x))x$ [51].

RQI has very good convergence properties for normal matrices. The minimal residual property of the RQ causes good global convergence behavior. The sequence of residuals $\{||(\mathbf{A} - \rho_k)v_k|| = ||r_k||, k = 0, 1, ...\}$ is monotone decreasing for all starting v_0. When RQI is applied to normal matrices, the following has been proven as $k \to \infty$:

1. $\rho_k = \rho(v_k)$ converges, and either

2. (ρ_k, v_k) converges cubically to an eigenpair (λ, k), or

3. ρ_k converges linearly to a point equidistant from $s \geq 2$ eigenvalues of $\mathbf{A}$, and the sequence $\{v_k\}$ cannot converge.

This means that when RQI converges to the correct eigenpair it does so rapidly. However, there is a risk that if a bad starting vector is selected it will not converge at all.

The monotone sequence $\{||r_k||\}$ is bounded from below by 0. If the limit of the sequence is 0 as $k \to \infty$ then case 2 will be observed and convergence will be cubic. If the limit is greater than 0, case 3 is found. Unfortunately, it does not seem that this can be know a priori. In practice, however, users found that it was difficult to make the method fail for normal matrices [51].

For non-normal matrices the stationary property does not hold, which means the convergence is quadratic at best. The residual sequence is also not guaranteed to be monotone decreasing and thus no global convergence properties can be proven. It has been found in practice that RQI will still converge for non-normal systems, just at a slower rate than for normal matrices. However, convergence cannot be guaranteed nor predicted in advance [51].

3.1.5 Krylov Methods

The multigroup Krylov solver is used over all energy groups inside the new eigenvalue solver. Because this requires applying a Krylov method to a shifted system, some additional facts about Krylov methods are needed. Krylov subspaces have the property that for any μ, $\mathcal{K}_k(\mathbf{A}, x) = \mathcal{K}_k(\mathbf{A} - \mu\mathbf{I}, x)$. This means that a Krylov method can be applied to a shifted system and the correct eigenvector will still be found [61].

Convergence and stability properties differ from Krylov method to Krylov method. Because GMRES is the most widely used Krylov method in this work, convergence and stability discussions will focus on this method. Paige et. at. [50] demonstrate that the least squares solution at every step in GMRES is always backwards stable. GMRES is also backwards stable when finding x in $\mathbf{A}x = b$ for "sufficiently nonsingluar $\mathbf{A}$." An $n \times n$ $\mathbf{A}$ that qualifies as nonsingular enough satisfies

$$\mathbf{A}x = b \neq 0\,, \qquad \mathbf{A} \in \mathbb{R}^{n\times n}\,, \qquad b \in \mathbb{R}^n\,, \qquad \sigma_{min}(\mathbf{A}) \gg n^2\epsilon||\mathbf{A}||_F\,. \tag{3.13}$$

Here ϵ is the floating point arithmetic unit roundoff, $||\cdot||_F$ is the Frobenius norm, and $\sigma_{min}(\mathbf{A})$ is the minimum singular value of $\mathbf{A}$. Using the notation that $\tilde{\mathbf{V}}_m$ is the $n \times m$ orthonormal matrix computed at step m of the modified Gram-Schmidt procedure during the Arnoldi process and includes rounding error, GMRES is also backwards stable up until [50]

$$\kappa_2(\tilde{\mathbf{V}}_m) = \frac{\sigma_{max}(\tilde{\mathbf{V}}_m)}{\sigma_{min}(\tilde{\mathbf{V}}_m)} \leq \frac{4}{3}\,. \tag{3.14}$$

Trefethen and Bau [65] point out that evaluation of the convergence properties of GMRES is facilitated by relating it to a polynomial approximation problem. GMRES solves this approximation problem: find $p_n \in P_n$ such that

$$||p_n(\mathbf{A})b||_2 = \text{minimum}, \qquad \text{where}$$
$$P_n = \{\text{polynomials } p \text{ of degree } \leq n \text{ with } p(0) = 1\}.$$

This polynomial requires the first coefficient to be 1, which is a normalization at $z = 0$. GMRES chooses the coefficients of the polynomial p_n to minimize the norm of the residual, $r_n = b - \mathbf{A}x_n = (\mathbf{I} - \mathbf{A}q_n(\mathbf{A}))b$; q_n is a polynomial of degree $n-1$ and $p_n(z) = 1 - zq(z)$ [65].

GMRES is monotonically convergent: $||r_{n+1}||_2 \leq ||r_n||_2$. In addition, $||r_n||_2 = ||p_n(\mathbf{A})b||_2 \leq ||p_n(\mathbf{A})||_2 \; ||b||_2$. Thus, the convergence rate of GMRES is determined by

$$\frac{||r_n||_2}{||b||_2} \leq \inf_{p_n \in P_n} ||p_n(\mathbf{A})||_2 \; . \tag{3.15}$$

To get an idea of how quickly GMRES can converge, the question of how small $||p_n(\mathbf{A})||$ can be for a given n and $\mathbf{A}$ must be answered. For this analysis let $\mathbf{A}$ be diagonalizable such that $\mathbf{A} = \mathbf{V}\Lambda\mathbf{V}^{-1}$, where $\Lambda = \Lambda(\mathbf{A})$ is a diagonal matrix of eigenvalues for $\mathbf{A}$ and $\mathbf{V}$ is a non-singular matrix of eigenvectors. Also define the scalar $||p||_S = \sup_{z \in S} |p(z)|$. Then [65]

$$||p(\mathbf{A})||_2 \leq ||\mathbf{V}||_2 \; ||p(\Lambda)||_2 \; ||\mathbf{V}^{-1}||_2 = \kappa_2(\mathbf{V})||p||_{\Lambda(\mathbf{A})} \; . \tag{3.16}$$

All of this can be combined to say that at step n of GMRES, the residual satisfies

$$\frac{||r_n||_2}{||b||_2} \leq \kappa_2(\mathbf{V}) \inf_{p_n \in P_n} ||p||_{\Lambda(\mathbf{A})} \; . \tag{3.17}$$

This means that the convergence of GMRES depends on how far $\mathbf{A}$ is from normal and how quickly the size of $p_n(\mathbf{A})$ on the spectrum $\Lambda(\mathbf{A})$ decreases with n [65].

Nachtigal et. al. [45] derive a similar bound for any arbitrary $\mathbf{A}$, removing the diagonalizable assumption. In this case let $\Lambda_\epsilon \supseteq \Lambda$ be the ϵ-pseudospectrum of $\mathbf{A}$. The set of pseudo-eigenvalues are the points $z \in \mathbb{C}$ that are eigenvalues of some matrix $\mathbf{A} + \mathbf{E}$ with $||\mathbf{E}|| \leq \epsilon$ and $\epsilon > 0$. Let L be the arclength of the boundary $\partial\Lambda_\epsilon$. Then

$$\frac{||r_n||_2}{||b||_2} \leq \frac{L}{2\pi\epsilon} \inf_{p_n \in P_n} ||p||_{\Lambda_\epsilon(\mathbf{A})} \; . \tag{3.18}$$

For an arbitrary $\mathbf{A}$, the convergence of GMRES depends on the polynomial approximation defined on the pseudospectrum rather than the spectrum. The implications of the relationships in Equations (3.18) and (3.17) are, however, similar.

It will prove to be of crucial import that Krylov methods tend to converge very slowly for ill-conditioned systems [65], [50]. As a result of this property, many researchers have found that Krylov methods must be preconditioned to be able to get good results in practice [56], [36], [65].

3.2 Past Work

The ideas from the previous section will provide meaning and significance for past related work as well as for the new work. There are three main areas of past work that are pertinent to the new method. The first is an overview of how people have and do solve the eigenvalue neutron diffusion and transport problems. The next is Denovo's current eigenvalue solution method. The third is what happens when Krylov methods are applied to ill-conditioned systems.

3.2.1 Eigenvalue Solution Methods

A review of the literature leads to a variety of useful observations that will be discussed: shifted inverse iteration (SII) is generally better than power iteration; the key to succeeding with SII is selecting a good shift; there are a variety of eigenvalue solution methods used in diffusion and transport; and no one has used RQI to solve the transport equation.

It has been common practice to use the power method to solve both the diffusion and transport eigenvalue equations. However, it is well known that PI converges very slowly for systems with loosely coupled or optically thick fission regions and thus SII is often pursued as an alternative choice. For many problems SII converges more rapidly than PI [3], [20]. Allen and Berry compared the power method and the shifted inverse power method for the one-group, one-dimensional transport equation and found that the inverse method converges much more quickly [6].

Itagaki used Wielandt's spectral shift technique to solve the one-group, 3-D diffusion equation for various problem types, including systems containing strong neutron absorption. Itagaki's numerical results illustrate some problems with this method. When the estimate λ_e is not good, the correct eigenvalue may never be found [35], [34].

An analysis of the Ringhals reactor was done using two-group, finite difference, diffusion with Wielandt's method. When doing actual core analysis it was found that the local flux shape around the control rods made the flux converge slowly. The only way to improve convergence was to use a good initial guess for the flux [31].

Quite recently Zinzani et. al. modified the Wielandt method for the two-group diffusion equation to be able to find multiple eigenmodes. The method removes the requirement of a good initial guess, making it more robust than Wielandt's original method. Numerical results showed their method to be accurate and robust, but the algorithm is complicated and quite slow. They also used an explicitly restarted Arnoldi method which they found to be much faster than their modified power method [73].

These studies demonstrate, as is expected given the theoretical considerations discussed above, that shifted inverse iteration performs better than power iteration for many problems and the degree to which it is better often depends upon the quality of the shift and the initial guess for the eigenvector.

Eigenvalue solution methods that are not based on power iteration have started to become prevalent in the last 20 years. Many of these methods use Krylov subspace methods, and some use the Rayleigh quotient in various ways.

In 1988 Suetomi and Sekimoto used a Rayleigh Quotient Minimization (RQM) method to solve the one-group, 2-D diffusion equation in the form of a generalized k-eigenvalue problem. This method is a minimization problem for the Rayleigh quotient; the minimal RQ gives the best eigenvalue and corresponding eigenvector estimate. The monoenergetic diffusion equation is symmetric positive definite (SPD), so incomplete factorization preconditioned CG was used to find the minimum RQ. The preconditioned RQM method performed better than both unpreconditioned RQM and power iteration with SOR as the inner iteration solver [64].

In 1991 Suetomi and Sekimoto extended their work to the few-group, 2-D, k-eigenvalue diffusion problem. This system is no longer SPD and as a result they cannot use CG. Instead they chose a conjugate residual method called ORTHOMIN(1). This functions like CG, but has a slightly different formulation and cannot use the Rayleigh quotient. The ORHTOMIN(1) method requires that the symmetric portions of the matrices be positive definite. They again used incomplete factorizations for preconditioning and compared the method with PI, finding results similar to the one-group case [63].

In 2004 Gupta and Modak used Suetomi and Sekimoto's work as a basis for a method to solve the generalized eigenvalue form of the transport equation. The group noted ORTHOMIN(1) could be applied to the transport equation in the same way that Suetomi and Sekimoto applied it to the diffusion equation. This method would solve the entire problem at once, removing the inner-outer iteration structure. However, Gupta and Modak elected not to pursue this approach because it requires the explicit formation of both the transport and fission matrices. For 3-D, multigroup transport problems, forming and storing these matrices can be quite expensive. In addition, rewriting transport solvers to handle the implementation would be difficult [26].

To overcome these obstacles, Gupta and Modak changed the original method such that it retained the inner-outer structure and did not require explicit matrix formation. This approach made the problem smaller and only required the ability to solve a fixed-source problem. They used TSA accelerated SI in the inner iterations to find group fluxes, and ORTHOMIN(1) was applied as the outer iteration method. The new method was compared to the power method using SI and TSA accelerated SI. In two test problems they found ORHTOMIN(1) with TSA to be faster than PI [26].

Beginning in 1997 Vidal, Verdú, and others began a development stream in which they solved the multigroup, downscattering only, k-eigenvalue diffusion equation with both the Implicitly Restarted Arnoldi Method (IRAM) and another subspace method that is a generalization of the power method [67], [68], [66].

Vidal et. al. used Rayleigh-Ritz projection and symmetric Rayleigh-Ritz projection as subspace iteration methods to solve the downscattering only, two-group, 3-D diffusion equation using nodal collocation as the spatial discretization. The methods use block sweeps that are solved with an iterative method like CG with a Jacobi preconditioner. For each iteration they get an approximate set of eigenvectors $\mathbf{X}_n$, orthonormalize the set to get $\mathbf{X}_{n+1}$, and then compute the Rayleigh-Ritz projection: $\hat{\mathbf{Y}}^{n+1} = (\mathbf{X}^{n+1})^T\mathbf{Y}\mathbf{X}^{n+1}$. The eigenvalue problem is then solved with $\hat{\mathbf{Y}}^{n+1}$ to get the new estimate for the eigenvectors [68].

They also developed a variational acceleration technique that takes advantage of the fact that all dominant eigenmodes in a reactor are physically real. They found the symmetric method to be better than the general method and that the acceleration technique gave some improvement [68]. In related work, Vidal et. al. compared the Rayleigh-Ritz methods with an explicitly restarted Arnoldi method, all of which have been parallelized, for the two-group diffusion equation [67].

Verdú et. al. extended Vidal et. al.'s 1998 work [68] to find multiple eigenvalues for the two-group diffusion equation. They compared the power method, the symmetric Rayleigh-Ritz method with variational acceleration, and an implicitly restarted Arnoldi method. The restarted Arnoldi code they chose was IRAM. Verdú's group slightly modified the original method for the diffusion equation. It was found that IRAM was much faster than the Rayleigh-Ritz method and both were much better than power iteration [66]. Hernández et. al. applied a parallel implementation of IRAM to the two-group neutron diffusion problem [28].

These different approaches have largely been applied to the diffusion approximation, though a few have been extended to the transport equation. Several of the diffusion solution methods make use of the Rayleigh quotient, though none have done Rayleigh Quotient Iteration itself. The author was unable to find any examples where RQI was used an eigenvalue solution method for the neutron transport equation.

3.2.2 Denovo's Power Iteration

Before this work, the eigenvalue solution method used by Denovo was not too different from what has been done in other 3-D transport codes. Recall the operator form of the transport equation seen in Equation (1.12). This equation can be combined with $\phi = \mathbf{D}\psi$ and then rewritten and manipulated in the following ways:

$$\phi = \mathbf{D}\mathbf{L}^{-1}\mathbf{M}\mathbf{S}\phi + \mathbf{D}\mathbf{L}^{-1}\mathbf{M}\frac{1}{k}\chi\mathbf{f}^T\phi\,, \tag{3.19}$$

let $\mathbf{T} = \mathbf{D}\mathbf{L}^{-1}$, then multiply both sides by f^T

$$f^T(\mathbf{I} - \mathbf{T}\mathbf{M}\mathbf{S})\phi = \frac{f^T}{k}\mathbf{T}\mathbf{M}\chi f^T\phi\,, \tag{3.20}$$

$$f^T\phi^{k+1} = \frac{f^T}{k}(\mathbf{I} - \mathbf{T}\mathbf{M}\mathbf{S})^{-1}\mathbf{T}\mathbf{M}\chi f^T\phi^k\,. \tag{3.21}$$

Equation (3.21) is formulated using traditional power iteration. The outer iterations update the eigenvalue and there are two sets of inner iterations: energy, and space-angle. Originally, a Krylov solver was used for the space-angle inner iterations and Gauss Seidel was used for the energy iterations [19], [20]. Denovo uses Trilinos [29] to provide the Krylov solver, with a choice of either GMRES or BiCGSTAB [18].

3.2.3 Krylov Methods Concerns

As has been mentioned, Krylov methods have frequently been applied to the solution of the transport equation. It has often been found that these Krylov methods do not always converge well when they are not preconditioned. The first observation of unsatisfactory behavior was by Lewis in 1977 when CG was applied to the 1-D, symmetrised, S_N transport equation. The formulation used had a large condition number and that likely caused the problem [42], [25].

Recent work in 1- and 2-D transport analysis has highlighted that restarted GMRES can stagnate in problems with optically thin subdomains [55]. In 1998 Oliveira and Deng studied the application of a variety of Krylov methods to the 1-D neutron transport equation. They found that all of these methods can converge slowly for problems of interest if they are not preconditioned [49]. The recognition that Krylov methods can have convergence problems for transport problems led Patton and Holloway to study how different preconditioners improve the performance of GMRES when applied to the 1-D, S_N transport equation. They focused on GMRES because they found it to be faster and less memory intensive than other Krylov methods [52].

These experiences emphasize that Krylov methods can have serious convergence problems in practice. These methods particularly have trouble with poorly conditioned problems and cases with optically thin regions, which are cases of interest in real transport calculations.

3.3 RQI in Denovo

The theory and past work both indicate that Power Iteration can converge slowly for some types of eigenvalue transport problems. A better eigenvalue solution method is needed to be able to do cutting edge, high-fidelity calculations. The last two sections illustrated that a shifted inverse iteration method could provide substantial benefit, particularly if an optimal shift is chosen. These two observations inspired the choice of Rayleigh Quotient Iteration for a new eigenvalue solver in Denovo.

A key element that enabled this choice is the multigroup Krylov solver. To implement RQI in Denovo, the shift was applied to the scattering matrix which effectively adds “upscattering” entries to S everywhere. Without the new Krylov solver, the energy iterations would be extremely time consuming. An added benefit is that RQI can take advantage of energy parallelization. By combining the new multigroup Krylov method and RQI, eigenvalue calculations can converge faster for some problems and can be parallelized across all energy groups. The balance of this chapter will cover the details of the method implementation, present some results, and discuss the implications of RQI.

3.3.1 Method

Chapter 2 introduced the block Krylov method that allows groups with non-zero entries in the upper triangular portion of the scattering matrix to be decomposed in energy. For fixed source problems this applies to groups with upscattering. The new method allows the new energy decomposition to be used in eigenvalue problems over all groups instead of just upscattering groups.

To see how this will work, subtract $\rho\mathbf{TMF}$ from both sides of $(\mathbf{I}-\mathbf{TMS})\phi = \lambda\mathbf{TMF}\phi$, where ρ is the Rayleigh quotient and $\lambda = \frac{1}{k}$. This gives the following shifted system:

$$[\mathbf{I} - \mathbf{TM}(\mathbf{S} + \rho\mathbf{F})]\phi = (\lambda - \rho)\mathbf{TMF}\phi\,. \tag{3.22}$$

Now define $\tilde{\mathbf{S}} \equiv \mathbf{S} + \rho\mathbf{F}$ to write

$$[\mathbf{I} - \mathbf{TM}\tilde{\mathbf{S}}]\phi = (\lambda - \rho)\mathbf{TMF}\phi\,. \tag{3.23}$$

The new matrix, $\tilde{\mathrm{S}}$, is dense since the fission matrix is dense. Because the shifted scattering matrix is dense, it looks like one big upscattering block and can be treated in just that way. This means the entire problem can be decomposed in energy and the new block Krylov solver can be used over all groups.

The energy-parallelized RQI method is shown in Algorithm 3. Here $\mathbf{A} = [\mathbf{I} - \mathbf{TMS}]$ and a block of energy groups corresponding to an energy set is denoted by superscript $\tilde{g}$. The moments being computed by a given energy set are indicated by $\phi^{\tilde{g}}$. Denovo's implementation of RQI uses

Algorithm 3 Rayleigh Quotient Iteration in Denovo

```
Break the problem into energy sets and distribute to the appropriate number of processors
Get initial guesses for $\phi$ and set $k_{old} = k_0$
Calculate the initial Rayleigh quotient, $\rho_{old}$ using Algorithm 4
Until convergence:
    Calculate the shift: $\mu = \frac{1}{k_{old}} - \rho_{old}$
    Apply the shift to both sides by
        1. subtracting $\rho_{old}[\mathbf{TMF}]^{\tilde{g}}$ from $\mathbf{A}^{\tilde{g}}$ and
        2. making the right hand side $\mu[\mathbf{TMF}]^{\tilde{g}}\phi^{\tilde{g}}$
    Use MG Krylov solver to solve $[\mathbf{I} - \mathbf{TM\tilde{S}}]^{\tilde{g}}\phi^{\tilde{g}} = \mu[\mathbf{TMF}]^{\tilde{g}}\phi^{\tilde{g}}$ for an updated $\phi^{\tilde{g}}$
    Global barrier so all processes can finish calculating $\phi^{\tilde{g}}$; all processes get the new $\phi$
    $k_{old} = \frac{1}{\rho_{old}}$
    Calculate a new Rayleigh quotient, $\rho_{new}$; $k_{new} = \frac{1}{\rho_{new}}$
    Compare $k_{old}$ and $k_{new}$ for convergence
    $k_{old} = k_{new}$; $\rho_{old} = \rho_{new}$
```

the generalized form of the Rayleigh quotient: $\frac{\phi^T \mathbf{A}\phi}{\phi^T \mathbf{TMF}\phi}$. The way this is computed using energy sets is shown in Algorithm 4.

There is also an option to simply do shifted inverse iteration instead of RQI. In this case the user chooses a fixed shift, β, that is used in place of ρ when shifting the matrices. Algorithm 3 can be modified to accommodate this choice by changing ρ to β in the calculation of μ and the "Apply shift to both sides" step. The Rayleigh quotient is still calculated at every iteration to make the new estimate for k.

Algorithm 4 Calculating the Rayleigh Quotient

```
denominator = $\phi_{old}^{\tilde{g},T}\mathbf{TMF}^{\tilde{g}}\phi_{old}^{\tilde{g}}$
Globally sum the denominator
If |denominator| $> 1 \times 10^{-13}$,
    numerator = $\phi_{old}^{\tilde{g},T}\mathbf{A}^{\tilde{g}}\phi_{old}^{\tilde{g}}$
    Globally sum the numerator
    $\rho = \frac{\text{numerator}}{\text{denominator}}$
Else, $\rho = \frac{1}{k_{old}+1\times10^{-6}}$
```

Unfortunately, the current implementation of RQI does not work with multisets when there are reflecting boundaries. When the standard transport operator in Denovo is applied to a vector, the contribution of the reflecting terms to the moments are not converged. These reflecting contributions must be included in the calculation for the RQ, but the contributions are only converged when a full solve is done. A full solve is not needed to calculate the RQ. The application of standard operator, which is needed, does not correctly include the reflecting contributions.

A new solver that acts like the operator but converges the reflecting terms was added to remedy this issue. This solver is used to calculation the RQ when there are reflecting boundaries. This operator has not yet been parallelized in energy. As a result, RQI can only use multisets with reflecting boundaries. A detailed explanation of this situation is given in Appendix C.

3.3.2 Results

A few calculations were done to investigate the improvement gained from RQI. The quality of improvement is measured by the total number of Krylov iterations rather than timing. This is a more meaningful measure because some calculations were so small and ran so fast that timing does not represent much, this is a metric that can be compared across computers, and there may be room for some optimization in the future that would change the timing.

Before RQI was added to Denovo, an infinite medium scoping code was written in Python to test that the algorithm worked in that very simple case. Using this code a 4-group, a 7-group, and a 27-group problem were run. In all cases RQI and PI both converged on the first iteration to the correct answer.

The first Denovo problem chosen was a small, few-group system intended to demonstrate the new solver's general functionality. This problem used a 3 $\times$ 3 $\times$ 3 grid with 0.1 spacing, vacuum boundary conditions, two materials, four groups with only downscattering, S_2, P_0, had all tolerances set to 1 $\times$ 10^{-6}, a dominance ratio of 0.1396, and a reference k of 0.11752. The small dominance ratio implies that PI should preform reasonably well on this problem. The debug version of Denovo was used and the calculation and was not parallelized in any way.

This problem was solved with RQI and PI, both using the MG Krylov solver for the multigroup iterations. Note that when PI is used with MG Krylov all the downscatter groups are still solved using the forward substitution method one group at a time rather than as a block. The RQI method uses MG Krylov over the whole block because of the effective upscattering added by the shift. GMRES was the Krylov method used in all cases unless noted otherwise.

Table 3.1 Small, Two Material, Four Group, Vacuum Boundary Problem RQI Test Results

Item	Power Iteration	RQI
k	0.11752	0.11752
group iters	2 or 3 / group	4 to 8
eigen iters	7	6
total Krylov	73	39

The results are given in Table 3.1. The term “group iters” either refers to the number of within group iterations (in the case of PI) or multigroup iterations where the Krylov vector is over all groups (in the case of RQI). Both RQI and PI got the correct answer. PI used a total of 73 Krylov iterations while RQI used a total of 39 Krylov iterations. The subspace sizes are not the same so the work per Krylov iteration is a little bit different, but RQI still used far fewer Krylov iterations than PI.

The second problem was nearly the same as the first, but was intended to test the reflecting boundary case. For this problem all boundaries were reflecting and only one material was used. These changes made the dominance ratio 1.630×10^{-15} and k became 2. These results can be seen in Table 3.2. Again RQI needed fewer Krylov iterations than PI and both methods converged to the correct answer.

Table 3.2 Small, One Material, Four Group, Reflecting Boundary Problem RQI Test Results

Item	Power Iteration	RQI
k	2	2
group iters	9 / group	17 to 18
eigen iters	2	2
total Krylov	72	35

This problem was also solved using the fixed shift option. A shift of 1.95 was selected, and this gave the same behavior as RQI. The fixed shift option was not extensively tested in this work. Shifted inverse iteration is a method that has been used for transport before and is therefore not of particular interest. It is simply noted here that this option exists and initial testing indicates that it works.

These first two small tests plus the infinite medium experience show that RQI can get the right answer and that it can converge in fewer iterations than Power Iteration. While these small successes were encouraging, they were not sufficient to demonstrate that the method is successful in general.

To continue investigation, an intermediately-sized problem was run next. This system had a 9 $\times$ 9 $\times$ 9 spatial grid with 0.5 spacing, reflecting boundary conditions, one material, 27 groups, 13 upscattering groups, S_2, P_1, the overall tolerance was 1×10^{-3}, the upscattering tolerance was 1

$\times$ 10^{-6}, and the eigenvalue tolerance was 1 $\times$ 10^{-5}. The dominance ratio was 6.32 $\times$ 10^{-13} and the reference k was 0.4.

When PI with MG Krylov was used the correct answer was obtained in 2.57 $\times$ 10^{2} seconds. The calculation needed 90 Krylov iterations per eigenvalue iteration and two outer iterations for a total of 180 Krylov iterations to converge.

When RQI was used with MG Krylov, the problem did not converge. The maximum number of Krylov multigroup iterations per eigenvalue iteration was set to 1,000 and this was taken every time except the first, which took 25. That means the eigenvector was not converging. The calculation was terminated after 40 eigenvalue iterations. The value of k oscillated between 0.3966 and 0.3967. While the problem did not converge, at least the estimated eigenvalue was close to the right value.

This test was also tried using BiCGSTAB instead of GMRES for the Krylov solver since BiCGSTAB has different convergence properties. This change did not improve things. The eigenvector exhibited the same behavior, requiring all the multigroup iterations for each eigenvalue iteration. After seven eigenvalue iterations the problem terminated because a negative eigenvalue was calculated. Choosing to use BiCGSTAB made the results worse for this problem.

With an even more difficult problem RQI could not even nearly get the eigenvalue when using GMRES. The two dimensional C5G7 mox benchmark problem [47] was selected for investigation and was run with an optimized version of Denovo. This problem used three enrichment levels, an external mesh file, two reflecting boundaries to make it 2-D, 10 materials, 7 groups, 4 upscattering groups, S_2, P_0, an overall tolerance and upscatter tolerance of 1 $\times$ 10^{-3}, an eigenvalue tolerance of 1 $\times$ 10^{-5}, k_0 set to 1.14, and had a dominance ratio of 0.7709. The reference k is 1.18655 $\pm$ 0.008.

The results from this calculation are presented in Table 3.3. Here Power Iteration was used with both accelerated Gauss Seidel (TTG GS) and with MG Krylov. PI + TTG GS needed a total of 21,000 single-group-sized Krylov iterations and PI + block Krylov needed 3,000 all-group-sized Krylov iterations to converge. Though the subspace sizes are different, the signification reduction in Krylov iterations with the MG Krylov solver provides another confirmation that MG Krylov is preferable to GS in most cases.

Table 3.3 2-D MOX C5G7 Benchmark RQI Results

Item	PI + TTG GS	PI + MG Krylov	RQI
k	1.18538	1.18702	$\sim$0.95
mg iters	157 GS	$\sim$100	1,000
eigen iters	32	32	120*
total Krylov	21,365	3,129	119,006

*Terminated manually

The most important result, however, was that with RQI the eigenvector did not converge and the eigenvalue was wrong. Except for the first, every set of mutligroup iterations went to the maximum iteration count of 1,000. A 3-D version of the C5G7 mox benchmark problem [48] was also tried with RQI. This, too, did not converge. Using BiCGSTAB instead of GMRES was also tried for the 2-D problem without success.

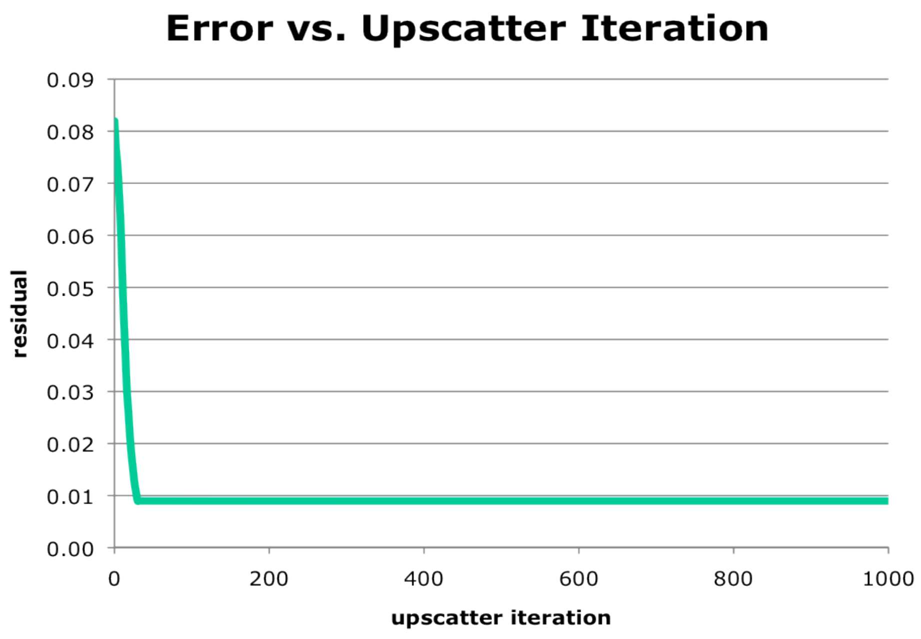


Figure 3.1 Residual as a Function of Upscatter Iteration Count for One Eigenvalue Iteration in the 2-D MOX Benchmark Problem

Figure 3.1 shows a plot of the residual vs. multigroup iteration count for one RQI eigenvalue iteration of the 2-D case. It is clear that error reduction stalls and the eigenvector does not converge. Without a good approximation to the eigenvector, the RQ is no longer a valid approximation to the eigenvalue and thus the eigenvalue problem does not converge.

It is likely that the eigenvector does not converge because RQI creates poorly conditioned systems. Krylov methods do not handle ill-conditioned systems very well and may converge extremely slowly. This hypothesis is supported by the fact that during the first eigenvalue iteration the Krylov method does converge fairly quickly. During this first iteration the Rayleigh quotient is computed from the initial guess and is likely not close to the actual eigenvalue. This means the shift on the first iteration is not close to λ_1 so the system will not be too ill-conditioned. After the eigenvector converges on that first iteration it may be that the RQ becomes much closer to λ_1, so the system becomes ill-conditioned and subsequent multigroup iterations do not converge.

An additional concern is that when the system is ill-conditioned GMRES may no longer be backwards stable. If GMRES is not backward stable, then there is no guarantee that RQI will converge. It is worth noting that there is no evidence that, given a sufficient amount of computing time, RQI would converge to the wrong answer. There is certainly not data to support the claim that RQI will converge to the proper eigenvalue in all cases, but it is heartening to notice there is also not data indicating that it will not.

Overall, though, the observed behavior strongly suggests that the multigroup Krylov method should be preconditioned so that the eigenvector will converge. If that happens it is likely that RQI will then converge as well. One with preconditioning can the real benefit of RQI be properly investigated.

3.3.3 Implications

Rayleigh quotient iteration is an old method that is an adaptation of shifted inverse iteration. Because the RQ provides an optimal shift, it is expected that using RQI in Denovo will converge in fewer iterations than Power Iteration for some problems, provided that the eigenvector is converged.

Further, this solver is wrapped around the MG Krylov solver from Chapter 2. It has already been demonstrated that the energy decomposition works and improves solution time and code scaling for fixed source multigroup problems. It is therefore reasonable to expect that when the method converges, decomposing the entire matrix in energy for eigenvalue calculations will work and provide the same kind of scaling improvement.

The intermediate and large problems that were tested demonstrate that RQI does not converge for even slightly challenging problems. This is probably caused by the poor condition number created by RQI. When the multigroup Krylov solver is not preconditioned it cannot converge the eigenvector in many cases. The incorrect eigenvector subsequently creates an incorrect eigenvalue.

The small problems showed that RQI can converge in fewer iteration than Power Iteration and may be quite beneficial if the multigroup iterations can be converged. If the MG Krylov solver is preconditioned, then it may be able to converge the eigenvector for cases of interest. To really investigate the benefit of RQI, a good preconditioner is needed. The work in the next chapter is tied to this issue.

RQI has not been applied to the transport equation before because it takes $O(n^3)$ operations for full dense matrices, and without parallelization in energy it could be prohibitively expensive [61]. In the past there was also no motivation to add a shift to make the scattering matrix block dense. If RQI had been used, the system would have been solved with Gauss Seidel and that would have been restrictively slow. The multigroup Krylov algorithm has enabled energy parallelization and made the calculation of the eigenvector tractable.

Computers that facilitate enough parallelization to decompose in energy and have enough memory to store Krylov subspaces for the full transport equation make combining RQI and MG Krylov possible. The resulting convergence of the eigenvalue should be faster than PI for at least some problems. The energy decomposition will allow eigenvalue calculations to overcome the limitations of KBA and be scaled to many more cores. These ideas have never been used together in this way. New kinds of problems can be solved to a new level of fidelity using these methods.

+

Chapter 4

Preconditioning

The new "grand challenge" problems facing the nuclear transport community are large and complex. Cutting edge methods are required to solve them. While the second and third chapters discussed new methods that enable the solution of such problems, low-cost preconditioners that can reduce the number of iterations needed for convergence will be invaluable. In Benzi et. al.'s 2002 survey paper on preconditioning techniques for large linear systems they state "it is widely recognized that preconditioning is the most critical ingredient in the development of efficient solvers for challenging problems in scientific computation [9]."

This is true for Krylov methods in particular because the memory required and cost per iteration increase dramatically with the number of iterations [9]. And, as discussed in Chapter 3, Krylov methods can converge very slowly for poorly conditioned systems. When this happens the eigenvector is not converged in a reasonable number of iterations and RQI cannot converge the eigenvalue. Preconditioning is therefore required to make RQI useful.

This chapter is about the new preconditioner added to Denovo. First some background information that includes an introduction to preconditioning, an overview of preconditioners used in the nuclear community, and a discussion of multigrid methods is given. Next, past and related work is discussed. Finally the new preconditioner is explained next, and results demonstrating its impact are given.

4.1 Background

The general idea of preconditioning is to transform the system of interest into another equivalent system that has more favorable properties, for example one with a smaller condition number. A preconditioner is a matrix that induces such a transformation by improving the spectral properties of the problem being solved. Let $\mathbf{G}$ be a non-singular preconditioner, then $\mathbf{A}x = b$ can be transformed in the following ways [9]:

$$\mathbf{G}^{-1}\mathbf{A}x = \mathbf{G}^{-1}b \qquad \text{left preconditioning,} \tag{4.1}$$

$$\mathbf{A}\mathbf{G}^{-1}y = b, \qquad x = \mathbf{G}^{-1}y \qquad \text{right preconditioning, and} \tag{4.2}$$

$$\mathbf{G}_1^{-1}\mathbf{A}\mathbf{G}_2^{-1}y = \mathbf{G}_1^{-1}b, \qquad \mathbf{G} = \mathbf{G}_1\mathbf{G}_2, \qquad x = \mathbf{G}_2^{-1}y \qquad \text{split preconditioning.} \tag{4.3}$$

If $\mathbf{A}$ and/or $\mathbf{G}$ are non-normal then each of the preconditioning constructs will likely give different behavior, though they will converge to the same answer because the matrices are similar and therefore have the same eigenvalues [9]. Right preconditioning leaves the right hand side of the equation unaffected and does not change the norm of the residual, which is used for convergence testing in most iterative methods. Right preconditioning is usually preferred over left preconditioning for iterative solvers for this reason [36]. A right preconditioner was implemented in this work, and the remaining discussion will be presented in right preconditioner format.

There are two extremes between which all other preconditioners lie: $\mathbf{G} = \mathbf{A}$, in which case the solution can be found directly, and $\mathbf{G} = \mathbf{I}$, which will have no effect. In general, the matrix $\mathbf{A}\mathbf{G}^{-1}$ is never formed. The preconditioner can be applied by using some method to solve $\mathbf{G}y = c \rightarrow y \approx \mathbf{G}^{-1}c$, or by otherwise implementing the action of $\mathbf{G}^{-1}$ without ever explicitly forming and inverting $\mathbf{G}$ [9], [65].

Functionally, a good preconditioner should make the system easier to solve and result in faster convergence. It should also be cheap to construct and apply. These tend to be competing goals in that the easier a preconditioner is to construct and apply the less it typically does to improve convergence. Generally, a preconditioner is good if $\mathbf{A}\mathbf{G}^{-1}$ is not too far from normal and its eigenvalues are clustered [65].

There are many different types of preconditioners, but they can be put into two general categories: matrix-based and physics-based. Matrix-based preconditioners rely entirely on the structure of the matrix $\mathbf{A}$, regardless of the physics going on in the problem. That is, these methods do not change when the underlying problem changes. This can be a very useful property because matrix-based methods are then broadly applicable and do not require any understanding of the physical problem. Extrapolation methods and incomplete factorization are examples of a matrix-based preconditioners [65].

Physics-based preconditioning uses knowledge about the physics of the problem in question to guide the creation of the preconditioner. This means that some methods only work with certain kinds of problems, and that the preconditioners may have to be tailored or adapted for different applications. However, such methods take advantage of knowing something about the problem and can be more effective than matrix-based methods for the range of problems for which they are intended. Rebalance and synthetic acceleration are examples of physics-based preconditioners [65].

4.1.1 Preconditioners in the Nuclear Community

A variety of acceleration methods have been used by the nuclear computational community over the years. This subsection gives a high-level overview of some common methods. In 2002 Adams and Larsen put together a comprehensive overview of the development of iterative methods for solving the S_N transport equation [3]. For more detail and history about each method, refer to this publication.

4.1.1.1 Extrapolation Methods

Extrapolation methods were historically used to accelerate k-eigenvalue calculations. These tend to be based on simple iterative solvers or polynomial approximations. Using one step of a simple iterative method such as Jacobi, Gauss Seidel, or successive over relaxation (SOR) can be used at the outset of a problem to serve as a preconditioner [65]. When applied to neutron transport, an overrelaxation method looks like $\mathbf{F}_{i+1} = \omega(\tilde{\mathbf{F}}_i - \mathbf{F}_i) + \mathbf{F}_i$ with $1 \leq \omega < 2$, where the unaccelerated new iterate for the fission source is designated $\tilde{\mathbf{F}}_i$ [43].

Polynomial preconditioners create a matrix polynomial $\mathbf{G}^{-1} = p(\mathbf{A})$, where $p(\mathbf{A})$ serves as a polynomial approximation to $\mathbf{A}^{-1}$. Common polynomial choices are truncated Neumann series and Chebyshev polynomials. An advanced variation of this method is to determine the polynomial coefficients adaptively [65]. When Chebyshev polynomials are applied to the transport equation, $\mathbf{F}_{i+1} = \mathbf{F}_i + \alpha_{i+1}(\tilde{\mathbf{F}}_i - \mathbf{F}_i) + \beta_i(\mathbf{F}_i - \mathbf{F}_{i-1})$, where α and β are iteration dependent [43]. Neither of these extrapolation methods are widely used today as they have not been very successful in multi-dimensional problems. There are analogous versions of these methods for fixed source problems as well [4].

4.1.1.2 Rebalance

Rebalance methods are designed to accelerate iteration on the scattering or fission source by imposing a balance condition on the unconverged solution over either coarse or fine regions. If the region is coarse, this is a coarse-grid approximation preconditioner since fine-grid physics are excluded. If the region is fine, this is a local approximation preconditioner where short range effects are permitted and long range interactions are excluded [65], [3].

In either case, the unaccelerated solution is multiplied by a constant in each region that satisfies the balance equation. The rebalance process adjusts the average amplitude of the flux over a region and the iteration adjusts the space-angle distribution. The notion is that these processes work in concert to eliminate all error modes simultaneously [3].

Coarse mesh rebalance (CMR) has been more successful than the fine version and is therefore used more often. In practice, choosing regions properly must be done with care [43]. Traditional forms of CMR can be unstable for problems where the spatial mesh is large compared to the neutron mean-free-path and where the scattering ratio is close to unity [4].

4.1.1.3 Incomplete Factorizations

Incomplete factorization methods were first introduced in the 1950s by Buleev, and independently by Varga. In the late 1970s these methods started gaining popularity, and since then have been under active development. The idea is to partially factor the matrix $\mathbf{A}$ such that the resulting matrices are sparse enough to take advantage of sparse solver methods, but close enough to $\mathbf{A}$ to be valuable as preconditioners [9].

Incomplete Cholesky (IC) or incomplete LU (ILU) factorization have both been extensively used within the nuclear community. The Cholesky version is used when $\mathbf{A}$ is symmetric and LU when non-symmetric. Factorizations typically destroy sparsity, resulting in matrices that are denser than the originals and making their use more costly. Different variations of incomplete factorization methods address this by permitting the new matrices to have values only in positions where $\mathbf{A}$ has values, allowing a prescribed amount of fill in, or only saving entries greater than some tolerance [65], [52], [49].

4.1.1.4 Synthetic Acceleration

In synthetic acceleration a low-order approximation to the transport operator is used to accelerate the full transport problem [43]. If a system is discretized with a high-order method it can be preconditioned with a lower-order approximation. The low-order method is often much sparser than the original, but still captures the general behavior of the problem [65]. This is generally done with either the diffusion equation, giving diffusion synthetic acceleration (DSA), or a transport equation that is simpler than the one actually being solved, giving transport synthetic acceleration (TSA). These can be used for fission or fixed source problems [3].

Original DSA formulations exhibited instability in some instances. In 1977 Alcouffe demonstrated with diamond difference that stability can be obtained by using a consistent spatial differencing scheme [4]. The transport and diffusion equations cannot be discretized independently or the diffusion equation may not satisfy the original transport equation. The discretization of the diffusion equation must therefore be derived from the discretization of the transport equation such that they are consistent. Alcouffe's idea has since been extended to accelerate currents, to more dimensions, and to different spatial discretizations [39].

DSA can be implemented a few different ways, each of which performs a transport sweep and then solves a corrected diffusion equation. Different terms in the diffusion equation can be corrected for different problem types [4]. DSA has been found to degrade in the presence of material discontinuities in addition to when it is not consistently derived. However, Warsa et. al. found that when DSA is used as a preconditioner for Krylov iterative methods, it is effective even when only partially consistent [69].

To address the concerns about spatial discretization associated with DSA, a low-order transport solve could be used to find the correction term instead. Larsen and Miller developed a method that neglects scattering in the low-order transport equation, but this can cause instability in multi-dimensional problems. Ramone et. al. proposed including some scattering by using a tunable parameter. A high-order transport equation is solved first, then a transport equation with a coarse quadrature and reduced scattering is solved to find the correction that is applied to the new iterate of the scalar flux [54]. This algorithm can be viewed as Richardson iteration with a coarse operator preconditioner. Synthetic acceleration methods are widely used and actively researched in the transport community today.

4.1.2 Multigrid Methods

The new preconditioner added to Denovo does multigrid in the energy dimension. To understand why multigrid in energy makes sense, why multigrid methods work must be understood. The material presented here is largely from *A Multigrid Tutorial* by Briggs, Henson, and McCormick [11], supplemented by Dr. Strang's lectures on multigrid and preconditioners available as MIT Open Courseware [62].

In what follows, the system of equations will be denoted as $\mathbf{A}u = b$ where u will represent the exact solution and v will indicate the approximate solution. A certain grid will be denoted as Ω^h where h is the grid spacing, and vectors on that grid are u^h. The ith entry in a vector is u_i. The algebraic error is given by $e = u - v$. Since e cannot be calculated exactly (otherwise the answer would be known), the residual, $r = b - \mathbf{A}v$ is often used.

The error in any guess can be written as a combination of Fourier modes. A Fourier mode is described by a wavenumber, k, which denotes the frequency of oscillation. The jth Fourier mode is

$$e_j = \sin\Big(\frac{jk\pi}{n}\Big)\,, \qquad 0 \le j \le n\,, \qquad 1 \le k \le n-1\,. \tag{4.4}$$

There are k half sine waves that comprise e on the domain of the problem. The term e_k indicates an e with wavenumber k. The kth wave has $\frac{k}{2}$ full sine waves with a wavelength of $\frac{2}{k}$. Modes with wavenumbers in the range $1 \le k \le \frac{n}{2}$ are called low-frequency or smooth modes and those in $\frac{n}{2} \le k \le n-1$ are called high-frequency or oscillatory modes. Figure 4.1 shows what a few different modes look like.

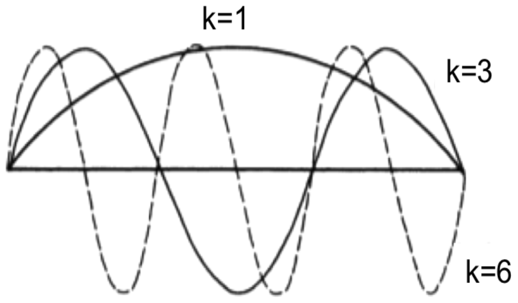


Figure 4.1 Three Fourier Modes [11]

Iterative methods, also referred to as smoothers or relaxers, remove high frequency error components very quickly, but take a long time to remove the low frequency components. This is illustrated in Figure 4.2, where weighted Jacobi was applied to Fourier modes of different frequencies on an n = 64 grid. The error is reduced much faster for higher modes. The rapid removal of oscillatory and slow removal of coarse error modes is often seen in practice. Figure 4.3 shows an error plot that exhibits this behavior

The idea of multigrid methods is to take advantage of the smoothing effect by making smooth error look oscillatory so it can be more easily removed. Error that is low frequency on a fine

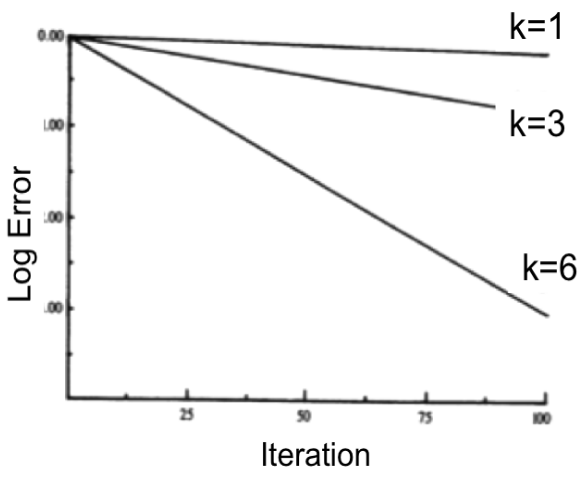


Figure 4.2 Log of Error As a Function of Iteration Count When a Relaxer is Applied to Three Fourier Modes [11]

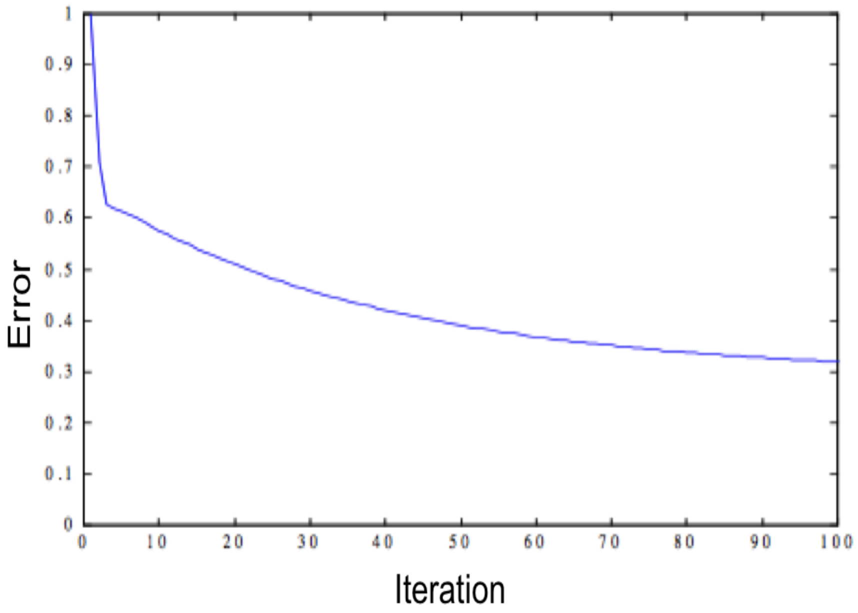


Figure 4.3 Error As a Function of Iteration Count. Oscillatory Components Are Removed Rapidly, Leaving Smooth Components [11]

grid can be mapped onto a coarser grid where it is oscillatory. This change can be thought of as increasing the number of oscillations per grid point. Look at Figure 4.4 for an example. When n =

12, the maximum number of half-sine waves is 12, so k = 4 is relatively smooth. When mapped to n = 6, k = 4 is relatively oscillatory.

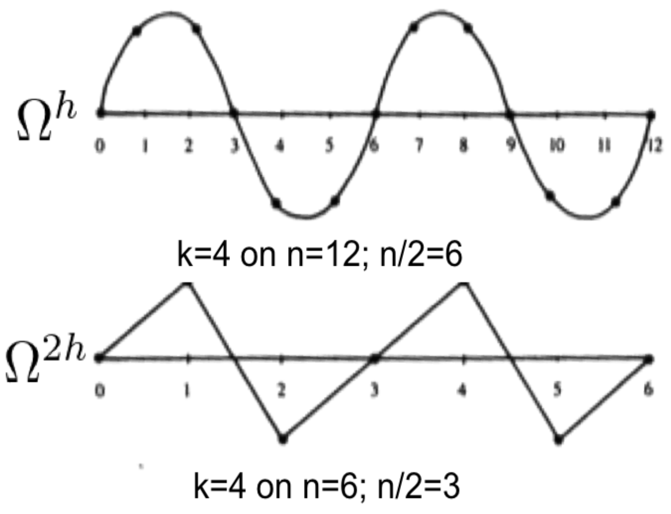


Figure 4.4 Relative Oscillation of a Fourier Mode Mapped Between Two Grids [11]

Using this information, multigrid methods can be understood. The error is mapped from a fine grid to a coarser grid. A smoother is applied on the coarse grid to remove the newly oscillatory error components. The result is mapped back to the fine grid and used to correct the solution there. A few more relaxations are done back on the fine grid. That whole process is called a v-cycle, which is described in Algorithm 5. A call to this method is symbolized as $v^h \leftarrow \mathbf{G}(v^h, b^h)$ and is effectively doing $v^h \approx \mathbf{G}^{-1}b^h$.

The details of how restriction and prolongation are done are problem dependent. Simple iterative schemes can have very simple operators while more complex schemes may require more complex mappings. Other implementation choices are what relaxer to use and how many relaxations to do on each grid.

There are many variations of how multigrid methods are put together. All methods, however, are built on the basic v-cycle correction scheme. A V-cycle is an extension of the v-cycle. Instead of only using two grids, many grids are used. The problem is restricted from grid to grid until it is on some grid which is coarse enough to directly invert the equations. Then the errors are prolongated

Algorithm 5 Multigrid v-cycle: $v^h \leftarrow \mathbf{G}(v^h, b^h)$

```
Relax $\nu_1$ times on $\mathbf{A}^h u^h = b^h$ on the fine grid $\Omega^h$ using initial guess $v^h$.
Compute the residual, $r^h = b^h - \mathbf{A}v^h$.
Using some restriction operator, $\mathbf{R}_h^{2h}$, restrict the residual to a coarser grid: $r^{2h} = \mathbf{R}_h^{2h} r^h$.
Solve the residual equation, $\mathbf{A}^{2h} e^{2h} = r^{2h}$, on coarse grid $\Omega^{2h}$.
Using some prolongation operator, $\mathbf{P}_{2h}^h$, prolong (interpolate) the coarse grid error back to a finer grid: $e^h = \mathbf{P}_{2h}^h e^{2h}$.
Add the error to the fine grid guess: $v^h \rightarrow v^h + e^h$.
Relax $\nu_2$ times on $\mathbf{A}^h u^h = b^h$ on $\Omega^h$ to get an improved solution.
```

back up the chain, continuously correcting on finer grids, until the finest grid is reached. Multigrid methods are differentiated by how many times the v-cycle is done and how many grids are used. A five-grid V-cycle can be seen in Figure 4.5 (a). A W-cycle is when V-cycles are repeated to further improve the answer. An example can be seen in Figure 4.5 (b).

Multigrid can also be used to obtain an initial guess where much of the smooth error has been removed. This is called nested iteration, and the iterations begin on the coarsest grid and move up to the finest gird. If nested iteration is combined with V-cycles, the full multigrid method (FMG) is obtained, as seen in Figure 4.5 (c). A call to any of the combinations of v-cycles is denoted $v \leftarrow \mathbf{G}(v, b)$. The optimal combination of grids and cycles may depend on problem type. The addition of more grids and cycles will reduce error, but at an added cost.

Multigrid methods can be thought of as stationary iterative schemes that can be used alone or as accelerators for other methods. In the past these methods were highly problem specific and only applied to second-order elliptic PDEs. Over time they have been extended to different problem types, geometries, and discretizations [9].

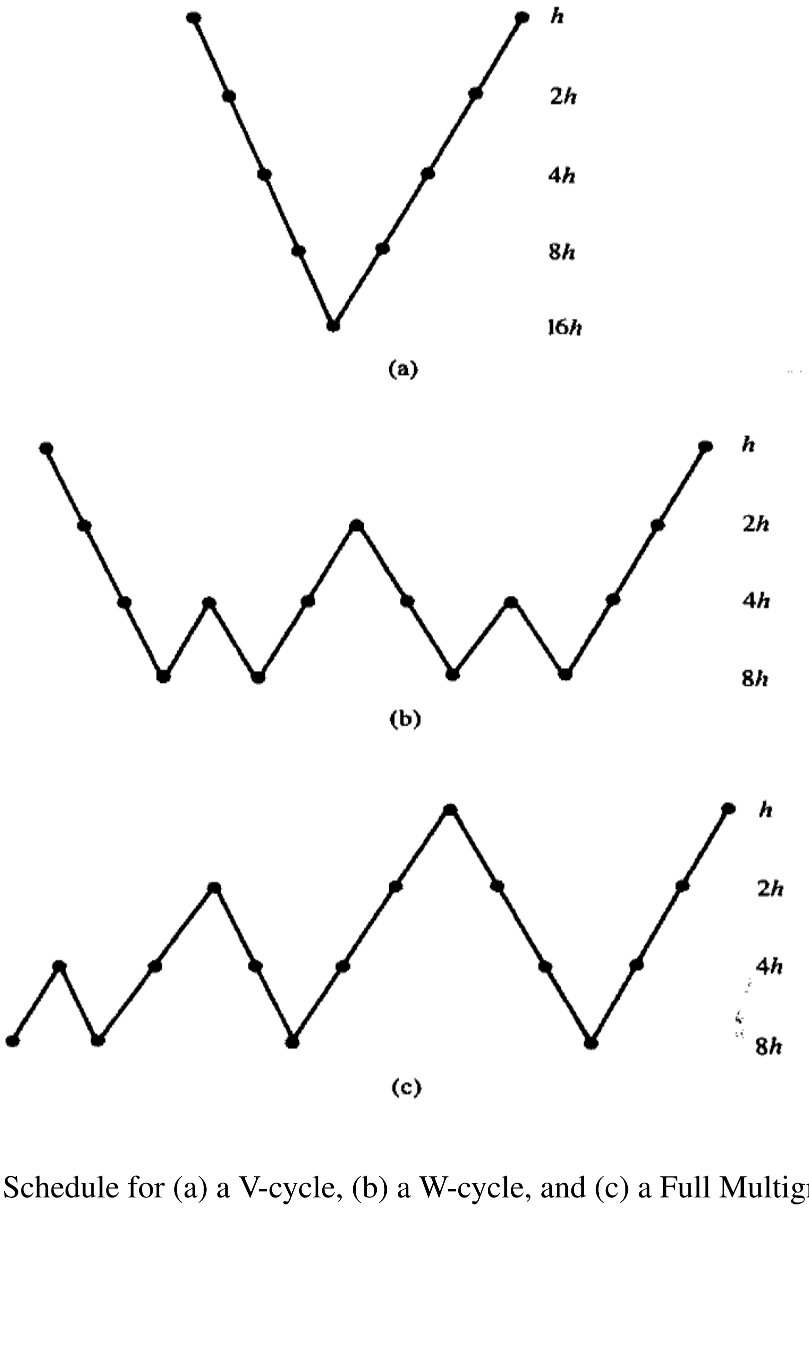


Figure 4.5 Grid Schedule for (a) a V-cycle, (b) a W-cycle, and (c) a Full Multigrid Pattern [11]

4.2 Past Work

This section discusses some past work from most of the categories of preconditioners that were presented above. The focus is on methods that have been used to precondition Krylov solvers and on multigrid methods. This section is intended to illustrate the need for preconditioning Krylov methods in transport and to demonstrate the new preconditioner's originality.

When using Krylov methods, how well a preconditioner is going to do for a certain problem cannot be known a priori. At this time there are no set methods or procedures for predicting preconditioner behavior with Krylov methods. As a result preconditioning Krylov methods is somewhat "guess and check." Despite this, the reward of preconditioning Krylov methods ensure that they are an essential part of their practical use.

4.2.1 Rebalance

CMR has been widely used in transport codes since at least the 1970s and new variants of it are being applied to Krylov solvers [15], [72]. For example, Dahmani et. al. investigated using GMRES and preconditioned GMRES for solving the 3-D transport equation using the method of characteristics (MOC) in 2005. Self-collision rebalance (SCR) was used as a left preconditioner. SCR uses the probability of a region scattering particles to itself to rebalance the energy distribution in each region. MOC is quite different from S_N and requires a specific derivation to be able to use GMRES [15].

4.2.2 Incomplete Factorizations

Incomplete factorizations have also been used to precondition Krylov methods, though not as successfully for large transport problems. Patton and Holloway investigated the use of a variety of preconditioners for GMRES to solve the multi-group, diamond-difference, 1-D, S_N equations. They compare matrix-based and physics-based right preconditioners. A variety of factorization methods, the matix-based preconditioners, were considered: ILU(0), modified ILU(0) (MILU),

ILU($\tau = 10^{-4}$), ILU($p = 10$), and ILU($\tau = 10^{-4}, p = 10$) where τ is the drop tolerance and p indicates the amount of fill-in allowed. A single integer indicates the fill-in limit.

The bandwidth of $\mathbf{A}$ increases as the number of energy groups and/or discrete ordinates increases [52]. The bandwidth of a matrix $\mathbf{A} = (a_{ij})$ is the maximum value of $|i - j|$ such that a_{ij} is nonzero [**?**]. The computational time required by ILU factorization increases with the bandwidth of $\mathbf{A}$. This makes factorization methods unattractive as accelerators for finely discretized problems. This prompted Patton and Holloway to consider physics-based methods.

A matrix splitting was chosen such that the original problem formulation is written as:

$$\mathbf{L}\psi - \mathbf{S}_{in}\psi = \mathbf{Q} \, , \tag{4.5}$$

where $\mathbf{L}$ is the streaming plus removal term, $\mathbf{S}_{in}$ is the inscattering source, and $\mathbf{Q}$ contains the external source. Patton and Holloway use $(\mathbf{L} - \mathbf{S}_{down})^{-1}$ as the preconditioner, where $\mathbf{S}_{down}$ is the downscatter part of the $\mathbf{S}$ matrix. They use source iteration as the within-group solver such that each new guess is obtained by applying $(\mathbf{L} - \mathbf{S}_{down})^{-1}(\mathbf{S}_{in} - \mathbf{S}_{down})$ to the old guess. For the 1-D case they found that this physics-based preconditioner is faster than the matrix-based factorization preconditioner, but that DSA is faster than both [52]. This is not surprising for the 1-D case with diamond difference.

In 2004 Chen and Sheu compared preconditioned conjugate gradient methods with SOR for 3-D, multigroup neutron transport. They used ILU and MILU to precondition both conjugate gradient squared (CGS) and BiCGSTAB. They chose these iterative techniques because they have good residual error control procedures that give good convergence rates in general [14].

Kozlowski, Downar, and Lewis investigated a Krylov preconditioning method for the multigroup SP_3 transport equations. This method involves reordering the fluxes to facilitate the use of block ILU as the preconditioner. This idea worked well for small problems, but may require too much storage for large problems [38].

4.2.3 Synthetic Acceleration

DSA has been under continuous development since Alcouffe's 1977 paper. For example, a recent development string began when Wareing, Larsen, and Adams developed a simple DSA scheme for bilinear discontinuous (BLD) discretizations schemes in 2-D. An unconditionally efficient multigrid technique for solving these 2-D, BLD equations was derived by Morel, Dendy, and Wareing. This was later extended to bilinear nodal differencing and bilinear characteristic differencing [3]. New versions of DSA are applied to Krylov solvers in current transport codes.

Very recently Rosa et. al. performed detailed Fourier Analysis on TSA combined with Inexact Parallel Block-Jacobi (IPBJ) splitting applied to the one- and two-dimensional transport cases. They noted that both experience in the nuclear community and analytical work have shown that solution methods, such as GMRES(m) can stagnate for problems containing optically thin spatial regions. Rosa et. al.'s analysis and results show that using modified TSA improves the spectral properties such that convergence can be obtained using a relatively small m when using GMRES(m) [55].

4.2.3.1 DSA in Denovo

Denovo has the option to use DSA to precondition the within-group transport equation, $\big(\mathbf{I} - \mathbf{D}\mathbf{L}^{-1}\mathbf{M}\mathbf{S}_{gg}\big)\phi_g = \mathbf{D}\mathbf{L}^{-1}Q_g$, which works very well when using a Krylov method. High-frequency error modes are what often cause instability in DSA. Krylov iteration, like iterative methods in general, will rapidly damp such oscillatory modes. Eliminating the high-frequency error enables the removal of the consistency requirement as DSA is not likely to fail [21]. This means DSA can be applied successfully for a variety of spatial discretizations.

The DSA-preconditioned one-group equation is:

$$\big(\mathbf{I} + \mathbf{P}\mathbf{C}^{-1}\mathbf{R}\mathbf{S}\big)\big(\mathbf{I} - \mathbf{D}\mathbf{L}^{-1}\mathbf{M}\mathbf{S}\big)\phi = \big(\mathbf{I} + \mathbf{P}\mathbf{C}^{-1}\mathbf{R}\mathbf{S}\big)\mathbf{D}\mathbf{L}^{-1}\bar{Q}\,. \tag{4.6}$$

Here $\mathbf{C}$ is the diffusion operator defined in Appendix A; $\mathbf{R}$ is the restriction operator, which maps the transport solution onto the diffusion vector; and $\mathbf{P}$ is the projection operator, which maps the diffusion vector onto the transport solution. Denovo does not actually form these operators, instead it solves the diffusion equation and updates the ϕ_{00} moments. In practice this means that for a Krylov iteration

$$\mathbf{C}z = \mathbf{RS}\big(\mathbf{I} - \mathbf{DL}^{-1}\mathbf{SM}\big)v \qquad (4.7)$$

is solved for each group, where v is the iteration vector. DSA was found to be useful for some problems. In Denovo DSA has been found to be beneficial for diffusive problems with high scattering ratios that are close to being isotropic [21]. This is consistent with experiences of the wider nuclear community.

4.2.4 Multigrid Methods

Beginning in the late 1980s, the nuclear community started using spatial multigrid and/or angular multigrid as both solvers and preconditioners. The first use of spatial multigrid for transport equations in 1-D and 2-D was investigated by Nowak et. al. Since that time multigrid has been used in multiple dimensions, for both isotropic and anisotropic scattering, and for various spatial discretizations [3]. Some highlights from recent work are discussed below. All are applied to the S_N neutron transport equation unless otherwise noted.

In 1996 Sjoden and Haghighat used a simplified spatial multigrid method that does not use the residual as a solver for the 3-D, parallel code PENTRAN [58]. In 1998 multigrid in space and multigrid in angle were used as preconditioners for Krylov methods and were tested for the 1-D, one-group, modified linear discontinuous (MLD) neutron transport equations by Oliveira and Deng. They looked at isotropic scattering without absorption, isotropic with absorption, and anisotropic cases. They had better results with multigrid than when using ILU as a preconditioner [49].

In 2007 Chang et. al. used 2-D spatial multigrid for the isotropic scattering case with corner balance finite difference in space and a four-color block-Jacobi relaxation scheme. A bilinear interpolation operator, and its transpose, were used for grid transfer. The method had some trouble with heterogeneous problems. The authors assert their algorithm is parallelizable [13].

In 2010 Lee developed a method to do multigrid in space and angle simultaneously for two and three dimensions, isotropic and anisotropic scattering, one energy group, and a variety of spatial discretizations. The method can perform multigrid only space, only angle, or some combination there of. It also handles thick and thin cells [40].

This list is hardly comprehensive, but is somewhat representative of new and recent developments in this area. The author was unable to find any cases where multigrid was used in the energy variable either as a solution technique or as a preconditioner.

4.2.4.1 Two-Grid Acceleration

The two-grid acceleration method developed by Adams and Morel is one of the earlier spatial multigrid methods, which has been built upon by others [2]. It is intended to accelerate convergence of the outer iterations for the transport equation when upscattering is present; the outer iteration method is Gauss Seidel. The original work was done for slab geometry with linear discontinuous (LD) discretization; it is a one-grid, one-cycle scheme in angle. The general approach is expounded upon here to provide an example of multigrid methods applied to the transport equation and because a variation of this method is used in Denovo.

Adams and Morel create a within group error equation by subtracting the GS equation from the transport equation. If the error in iteration k is ϵ^k and the lth moment of the residual is R_l^k, then for group g in 1-D this gives:

$$\mu\frac{\partial\epsilon_g^{k+1}(\mu)}{\partial x} + \Sigma_{t,g}\epsilon_g^{k+1}(\mu) = \sum_{l=0}^{L}\frac{2l+1}{4\pi}\Big(\sum_{g'=1}^{g}\Sigma_{s,g'\to g,l}\epsilon_{g',l}^{k+1} + \sum_{g'=g+1}^{N}\Sigma_{s,g'\to g,l}\phi_{g',l}^{k} + R_{g,l}\Big)P_l(\mu)\ , \tag{4.8}$$

$$\epsilon_g^{k+1}(\mu) = \psi_g(\mu) - \psi_g^{k+1}(\mu)\ , \tag{4.9}$$

$$\epsilon_{g',l}^{k+1} = 2\pi\int_{-1}^{1}\epsilon_{g'}^{k+1}(\mu')P_l(\mu')d\mu'\ , \tag{4.10}$$

$$R_{g,l}^{k+1} = \sum_{g'=g+1}^{N}\Sigma_{s,g'\to g,l}\big(\phi_{g,l}^{k+1} - \Sigma_{g',l}^{k}\big)\ . \tag{4.11}$$

The diffusion approximation is applied to Equation (4.8), giving a coarse grid equation. The coarse grid equation is then solved on the coarse grid for the error, which is in turn added to the flux approximation.

Next, it is assumed that the zeroth moment of the error is a product of a spectral shape function, ξ_g and a space-dependent modulation function, $E(x)$:

$$\epsilon_{g,0}^{k+1}(x) = E(x)\xi_g \text{ , and} \tag{4.12}$$

$$\sum_{g=1}^{N} \xi_g = 1 \, . \tag{4.13}$$

The spectral shape function corresponds to the slowest converging error mode, i.e. the mode to be eliminated. To find the shape function, Fourier analysis is performed on the GS iterative method. The zeroth moment of the cross sections is used to form the Fourier matrix, and then an eigenproblem can be formed. The spectral radius of the isotropic GS matrix is the eigenvalue, and the corresponding eigenvector is the shape function. Because the shape function is dependent on materials, one such calculation must be done for each material region. Note that by using the zeroth moment only the isotropic component of the solution is accelerated.

The coarse grid diffusion equations that result from Equations (4.8) and (4.12) differ slightly in form from the standard diffusion equation in that there is an extra term containing the gradient of the shape function. This is zero in homogeneous regions, but undefined at material interfaces. Adams and Morel found that neglecting the gradient term all together still gave good results for their test problems. This may not be true for more complex cases.

The two-grid method requires the diffusion operator to be consistent with the transport operator, just like in DSA. This requirement can be difficult to meet for multi-dimensional problems, particularly for some spatial discretizations.

4.2.4.2 Two-Grid in Denovo

Denovo has a two-grid acceleration scheme based on the one developed by Adams and Morel. As noted above, the iteration procedure uses a collapsed one-group diffusion equation to correct the low-order Fourier modes [2]. Because of the consistency requirement for the discretization of the diffusion operator in multi-dimensional and multi-material problems this method can fail for systems of interest [21].

The original two-grid method was modified by Evans et. al. to make it applicable for the desired cases by using a one-group transport equation instead of the diffusion equation. The modified method is called transport two-grid (TTG). Adams and Morel showed that the slowest converging spatial modes are diffusive and can be exactly computed in the infinite homogeneous case. To preserve this, the TTG method should give the correct error estimation in that limit [21].

To preserve this, the cross sections used in TTG are calculated to give the same energy-collapsed cross sections as the diffusion equation for the infinite homogeneous case, as seen in Equations (4.15) and (4.16). TTG solves a one-group transport equation for the low-order error in each GS iteration:

$$\hat{\mathbf{\Omega}} \cdot \nabla\psi_\epsilon + \bar{\Sigma}\psi_\epsilon = \frac{1}{4\pi}\bar{\Sigma}_s\phi_\epsilon + \frac{1}{4\pi}\bar{R}\,, \tag{4.14}$$

$$\bar{\Sigma} = \frac{1}{\sum_{g=g_1}^{g_2}\frac{1}{\Sigma^g}\zeta^g}\,, \tag{4.15}$$

$$\bar{\Sigma}_s = \frac{1}{\sum_{g=g_1}^{g_2}\frac{1}{\Sigma^g}\zeta^g} - \sum_{g=g_1}^{g_2}\bigl(\Sigma^g\zeta^g - \sum_{g'=g_1}^{g_2}\Sigma_{s0}^{gg'}\zeta^{g'}\bigr)\,. \tag{4.16}$$

The spatial components of the error are ψ_ϵ and ϕ_ϵ; $\bar{R}$ is the residual. Just as in the original method, it is assumed that the error is separable in space and energy at each iteration: $\epsilon_g^k = \phi_g - \phi_g^k = \phi_\epsilon(\vec{r})\zeta^g$, where ζ^g is a material-dependent spectral function [21].

To execute the TTG scheme in Denovo, a transport sweep in conducted in each group, a residual is calculated, the low-order transport solve described above is performed, an error form of the transport sweep is conducted, and the scalar flux is updated. All of that can be seen in the following equations:

$$\mathbf{L}_g \psi_g^{k+\frac{1}{2}} = \mathbf{M}\big(\mathbf{S}_{gg}\phi_g^{k+\frac{1}{2}} + \sum_{g'=g_1}^{g-1} \mathbf{S}_{gg'}\phi_{g'}^{k+\frac{1}{2}} + \sum_{g'=g+1}^{g_2} \mathbf{S}_{gg'}\phi_{g'}^{k}\big) + q_{eg} \,, \tag{4.17}$$

$$R_g^{k+\frac{1}{2}} = \mathbf{M} \sum_{g'=g+1}^{g_2} \mathbf{S}_{gg'}\big(\phi_{g'}^{k+\frac{1}{2}} - \phi_{g'}^{k}\big) \,, \qquad l = m = 0 \,, \tag{4.18}$$

$$\bar{\mathbf{L}}\psi_\epsilon = \mathbf{M}\bar{\mathbf{S}}\phi_\epsilon + \bar{R} \,, \qquad l = m = 0 \,, \tag{4.19}$$

$$\phi_g^{k+1} = \phi_g^{k+\frac{1}{2}} + \phi_\epsilon \zeta^g \,, \qquad l = m = 0 \,. \tag{4.20}$$

The operators with over-bars use the collapsed cross sections given in Equations (4.15) and (4.16), and $\bar{R} = \sum_{g=g_1}^{g_2} R_g^{k+\frac{1}{2}}$.

The ζ_g term is calculated from the eigenvalue problem obtained through Fourier analysis of the GS method using isotropic scattering, just like the original two-grid method:

$$\big(\mathbf{T}_\Sigma - \mathbf{S}_L + \mathbf{S}_D\big)^{-1}\mathbf{S}_U\zeta = \rho\zeta \,, \tag{4.21}$$

where $\mathbf{T}_\Sigma$ is the diagonal, total cross section matrix. As with the original two-grid method, the TTG method is limited to correcting only the isotropic flux moments. This is an adequate limitation as these moments are dominant in thermal groups where upscattering is most prevalent [21].

The additional cost of the method is like solving one extra group, so for cases with many upscattering groups the cost can be amortized. The acceleration equation can be preconditioned with DSA for additional speed. Finally, a reduced quadrature set can be used when solving Equation (4.19) to limit the cost. Some test problems have shown TTG to be quite effective in improving the speed of convergence for upscatter problems when compared to unaccelerated GS [21].

Clearly, a wide variety of preconditioning techniques have been applied to the neutron transport equation. It is worth noting that many of these methods are dependent upon the choice of spatial discretization employed, only apply to within group iterations, or have other important limitations. Preconditioning Krylov methods for solving the neutron transport problem is an active and vital area of research where much progress has been made and in which there is still much room for development.

4.3 Multigrid in Energy

Preconditioning is a very important part of increasing the robustness of Krylov methods. This is particularly true in this work for two reasons. The first is that the multigroup Krylov solver can create very large Krylov subspaces because it forms the subspaces with multiple-group-sized vectors. As a result, any reduction in iteration count will have a large benefit in terms of both memory and cost-per-iteration. The second is that preconditioning is needed to compensate for the ill-conditioned systems created by RQI so that the eigenvector can be converged.

A new physics-based preconditioner has been added to Denovo to improve the performance of the Krylov solves. It does multigrid in energy. Choosing a physics-based preconditioner supports the goal of accelerating a code that solves a specific equation, not developing an all purpose preconditioner. It makes sense to take advantage of information specific to the neutron transport equations. The past work sections above show physics-based preconditioners often provide more benefit than matrix-based methods. Further, the matrix $\mathbf{A}$ is never formed in Denovo making matrix-based preconditioners impossible anyway.

The concept of multigrid in energy is that the grid is in energy, not space or angle like methods used in other work. The motivation behind this choice is that multigrid methods have been successful at accelerating the transport equation. Using grids in energy rather than space or angle is partially for simplicity, grids in the other dimensions can be complex and expensive, and partially because it seemed like it might work very well. Finally, the convergence behavior of the Krylov iterations inside RQI looks like a very good candidate for multigrid methods.

4.3.1 Method

To make energy grids, the energy group structure is coarsened so each lower grid has fewer groups on it. The finest grid is the input energy structure and the coarsest grid has one or a few groups. Each level has half as many groups as the previous level, rounded up if applicable. This is conceptually straightforward because the equations are only one-dimensional in energy and energy groups can be combined (restricted) and separated (prolonged) linearly.

The implemented restriction operator is a simple averaging scheme. Neighboring fine data are averaged together to make coarse data. Recall that for a grid with spacing h, $2h$ is the next-coarser grid. The moments are restricted as $\phi_g^{2h} = \frac{1}{2}(\phi_{2g}^h + \phi_{2g+1}^h)$ for $g = 1, ..., G-1$. If there are an odd number of groups the lowest energy group is just copied. This scheme was chosen so that the thermal energy groups would retain more granularity, which should improve accuracy for thermal reactors. The moments are restricted every time there is transfer from a given grid to a coarser grid.

The cross sections are restricted from the finest to the coarsest grid during problem initialization and they do not change thereafter. The total and fission cross sections are restricted in the same way as the moments. Scattering is slightly more complicated since it is in two dimensions, g and g'. When there are an even number of groups all cross sections are treated the same way. For $g' = 1, ..., G'$ and $g = 1, ..., G$ where G is the number of groups on coarse grid and $2G+1$ is the number of groups on the fine grid,

$$\Sigma_s^{2h}(g, g') = \frac{1}{4}[\Sigma_s^h(2g, 2g') + \Sigma_s^h(2g+1, 2g') + \Sigma_s^h(2g, 2g'+1) + \Sigma_s^h(2g+1, 2g'+1)] \; . \quad (4.22)$$

Unless there is upscattering in every group, some of the entries in Equation (4.22) will be zero. When there are an odd number of groups Equation (4.22) is used until $g = G-1$ and $g' = G'-1$. For the last group

$$\Sigma_s^{2h}(G, g') = \frac{1}{2}[\Sigma_s^h(2G, 2g') + \Sigma_s^h(2G, 2g'+1)] \qquad \text{for } g' = 0, ..., G'-1 \; ,$$
$$\Sigma_s^{2h}(g, G') = \frac{1}{2}[\Sigma_s^h(2g, 2G') + \Sigma_s^h(2g_1, 2G')] \qquad \text{for } g = 0, ..., G-1 \; ,$$
$$\Sigma_s^{2h}(G, G') = \Sigma_s^h(2G, 2G') \; .$$

To prolong from a coarse grid to a fine, the points that line up between the grids are mapped directly, $\phi_{2g}^h = \phi_g^{2h}$. To fill in the intermediate points on the fine grid, the adjacent coarse values are linearly interpolated, $\phi_{2g+1}^h = \frac{1}{2}(\phi_g^{2h} + \phi_{g+1}^{2h})$. The moments are prolonged every time there is a transfer from a given grid to a finer grid. Since cross sections do not change they are never prolonged.

The choice of how to restrict and prolong between grids is hard-coded into the preconditioner. There are other multigrid method choices that are easier to vary and some of these are taken as user inputs. The number of V-cycles is a user input. One V-cycle goes from the finest grid to the coarsest grid and back up, but the user can choose to do more than one V-cycle per application. That means that for one application of the preconditioner ($v \leftarrow \mathbf{G}(v, b)$), any number of the large Vs seen in Figure 4.5 are concatenated together. One cycle looks like one V, two cycles looks like one W, and so on. The default number of V-cycles is 2. The idea is that each additional V-cycle will remove more error.

At this time the depth of the V-cycle is prescribed by the number of groups. This is set such that the grids will be coarsened until there is only one energy group. The number of grids needed is given by [**?**]

$$\text{floor}\left(\frac{ln(G-1)}{ln(2)}\right) + 2\,. \tag{4.23}$$

Note, this is handled differently with energy sets as discussed below. The depth of the cycle is an option that could be changed in the future if it is found that restricting down to only one group is unnecessary, particularly if there are a large number of groups. Investigating the optimal V-cycle depth is an interesting issue, but beyond the scope of this work.

On each grid that makes up a V-cycle some number of relaxations are performed. The number of relaxations per level, ν, is a user input choice with a default of 2. Doing more relaxations per grid should remove more error overall. The relaxation method selected is weighted Richardson iteration. In Equation (4.24) k is the iteration index, where $k = 1, ..., \nu$, and ω is the weighting parameter. This formulation takes the identity portion of $\mathbf{A}$ into account and b is either $\mathbf{TM}q_e$ or $\frac{1}{k}\mathbf{TMF}\phi$.

$$\begin{aligned}
\phi^{k+\frac{1}{2}} &= \mathbf{TMS}\phi^k + b\,, \text{and} \\
\phi^{k+1} &= \omega\phi^{k+\frac{1}{2}} + (1-\omega)\phi^k\,, \text{ which can be combined to give} \\
\phi^{k+1} &= \omega\mathbf{TMS}\phi^k + \omega b + (1-\omega)\phi^k\,.
\end{aligned} \tag{4.24}$$

The weight is also selected by the user and defaults to 1. In general the weight should be chosen such that $1 \leq \omega < 2$. With over relaxation methods, increasing the weight typically reduces the number of required iterations up until some turning point that is problem dependent, though often close to 2, where the iteration count increases again. While this is the case in methods like successive over relaxation, it may not hold true when applied to weighted Richardson inside of a preconditioner since preconditioner behavior is often a little different than when the method is used as an ordinary solver.

An important principle to keep in mind is that the preconditioner is only attempting to roughly invert $\mathbf{A}$. It therefore might be a good idea to use simplifications besides just using weighted Richardson instead of a Krylov method. In this vein there is an option to use a smaller angle set in the preconditioner than the rest of the solution. For example, the whole problem can be solved at S_{10}, but the preconditioner would only use S_2. There is an input option to specify what to use in preconditioner; the default is whatever is being used by the entire problem. This is only available for problems with vacuum boundary conditions at this time.

Recall that right preconditioning is applied as $\mathbf{AG}^{-1}\mathbf{G}\phi = b$, where $\mathbf{A} = \mathbf{I} - \mathbf{TMS}$. To actually implement this in Denovo, the problem must be broken up into several steps:

$$\text{Define} \qquad y = \mathbf{G}\phi\,; \tag{4.25}$$

$$\text{with a Krylov method solve} \qquad \mathbf{AG}^{-1}y = b\,. \tag{4.26}$$

$$\text{After finding } y\text{, the final step is} \qquad \phi = \mathbf{G}^{-1}y\,. \tag{4.27}$$

Because the Krylov solvers are provided by an external library they cannot be modified easily, so carrying out the second step means the preconditioner must be applied to the iteration vectors handed to the solver. Equations (4.28), (4.29), and (4.30) show how this works.

Let v^j be an iteration vector that represents y, and let z^j be an intermediate iteration vector. At each step below the equations are shown three ways to make clear what is going on algorithmically: the equation being solved, the symbolic representation of the outcome, and the way this is written in multigrid syntax. The first thing is to apply the preconditioner to the intermediate vector:

$$\begin{aligned} \mathbf{G}z^j &= v^j\,, \\ z^j &\approx \mathbf{G}^{-1}v^j\,, \\ z^j &\leftarrow \mathbf{G}(z^j, v^j)\,. \end{aligned} \tag{4.28}$$

Next, apply $\mathbf{A}$ to z^j and set it equal to v^{j+1}; y is also v^{j+1}:

$$\begin{aligned} v^{j+1} &= \mathbf{A}z^j \ , \qquad (4.29) \\ v^{j+1} &\approx \mathbf{A}\mathbf{G}^{-1}v^j \ , \\ y &= \mathbf{A}[z^j \leftarrow \mathbf{G}(z^j, v^j)] \ . \end{aligned}$$

To get the final ϕ from $y = \mathbf{G}\phi$, apply the preconditioner again to an iteration vector w:

$$\begin{aligned} \mathbf{G}w^j &= y \ , \qquad (4.30) \\ w^j &\approx \mathbf{G}^{-1}y \ , \\ \phi &= w^j \leftarrow \mathbf{G}(w^j, y) \ . \end{aligned}$$

The operator is used within the preconditioner to compute the residual, $r = \mathbf{A}\phi - b$. That is the only part of the preconditioning algorithm that "applies $\mathbf{A}$". When doing RQI the default behavior is to use an unshifted operator in the preconditioner. There is an option to use the shifted operator instead. In that case $\mathbf{S}$ becomes $\tilde{\mathbf{S}} = \mathbf{S} + \rho\mathbf{F}$ and the right hand side becomes $(\frac{1}{k} - \rho)\mathbf{TMF}\phi$. The use of the shifted operator can be turned on through an input option.

An important attribute of this preconditioner is that it is parallelizable in energy because it can use the energy sets introduced earlier in this work. There are two ways to handle energy grids and energy sets together. One way is to restrict from G groups down to 1 group just like there weren't any energy sets. This requires cross-set communication as soon as there are fewer groups than sets. This also causes some logistical difficulties related to what data is held by which sets at various points in the calculation.

The other way is to prohibit cross-set communication by having each set do its own "mini" V-cycle. Each set restricts, prolongs, and relaxes on only its groups. Instead of restricting to one group overall, each set restricts to one or two group(s), giving approximately num_sets groups overall. This strategy means there must be at least two groups on every set. The number of grids needed is determined by the set with the minimum number of groups since it will be the first to reach a grid with one group. To choose the number of levels, Algorithm 6 is used.

The communication costs and logistical complications of the first strategy seem likely to overwhelm the benefit gained by going to one group instead of num_sets groups. The second strategy

Algorithm 6 Setting the Number of Grids When There Are Energy Sets

```
num_levels = 1, num_local_min = floor$\left(\frac{\text{num\_groups}}{\text{num\_sets}}\right)$
while (num_local_min > 1)
    num_levels = num_levels + 1
    num_groups = (num_groups + 1) / 2
    num_local_min = floor$\left(\frac{\text{num\_groups}}{\text{num\_sets}}\right)$
```

was chosen because it involves much less communication and overhead cost. The validity of this choice will be commented upon in the results section.

With the implemented energy set strategy there are tradeoffs between the number of sets and number of grids for a fixed number of groups. When there are more sets, more cores can be used at one time and wall time should decrease. When there are fewer sets each V-cycle can go deeper so the preconditioner should be more effective, which will reduce iteration count and hopefully decrease wall time. This tradeoff is also investigated.

4.3.2 Results

Investigating the performance of the preconditioner is not simple because there are so many variables. The varsiety of knobs to turn are grouped into three categories: 1) problem and solver type, 2) preconditioning parameters, 3) number of groups and sets. The problem types and solvers are fixed source, eigenvalue with power iteration, and eigenvalue with Rayleigh quotient iteration. All were solved with the multigroup Krylov solver unless otherwise noted. The preconditioning parameters are weight, number of V-cycles, and number of relaxations per level. The group consideration deals with whether there are many or few groups and how many groups per set were used when doing multisets. Calculations were done to investigate all of these combinations.

4.3.2.1 Fixed Source

Shows that preconditioner reduces multigroup Krylov iteration count, useful even for these problems. True for reflecting and vacuum, few and many groups.

* parameter variation with fixed source vacuum, 10 groups (FxdSrcTst_vac) one material, 10 groups with 5 upscattering, first 3 groups have an isotropic source, P_0, S_4, $3 \times 3 \times 3$ grid, vacuum boundaries, tolerance and upscatter tolerance of 1×10^{-6}. This is a fixed source manager unit test run on my mac with the debug version; no timing information is available. GMRES with multigroup Krylov solver.

See the results for varying the weight with 1 relaxation per level and 1 V-cycles in the top plot in Figure 4.6. The results for varying with number of relaxations per grid and the number of V-cycles with the weight fixed at one can be seen in the bottom plot of that figure. When the preconditioning parameters were really cranked up nothing changed, see Table 4.1.

Table 4.1 Small Fixed Source Vacuum Boundary, With Much Preconditioning

Weight	Relaxations	V-cycles	Iterations
0	0	0	10
2	2	2	4
2	3	3	4
1.3	4	4	4
1.3	10	10	4

* parameter variation with fixed source reflecting, 10 groups (FxdSrcTst_refl) one material, 10 groups with 5 upscattering, first 3 groups have an isotropic source, P_0, S_4, $3 \times 3 \times 3$ grid, reflecting boundaries, tolerance and upscatter tolerance of 1×10^{-6}. This is a fixed source manager unit test run on my mac with the debug version; no timing information is available. GMRES with multigroup Krylov solver.

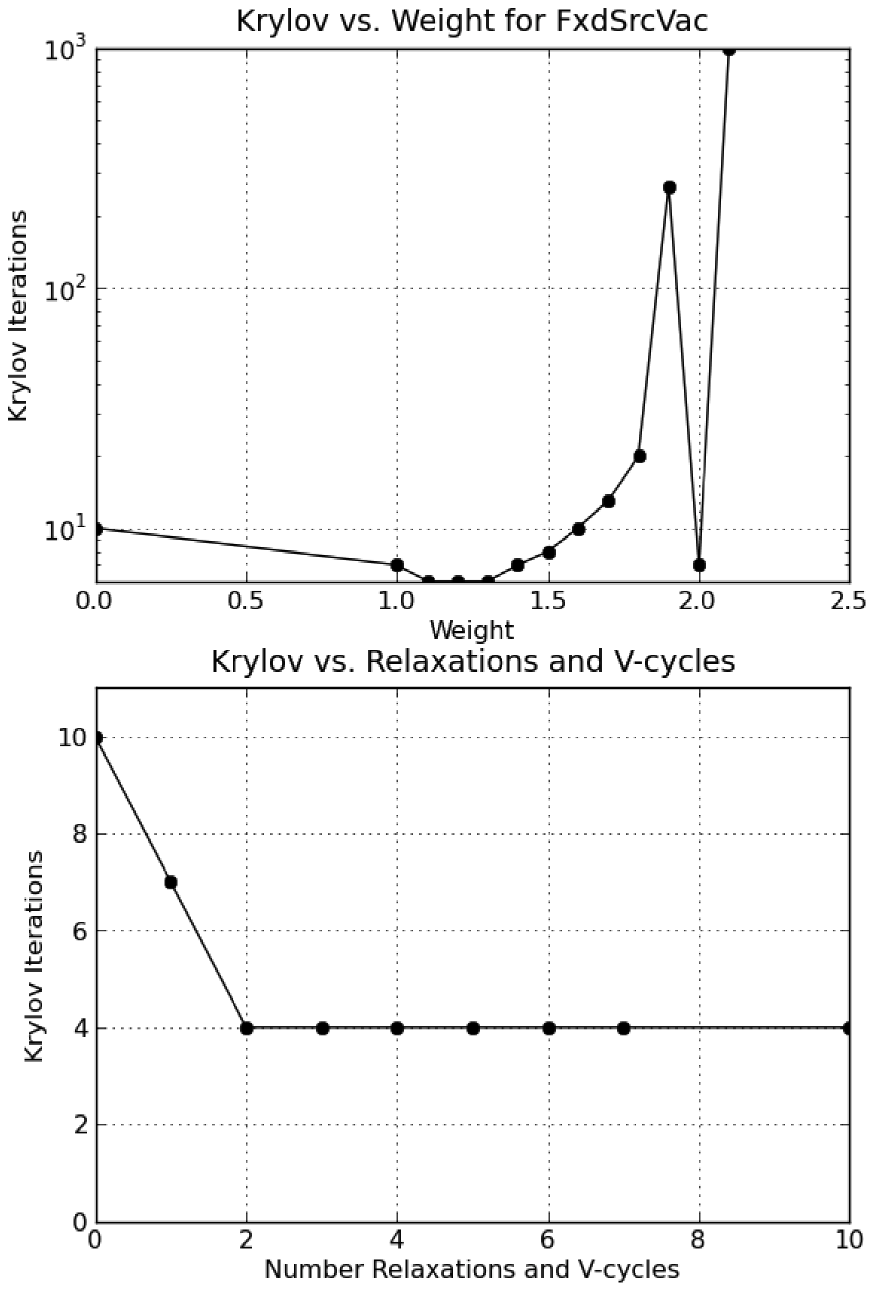


Figure 4.6 Preconditioning Parameter Studies for the Small Fixed Source Problem with Vacuum Boundaries

See the results for varying the weight with 1 relaxation per level and 1 V-cycles in the top plot of Figure 4.7. The results for varying with number of relaxations per grid and the number of V-cycles with the weight fixed at one can be seen in the bottom plot.

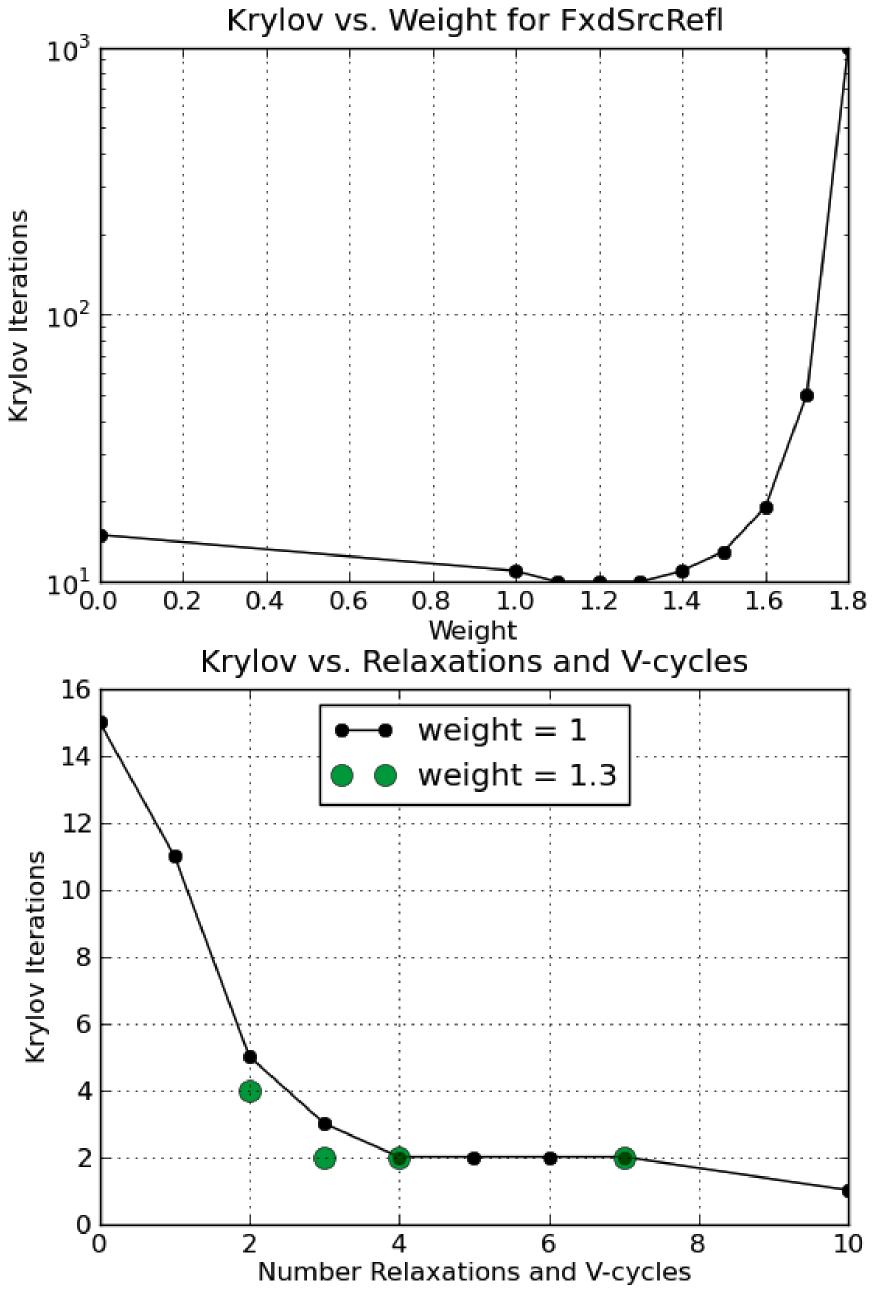


Figure 4.7 Preconditioning Parameter Studies for the Small Fixed Source Problem with Reflecting Boundaries

* comparison of gs, ttg gs, mg krylov, and preconditioned mg krylov + demo of reduced angle solve for fixed source, 27 groups (FeC_vac) The half iron half carbon toy problem from Chapter 2.

Run on the emac with the debug version of the code. $10 \times 10 \times 10$, tolerance and upscatter tolerance of 1×10^{-6}, S_8. A few results comparing solvers can be seen in Table 4.2. The syntax is that w# is the weight, r# is the number of relaxations per level, and v# is the number of V-cycles, e.g. w1r1v1 is one relaxation per level, one V-cycle, and a weight of 1.

This problem also very briefly investigated the option of using a different angle set within the preconditioner. The overall problem was solved with S_8 and the preconditioner only used S_2. For the w1r2v2 case the time was reduced from 7.07 $\times 10^2$ seconds to 1.57 $\times 10^2$ seconds.

Table 4.2 Iron Carbon Fixed Source Cube, Solver and Preconditioning Comparison

Solver	GS iters	Krylov	time (s)
GS	12	1,727	1.12 $\times 10^2$
GS TTG	11	1687	1.99 $\times 10^2$
MG Krylov	n/a	30	8.78 $\times 10^1$
w1 r2 v2	n/a	10	7.07 $\times 10^2$
w1.3 r4 v4	n/a	4	1.29 $\times 10^3$

4.3.2.2 Power Iteration

Shows that reducing mg Krylov iterations in multigroup solves is useful for eigenvalue problems. True for few and many groups, reflecting (vacuum to come)

* parameter variation with power iteration reflecting, 7 groups (2D Orhtanc) Results are in Figure 4.8. All k values were within the uncertainty of the benchmark and so are not reported. These were run on the small orthanc cluster at Oak Ridge with 16 cores; 4 x-blocks and 4 y-blocks, 1 z-block and 1 set. The total and upscatering tolerances were 1 $\times 10^{-3}$, with a k tolerance of 1 $\times 10^{-5}$. In the table "Krylov" is the total number of Krylov iterations and "Eigenvalue" is the total number of Eigenvalue iterations. Two other calculations with a higher level of preconditioning were also done, all preconditioned cases used 31 eigenvalue iterations. When the parameters were

w1.4, r2, v2 the number of Krylov iterations were reduced to 438, and the calculation took 1.77 $\times 10^4$ seconds. For w1r3v1, 253 Krylov iterations and 2.28 $\times 10^4$ seconds were required.

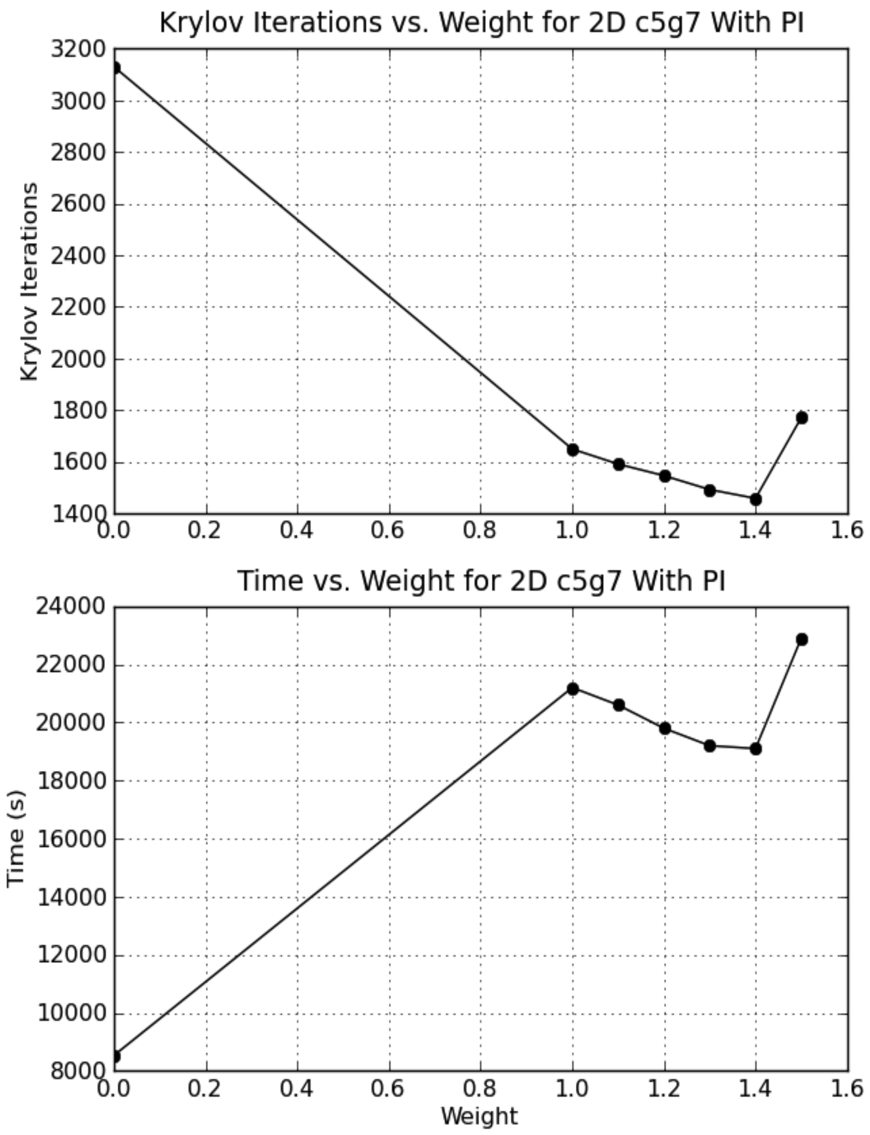


Figure 4.8 Preconditioning Parameter Studies for the Small Fixed Source Problem with Reflecting Boundaries

* parameter variation with power iteration reflecting, 7 groups (3D Orhtanc) The same type of study was done with the 3D benchmark on the oic cluster at Oak Ridge. This used 720 cores with 40 x-blocks, 18 y-blocks and 5 z-blocks. The total and upscattering tolerances were 1 $\times 10^{-4}$,

with a k tolerance of 1×10^{-5} unless otherwise indicated. Results seen in Table 4.3. None of these results were within the uncertainty bounds of the reported benchmark. All results were low by about 0.011.

Table 4.3 3D C5G7 Benchmark With Power Iteration, Preconditioning Parameter Scoping

Weight	Relaxations	V-cycles	Krylov	PI	Time (s)
0	0	0	1,224	32	4.46×10^3
1	1	1	708	32	2.12×10^4
1.2	1	2	448	32	2.38×10^4
1.2	2	1	448	32	2.39×10^4
1.3	2	2	288	32	2.84×10^4
1.5	3	3	192	32	3.73×10^4
1	3	3	126	14*	4.04×10^4
1	4	4	n/a	n/a	exceeded wall time
1	4	4	n/a	n/a*	exceeded wall time
1.5	5	5	n/a	n/a	exceeded wall time

*tol and upscatter tol = 1×10^{-5}, k tol = 1×10^{-3}.

4.3.2.3 RQI

Shows that the preconditioner is useful for problems we can solve with rqi, and it makes rqi possible for problems we couldn't solve before.

* parameter variation with rqi vacuum, 4 groups (RQI_UnitTest_vac) Small RQI unit test with vacuum boundary conditions. This is the same problem that was reported upon in Chapter 3. The correct k and flux were found in all cases. The tolerance used to compare the flux to the reference case was 1×10^{-5}. The results are shown in Table 4.4.

* parameter variation with rqi vacuum + shifted operator, 4 groups (RQI_UnitTest_vac) Small RQI unit test with vacuum boundary conditions. This test used the shifted version of the operator

Table 4.4 RQI Unit Test With Vacuum Boundaries, Preconditioning Parameter Study

Weight	Relaxations	V-cycles	Krylov	RQI
0	0	0	39	6
1	1	1	27	6
1.2	1	1	31	6
1	2	1	16	6
1.2	2	1	19	6
1	1	2	16	6
1.2	1	2	19	6
1	2	2	11	6
1.2	2	2	11	6
1	2	3	10	6
1.2	2	3	10	6
1.3	2	3	10	6
1.4	2	3	10	6
1	3	3	6	6
1	4	4	6	6
1.3	4	4	6	6
1	5	5	6	6
1.3	5	5	6	6

in the preconditioner. Unless otherwise noted the correct k and flux were found. The flux checking tolerance was again 1×10^{-5}. The results are shown in Table 4.5.

* parameter variation with rqi reflecting, 4 groups (RQI_UnitTest_refl) This is the small RQI unit test with reflecting boundary conditions, the same that was discussed in Chapter 3.The flux was not tested against a very tight tolerance, either 1×10^{-2} or 1×10^{-3} unless otherwise noted.

Table 4.5 RQI Unit Test With Vacuum Boundaries and Shifted Operator, Preconditioning Parameter Study

Weight	Relaxations	V-cycles	Krylov	RQI
0	0	0	39	6
1	1	1	20	6*
1.2	1	1	31	7†
1	2	1	12	6
1.2	2	1	16	6
1	1	2	12	6
1.2	1	2	16	6
1	2	2	10	6
1.2	2	2	10	6
1	2	3	6	6

*flux was not correct and k was 0.17632 instead of 0.17528

†flux was not correct and k was 0.17494 instead of 0.17528

The results are show in Table 4.6. Note that here the max number of Krylov iterations was set to 100.

* parameter variation with rqi reflecting + shifted operator, 4 groups (RQI_UnitTest_refl) This is the small RQI unit test with reflecting boundary conditions. This test also used the shifted version of the operator in the preconditioner. Many of these tests did not pass with the preconditioner. The flux was tested against the same loose tolerance unless otherwise noted. The results are show in Table 4.7. The max number of Krylov iterations was set to 100 here as well.

*** conclusions about using the shifted operator ***

* parameter variation with rqi reflecting, 27 groups (im_pi) This is the infinite medium problem from Chapter 3 that has 27 groups and a very small dominance ratio. RQI was originally unable to converge the eigenvector for this problem, but got a close guess for the eigenvalue. These results are shown in Table 4.8. This was solved in serial using the debug version of Denovo on a macbook

Table 4.6 RQI Unit Test With Reflecting Boundaries and Shifted Operator, Preconditioning Parameter Study

Weight	Relaxations	V-cycles	k	Krylov	RQI
0	0	0	2	35	5
1	1	1	2	30	2^{*}
1.2	1	1	2	127	$2^{*\dagger}$
1.3	1	1	2	200	$2^{\dagger}$
1.4	1	1	1.9977	n/a	test failed
0.7	2	2	2	13	2
0.9	2	2	2	12	2
1	2	2	2	12	2
1.2	2	2	2	29	2
1	3	3	2	8	2
1	4	4	2	6	2
1	5	5	2	4	2^{*}
1.2	5	5	2	14	2
1	4	1	2	12	2
1	1	4	2	12	2
1	4	2	2	8	2
1	2	4	2	8	2
1	4	3	2	6	2
1	3	4	2	6	2

*used a tighter comparison tolerance of 1×10^{-5} and still passed

†at least one eigenvector iteration did not converge

pro. A strange issue with this calculation is that most of the time the flux was correct in magnitude,

Table 4.7 RQI Unit Test With Reflecting Boundaries and Shifted Operator, Preconditioning Parameter Study

Weight	Relaxations	V-cycles	k	Krylov	RQI
0	0	0	2	35	2
1	1	1	test failed		
1.4	1	1	test failed		
1	2	2	test failed		
1.4	2	2	test failed		
1	3	3	test failed		
1.4	3	3	test failed		
1	4	1	test failed		
1	4	2	test failed		
1	4	3	test failed		
1	4	4	2	18	6
1	3	4	test failed		
1.2	3	4	2.0024	41	6
1.3	3	4	1.9997	1200	12
1.4	3	4	2.0037	4100	41
1	5	5	2	8	4*
1.2	5	5	2	32	4

*used a tighter comparison tolerance of 1×10^{-5} and still passed

but negative. Note that the timing comparison should not be taken strictly because other work was being done on the computer while these calculations were running.

An important comparison is that is was solved with power iteration using no preconditioning and using a lot of preconditioning. Without preconditioning the problem took 180 Krylov iterations

Table 4.8 Intermediate Problem With Rayleigh Quotient Iteration, Preconditioning Parameter Scoping

Weight	Relaxations	V-cycles	k	Krylov	RQI	Time (s)
0	0	0	0.397	39,025	40*	5.43 $\times 10^4$
1	1	1	0.398	3,014†	4	2.66 $\times 10^4$
1	3	1	0.398	50	3	1.03 $\times 10^4$
1	4	1	0.398	44†	3	1.16 $\times 10^3$
1	2	2	0.398	44†	3	2.86 $\times 10^2$
1	4	4	0.4	16	3	1.99 $\times 10^3$
1	5	4	0.4	14†	3	2.27 $\times 10^2$
1.4	5	4	-0.461	12†	2	1.74 $\times 10^3$
1	5	5	-0.457	7†	2	1.58 $\times 10^3$
1.1	4	1	0.398	43†	3	1.16 $\times 10^3$
1.1	1	4	0.398	43†	3	1.35 $\times 10^3$
1.1	2	2	0.398	43†	3	1.18 $\times 10^3$

*terminated manually

†negative flux with correct magnitude

and 2.57×10^2 seconds. With w1r5v5 preconditioning it took 12 Krylov iterations and 2.19×10^3 sections. Both calculations obtained an eigenvalue of 0.4 and a positive correct flux.

* parameter variation with rqi reflecting using BiCGSTAB, 27 groups (im_pi) This problem was attempted with BiCGSTAB instead of GMRES with no success. The preconditioning was set to 2 relaxations per level and 2 V-cycles, the weight was increased in 0.1 increments from 1.0 to 1.5. In the cases where the calculation terminated itself it was because a negative eigenvalue was found (which automatically ends the calculation), and the answers were entirely incorrect. In the three cases where it didn't; w1.1, w1.2, w1.3; the problems were terminated manually. The eigenvalue was oscillating between a value close to the correct answer and a large number like

11. When a problem is terminated manually there is no way to obtain the flux, so it is unknown whether that was correct or not.

* parameter variation with rqi reflecting, 7 groups (2D Orhtanc) This used the same settings as the power iteration version. In all cases, except the unpreconditioned one, k was within the uncertainty of the benchmark value. This test case was the first to really focus on whether the preconditioner could get RQI to converge. Recall that this was the first test problem for which RQI did not get the right answer at all. The results of this study can be seen in Table 4.9.

Table 4.9 2D C5G7 Benchmark With Rayleigh Quotient Iteration, Convergence Study

Weight	Relaxations	V-cycles	Krylov	RQI	$<$ 1,000?	Time (s)
0	0	0	119,006	120*	no	9.38×10^4
1	1	1	16,007	17	no	2.02×10^5
1.2	1	1	40,008	41*	no	2.06×10^5
1	3	1	n/a	n/a*	7	n/a
1	2	2	11,158	19	alternated	3.99×10^5
1	3	2	3,320	19	14	1.64×10^5
1	3	3	299	19	yes	2.57×10^4
1.1	3	3	281	19	yes	2.40×10^4
1.3	3	3	254	19	yes	2.19×10^4
1.5	3	3	n/a	n/a*	no	n/a

*terminated manually

The "$<$ 1,000?" column indicates whether or not the multigroup iterations converged during the RQI process. If the value is "no" that means the eigenvector only converged during the first iteration when the Rayleigh quotient was not yet a good guess and the system was therefore not yet ill-conditioned. A number indicates the last eigenvalue iteration for which the Krylov method took less than 1,000 iterations. A "yes" means all of the Krylov iterations converged.

These results show a few important things. The first is another example that preconditioning can hold RQI on track enough to get the right eigenvalue, even when the eigenvector is not quite converging. For many of the calculations the eigenvector didn't converge, or didn't converge all the time, but the correct eigenvalue was still found (even in the w1.2r1v1 case). As the preconditioning increased, the eigenvector came closer to converging for all iterations.

The second is that with enough preconditioning, the eigenvector within RQI can be converged and the right eigenpair can be found. For all of the w#r3v3 cases all of the Krylov iterations converged. As a result, the calculation time decreased by an order of magnitude. Increasing the weight then decreased iteration count and wall time up until there was too much weight and the calculation did not converge at all.

* parameter demonstration with rqi reflecting using BiCGSTAB, 7 groups (2D Orthanc)

* parameter variation with rqi reflecting, 7 groups (3D Orhtanc) The 3D benchmark was also done using the same parameters as the power iteration version. The degree to which the RQI-computed k matched the benchmark was the same as PI: within 0.011 when the tighter calculation tolerance was used, but not within the reported uncertainty. The results are in Table 4.10.

What is likely happening when the problems with less preconditioning run out of time is that the eigenvector is not converging. As was seen before, the calculations take a long time when ever eigenvalue iteration uses 1,000 Krylov iterations. It may be that the eigenvalue is correct or nearly correct, but this machine does not store any output from jobs that exceed the wall time limit. Thus, no information about what is happening during the calculation can be gleaned.

When the problem does finish and return results in time, it does so in fewer eigenvalue iterations, fewer Krylov iterations, and less time than Power Iteration. It even manages to finish in time for a calculation when PI did not.

*** conclusions about precond making rqi possible ***

*** conclusions about parameters based on all experiences ***

Table 4.10 3D C5G7 Benchmark With Rayleigh Quotient Iteration, Preconditioning Parameter Scoping

Weight	Relaxations	V-cycles	Krylov	PI	Time (s)
0	0	0	n/a	n/a	exceeded wall time
1	1	1	n/a	n/a	exceeded wall time
1.5	1	1	n/a	n/a	exceeded wall time
1.2	2	1	n/a	n/a	exceeded wall time
1.3	2	2	302	19	2.32×10^4
1	3	3	103	9*	3.02×10^4
1	3	3	164	15†	3.38×10^4
1.5	3	3	187	19	3.24×10^4
1	4	4	n/a	n/a	exceeded wall time
1	4	4	74	9*	2.29×10^4
1.5	5	5	n/a	n/a	exceeded wall time

*tol and upscatter tol = 1×10^{-5}, k tol = 1×10^{-3}

†tol and upscatter tol = 1×10^{-4}, k tol = 5×10^{-5}

4.3.2.4 Multisets

Shows that preconditioner actually works great with multisets and we probably chose the right implementation.

* multisets with fixed source, 27 groups (FeC_vac) The last but certainly not least area of investigation was how the preconditioner faired when using multisets. To investigate its effect on the Krylov iterations without worrying about impacts of an eigenvalue calculation, the iron graphite fixed source problem was investigated. This was run on orhtanc using the optimized version of the code.

To make the problem large enough to be able to use energy sets, the grid was increased to $50 \times 50 \times 50$, though S_4 was used instead of S_8. The unpreconditioned version was compared to one with w1r2v2 on 1 to 10 sets. Note that with 27 groups 10 sets is the maximum possible to still be able to use the preconditioner. In addition, 2 x-blocks, 2 y-blocks, and 1 z-block were used. The calculations therefore used between 4 and 40 cores. With all set combinations the preconditioned calculation took 27 GMRES iterations while the unpreconditioned took 123.

Because the number of iterations did not change with sets, the only thing to compare is time. The focus here is on relative change in time rather than absolute time since there is still room for the preconditioner to be optimized. A Table with the data can be found in Appendix D, Table D.6.

Three plots are shown in Figure 4.9. From the top the first shows the wall time for the preconditioned and unpreconditioned (regular) calculations as a function of time. The second plots the ratio of the time the calculation with N sets took to the time with 1 set. The last plot is of the relative difference between the two times.

This test was also run with increased preconditioning parameters for 1 set and 10 sets to see if that made a difference. The angle set was changed back to S_8 and there was no decomposition in space. When w1.3r4v4 was used the number of Krylov iterations decreased to 11. With 1 set this took 6.49 $\times 10^4$ sec to complete. With 10 sets this was reduced to 5.02 $\times 10^3$ sec. A 10 fold increase in computing power gave more than a 10 fold decrease in run time.

When the problem was not preconditioned 124 Krylov iterations were needed. With 1 set the wall time was 8.81 $\times 10^3$ sec and with 10 sets it was 1.04 $\times 10^3$ sec. Without preconditioning a 10 fold increase in computing power gave less than a 10 fold decrease in run time.

* multisets with power iteration, 27 groups (im_pi) The infinite medium, 27 group problem was also used to study how the preconditioner faired with energy set decomposition. Because this problem has reflecting boundary conditions it could only be used with power iteration and not RQI (recall, RQI cannot do multisets with reflecting boundaries at this time). This problem was also run on orthanc using between 1 and 10 sets. No other problem parameters were changed from what was presented above.

The preconditioning parameters were w1r2v2 which brought the Krylov iterations from 180 (90 Krylov per PI with 2 PI) to 46 (23 Krylov per PI with 2 PI). Three plots are shown in Figure 4.10,

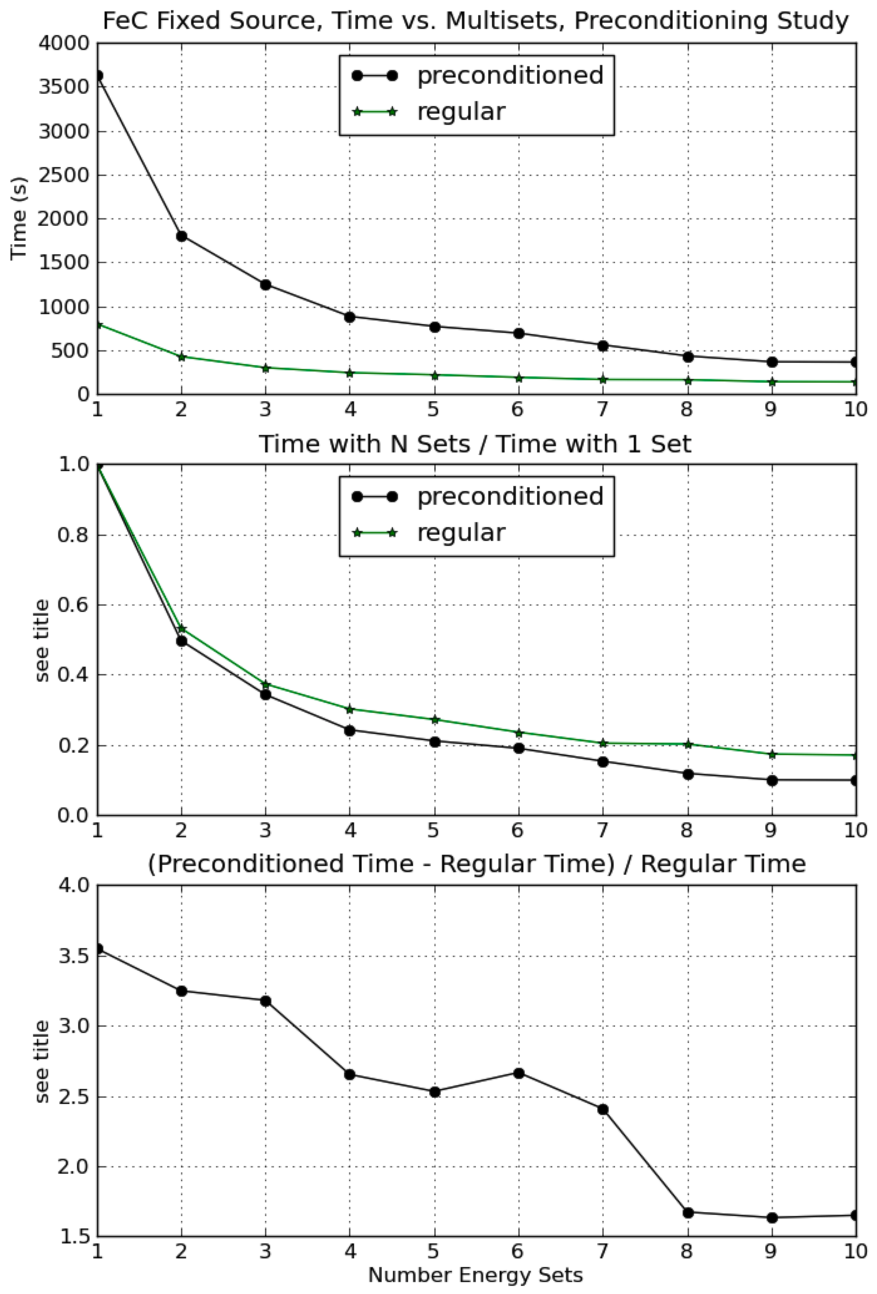


Figure 4.9 Multiset Study with Preconditioner for Iron Graphite Fixed Source Problem

they are of the same variables as in the iron cube problem. A table of the data can be found in Appendix D, Table D.7.

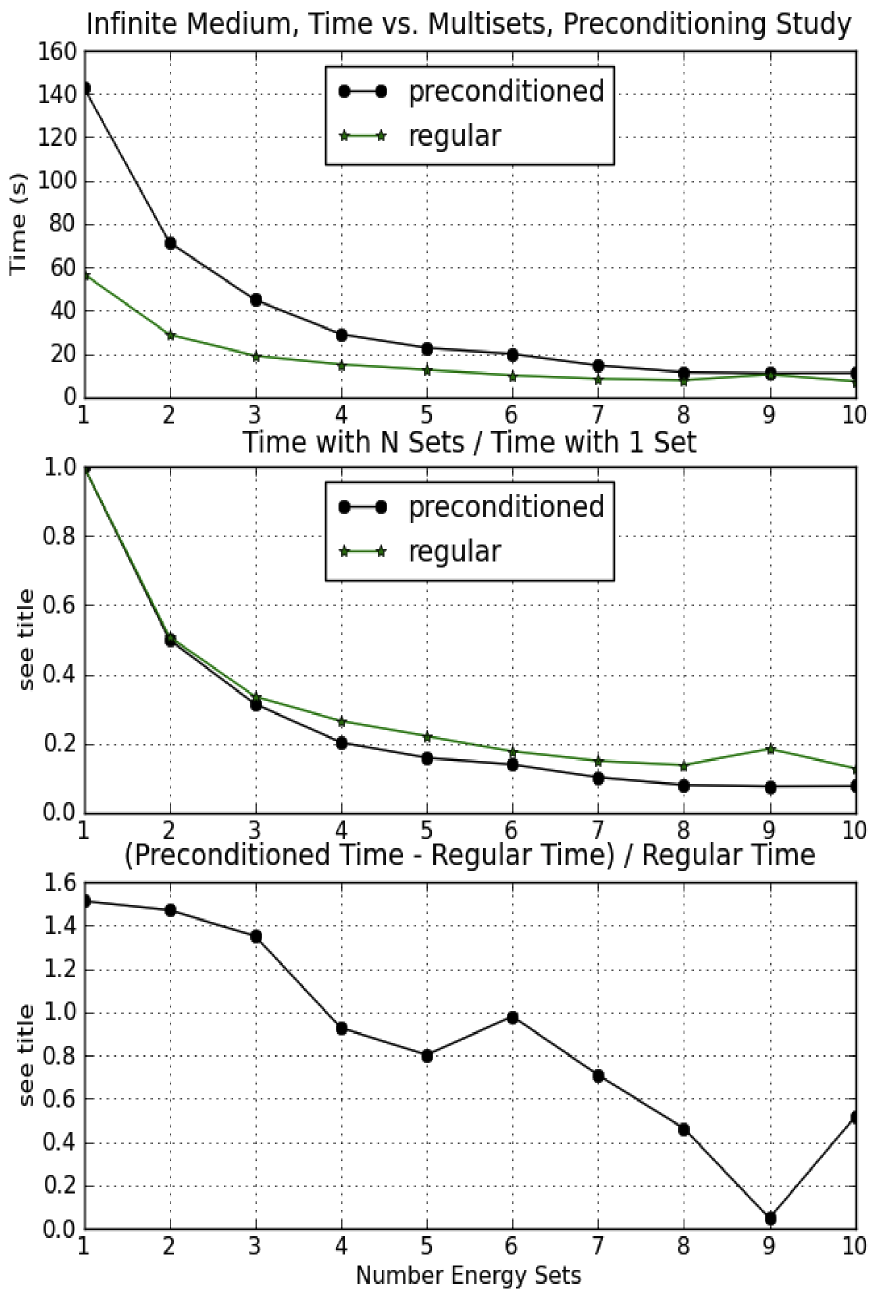


Figure 4.10 Multiset Study with Preconditioner for Infinite Medium Eigenvalue Problem

This test was run with increased preconditioning parameters for 1 set and 10 sets as well. When w1r4v4 was used the number of Krylov iterations decreased to 8 per eigenvalue iteration, and 2 eigenvalue iterations were needed. With 1 set this took 4.02 $\times 10^2$ sec to complete. With 10 sets this was reduced to 2.94 $\times 10^1$ sec. For this problem, too, a 10 fold increase in computing power gave more than a 10 fold decrease in run time.

When the problem was not preconditioned with 1 set the wall time was 56.9 sec and with 10 sets it was 7.32 sec. Once again, without preconditioning a 10 fold increase in computing power gave less than a 10 fold decrease in run time.

* multisets with power iteration, 44 groups (Full_PWR)

* multisets with rqi, 44 groups (Full_PWR)

*** conclusions about set - groups/set tradeoffs and overall usefulness of precond with multi-sets ***

1) this indicates that the improvement from the preconditioner does not come from the depth of V-cycle. If only going down 1 or 2 grids has as much of an impact as going down 6 perhaps it is not always necessary to coarsen down to one group. However, this is only one problem.

2) preconditioning has a super-linear improvement with energy sets. This is because it keeps the number of iterations the same, but each application of the preconditioner becomes less costly since the V-cycle becomes shallower. Once it is optimized this may become less super-linear since it will take a smaller total fraction of the runtime. Nevertheless, this is a good property.

4.3.3 Implications

+

Chapter 5

Conclusions

LIST OF REFERENCES

[1]

[2] B. T. Adams and J. E. Morel. A Two-Grid Acceleration Scheme for the Multigroup Sn Equations with Neutron Upscattering. *Nuclear Science and Engineering*, 115:253–264, 1993.

[3] M. L. Adams and E. W. Larsen. Fast Iterative Methods for Discrete-Ordinates Particle Transport Calculations. *Progress in Nuclear Energy*, 40(1):3–159, 2002.

[4] R E Alcouffe. Diffusion Synthetic Acceleration Methods for the Diamond-Differenced Discrete-Ordinates Equations. *Nuclear Science and Engineering*, 64(1):344–355, 1977.

[5] R E Alcouffe and R S Baker. Time-Dependent Deterministic Transport on Parallel Architectures Using PARTISN. In *Radiation Protection and Shielding Division Topical Conference*, number 836, Nashville, TN, 1998. American Nuclear Society.

[6] E. J. Allen and R. M. Berry. The Inverse Power Method for Calculation of Multiplication Factors. *Annals of Nuclear Energy*, 29(8):929–935, May 2002.

[7] Randal S Baker and Kenneth R Koch. An SN Algorithm for the Massively Parallel CM-200 Computer. *Nuclear Science and Engineering*, 128:312–320, 1998.

[8] Richard Barrett, Berry Michael, Tony F. Chan, Demmel James, June M. Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk Van Der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 1994.

[9] Michele Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *Journal of Computational Physics*, 182:418–477, 2002.

[10] Birkhoff and Lynch. *Numerical Solution of Elliptic Problems*. SIAM, Philadelphia, PA, 1984.

[11] William L. Briggs, Van Emden Henson, and Steve F. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, PA, second edition, 2000.

[12] I.J. Bush and W. Smith. The Weak Scaling of DL_POLY 3, 2010.

[13] B. Chang, T. Manteuffel, S. McCormick, J. Ruge, and B. Sheehan. Spatial Multigrid for Isotropic Neutron Transport. *SIAM Journal on Scientific Computing*, 29(5):1900–1917, 2007.

[14] G Chen and R Sheu. Application of Two Preconditioned Generalized Conjugate Gradient Methods to Three-Dimensional Neutron and Photon Transport Equations. *Progress in Nuclear Energy*, 45(1):11–23, 2004.

[15] M. Dahmani, R. Le Tellier, R. Roy, and A. Hebert. An efficient preconditioning technique using Krylov subspace methods for 3D characteristics solvers. *Annals of Nuclear Energy*, (32):876–896, 2005.

[16] Gregory G Davidson, Thomas M Evans, Rachel N Slaybaugh, and Christopher G Baker. Massively Parallel Solutions to the k-Eigenvalue Problem. In *American Nuclear Society Transations Vol 103*, number 1, La Grange Park, IL, 2010. American Nuclear Society.

[17] James J. Duderstadt and Louis J. Hamilton. *Nuclear Reactor Analysis*. John Wiley & Sons, Inc., New York, 1 edition, 1976.

[18] T M Evans. Denovo Methods Manual. Technical report, Oak Ridge National Lab, Oak Ridge, TN, 2009.

[19] T M Evans. Research Objectives (from INCITE proposal). Technical report, Oak Ridge National Laboratory, Oak Ridge, TN, 2009.

[20] T M Evans. Reflecting Boundary Conditions , 2011.

[21] Thomas M. Evans, Alissa S. Stafford, Rachel N. Slaybaugh, and Kevin T. Clarno. DenovoA New Three-Dimensional Parallel Discrete Ordinates Code in SCALE. Technical report, Oak Ridge National Lab, Oak Ridge, TN, 2009.

[22] Tom Evans and Greg Davidson. Technical Note, Subject: Parallel Energy Decomposition in Denovo (Rev. 1). Technical report, Oak Ridge National Laboratory, Oak Ridge, TN, 2010.

[23] Larry E Fennern. ESBWR Seminar Reactor , Core & Neutronics. In *ESBWR Seminar*. General Electric, 2006.

[24]

[25] A. Gupta and R. S. Modak. On the Use of the Conjugate Gradient Method for the Solution of the Neutron Transport Equation. *Annals of Nuclear Energy*, 29(16):1933–1951, November 2002.

[26] A. Gupta and R. S. Modak. Krylov Sub-space Methods for K-eigenvalue Problem in 3-D Neutron Transport. *Annals of Nuclear Energy*, 31:2113–2125, 2004.

[27] Michael T. Heath. *Scientific Computing An Introductory Survey*. McGraw-Hill, New York, NY, second edition, 2002.

[28] Vicente Hernandez, Jose E. Roman, Anonio M. Vidal, and Vicent Vidal. Calculation of Lamdba Modes of a Nuclear Reactor: A Parallel Implementation Using the Implicitly Restarted Arnoldi Method. In Jose M.L. Plama, Jack Dongarra, and Vicente Hernandez, editors, *Vector and Parallel Processing - VECPAR '98*, pages 43–57, Porto, Portugal, 1998. Springer.

[29] Michael Heroux, Roscoe Bartlett, Vicki Howle, Robert Hoekstra, Jonathan Hu, Tamara Kolda, Richard Lehoucq, Kevin Long, Roger Pawlowski, Eric Phipps, Andrew Salinger, Heidi Thornquist, Ray Tuminaro, James Willenbring, and Alan Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, Albuquerque, NM, 2003.

[30] Michael A Heroux. AztecOO User Guide. Technical Report August, Sandia National Laboratory, Albuquerque, NM, 2007.

[31] Akitoshi Hotta, Yojiro Suzawa, and Hiroyuki Takeuchi. Development of BWR Regional Instability Model and Verification Based on Ringhals 1 Test. *Annals of Nuclear Energy*, 24(17):1403–1427, November 1997.

[32] Ilse C. F. Ipsen. A History of Inverse Iteration. In B. Huppert and E. Schneider, editors, *Mathematical Works, Volume II: Linear Algebra and Analysis*, pages 464–72. Walter De Gruyter, Berlin, 1996.

[33] Ilse C. F. Ipsen and Carl D. Meyer. The Idea Behind Krylov Methods. *Amer. Math. Monthly*, 105(10):889–899, 1998.

[34] Masafumi Itagaki. Matrix-Type Multiple Reciprocity Method Applied to the Modified Helmholtz Neutron Flux Mode in Nuclear Criticality System. *Engineering Analysis with Boundary Elements*, 26:807–812, 2002.

[35] Masafumi Itagaki and Naoki Sahashi. Three-Dimensional Multiple Reciprocity Boundary Element Method for One-Group Neutron Diffusion Eigenvale Computations. *Nuclear Science and Technology*, 33(2):101–109, 1996.

[36] D Knoll and D E Keyes. Jacobian-free NewtonKrylov methods: a survey of approaches and applications. *Journal of Computational Physics*, 193(2):357–397, 2004.

[37] Keisuke Kobayashi, Naoki Sugimura, and Yasunobu Nagaya. 3D Radiation Transport Benchmark Problems and Results for Simple Geometries with Void Regions. Technical report, Nuclear Energy Agency Nuclear Science Committee, OECD, 2000.

[38] T. Kozlowski, T. J. Downar, and E. E. Lewis. Optimal Krylov Preconditioning for Solution of the Multigroup, SP3 Neutron Transport Equations. In *International Conference on Nuclear Technology*, pages 498–499, New Orleans, LA, 2003. American Nuclear Socieity.

[39] Edward W. Larsen. Unconditionally Stable Diffusion-Synthetic Acceleration Method for the Slab Geometry Discrete Ordinates Equations. Part I: Theory. *Nuclear Science and Engineering*, 82(1):47–63, 1982.

[40] B. Lee. A Novel Multigrid Method for SN Discretizations of the Mono-Energetic Boltzmann Transport Equation in the Optically Thick and Thin Regimes With Anisotropic Scattering, Part I. *SIAM Journal on Scientific Computing*, 31(6):4744–4773, 2010.

[41] Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. SIAM, Philadelphia, PA, 1 edition, 2007.

[42] E. E. Lewis. Conjugate Gradient Solutions to the Discrete Ordinates Equations, 1977.

[43] E. E. Lewis and Jr. W. F. Miller. *Computational Methods of Neutron Transport*. American Nuclear Society, La Grange Park, IL, 1 edition, 1993.

[44] John H. Matthews and Kurtis Fink. Numerical Methods Using Mathematica Complementray Supplement for Numerical Analysis, Numerical Methods, 2005.

[45] Noel M. Nachtigal, Satish C. Reddy, and Lloyd N. Trefethen. How Fast are Nonsymmetric Matrix Iterations? *SIAM Journal on Matrix Analysis and Applications*, 13(3):778–795, 1992.

[46] Shoichiro Nakamura. *Computational Methods in Engineering and Science, With Applications to Fluid Dynamics and Nuclear Systems*. John Wiley & Sons, Inc., New York, 1st edition, 1977.

[47] OECD-NEA. Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation, A 2-D/3-D MOX Fuel Assembly Benchmark. Technical report, Nuclear Energy Agency, Organisation for Economic Co-operation and Development, Paris, France, 2003.

[48] OECD-NEA. Benchmark on Deterministic Transport Calculations Without Spatial Homogenisation, MOX Fuel Assembly 3-D Extension Case. Technical Report 5420, Nuclear Energy Agency, Organisation for Economic Co-operation and Development, Paris, France, 2005.

[49] Suely Oliveira and Yuanhua Deng. Preconditioned Krylov Subspace Methods for Transport Equations. *Progress in Nuclear Energy*, 33(1):155–174, 1998.

[50] Christopher C. Paige, Miroslav Rozloznik, and Zdenvek Strakos. Modified Gram-Schmidt (MGS), Least Squares, and Backward Stability of MGS-GMRES. *SIAM Journal on Matrix Analysis and Applications*, 28(1):264–284, 2006.

[51] B N Parlett. Rayleigh Quotient Iteration and Some Generalizations for Nonnormal Matrices. *Mathematics of Computation*, 28(127):679–693, 1974.

[52] Bruce W Patton and James Paul Holloway. Application of preconditioned GMRES to the numerical solution of the neutron transport equation. *Annals of Nuclear Energy*, 29:109–136, 2002.

[53] G. Peters and J. H. Wilkinson. Inverse Iteration, Ill-Conditioned Equations and Newton's Method. *Society for Industrial and Applied Mathematics Review*, 21(3):339–360, 1979.

[54] Gilles L. Ramone, Marvin L. Adams, and Paul L. Nowak. A Transport Synthetic Acceleration Method for Transport Iterations. *Nuclear Science and Engineering*, 125:257–283, 1997.

[55] Massimiliano Rosa, James S. Warsa, and Jae H. Chang. Fourier analysis of inexact parallel block-Jacobi splitting with transport synthetic acceleration. *Nuclear Science and Engineering*, 164:248–263, 2010.

[56] Youcef Saad and Martin H Schultz. Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[57] National Center For Computational Sciences. World's Most Powerful Computer For Science, 2010.

[58] Glenn E. Sjoden and Alireza Haghighat. Simplified Multigrid Acceleration in the PENTRAN 3-D Parallel Code. In *Transactions of the American Nuclear Society*, pages 152–154, Washington, DC, 1996. ANS.

[59] Robert D. Skeel. Numerical Linear Algebra. Number x, chapter Chapter 7, pages 163–164. Robert D. Skeel, West Lafayette, IN, 2006.

[60] Danny C. Sorensen. Implicitly Restarted Arnoldi/Lanczos Methods for Large Scale Eigenvalue Calculations. Technical report, ICASE, NASA Contractor Report 198342, Hampton, VA, 1996.

[61] G. W. Stewart. *Matrix Algorithms Volume II: Eigensystems*. SIAM, Philadelphia, PA, 1 edition, 2001.

[62] Gilbert Strang. Mathematical Methods for Engineers II.

[63] E. Suetomi and H. Sekimoto. Conjugate Gradient Like Methods and Their Application to Eigenvalue Problems for Neutron Diffusion Equation. *Annals of Nuclear Energy*, 18(4):205–227, 1991.

[64] Eiichi Suetomi and Hiroshi Sekimoto. Application of Preconditioned Conjugate Gradient Method to Eigenvalue Problems for One-Group Neutron Diffusion Equation. *Nuclear Science and Technology*, 25(1):100–103, 1988.

[65] Lloyd N. Trefethen and David Bau, III. *Numerical Linear Algebra*. SIAM, Philadelphia, PA, 1997.

[66] G. Verdu, R. Miro, D. Ginestar, and V. Vidal. The Implicit Restarted Arnoldi Method, an Efficient Alternative to Solve the Neutron Diffusion Equation. *Annals of Nuclear Energy*, 26(7):579–593, May 1999.

[67] V. Vidal, G. Verdú, D. Ginestar, and J. L. Muñoz Cobo. Eigenvalues Calculation Algorithms for Lambda-modes Determination. Parallelization Approach. *Annals of Nuclear Energy*, 24(5):387–410, 1997.

[68] V. Vidal, G. Verdú, D. Ginestar, and J. L. Muñoz Cobo. Variational acceleration for Subspace Iteration Method. Application to nuclear power reactors. *International Journal for Numerical Methods in Engineering*, 41(3):391–407, February 1998.

[69] James S Warsa, Todd A Wareing, and Jim E Morel. Krylov Iterative Methods and the Degraded Effectiveness of Diffusion Synthetic Acceleration for Multidimensional SN Calculations in Problems with Material Discontinuities. *Nuclear Science and Engineering*, 147:218–248, 2004.

[70] James S Warsa, Todd A Wareing, Jim E Morel, John M McGhee, and Richard B Lehoucq. Krylov Subspace Iterations for Deterministic k-Eigenvalue Calculations. *Nuclear Science and Engineering*, 147:26–42, 2004.

[71] Wikipedia. Supremum, 2011.

[72] Akio Yamamoto. Generalized Coarse-Mesh Rebalance Method for Acceleration of Neutron Transport Calculations. *Nuclear Science and Engineering*, 151:274–282, 2005.

[73] F. Zinzani, C. Demaziere, and C. Sunde. Calculation of the Eigenfunctions of the Two-Group Neutron Diffusion Equation and Application to Modal Decomposition of BWR Instabilities. *Annals of Nuclear Energy*, 35(11):2109–2125, November 2008.

Appendix A: Diffusion Equation

The diffusion approximation is a widely used simplification that reduces the computational complexity of the transport equation. The approximation is that the angular dependence of the flux is unimportant, so the direction component of the transport equation can be discarded. Physically this means that neutrons move against their concentration gradient like just heat diffuses through a conductor. The information in this section is derived from Duderstadt and Hamilton's *Nuclear Reactor Analysis* [17] and neglects fission for simplicity.

The first step in applying this approximation is to integrate the angular dependence out of Equation (1.1), resulting in the neutron continuity equation:

$$\nabla \cdot J(\vec{r}, E) + \Sigma(\vec{r}, E)\phi(\vec{r}, E) = \int dE' \, \Sigma_s(\vec{r}, E' \to E)\phi(\vec{r}, E') + Q_{ex}(\vec{r}, E) \; , \tag{A.1}$$

where the following definitions have been used:

$J(\vec{r}, E) = \int d\hat{\Omega} \, \hat{\Omega}\psi(\vec{r}, \hat{\Omega}, E)$ is the neutron current

$\phi(\vec{r}, E) = \int d\hat{\Omega} \, \psi(\vec{r}, \hat{\Omega}, E)$ is the scalar flux, and

$Q_{ex}(\vec{r}, E) = \int d\hat{\Omega} \, q_{ex}(\vec{r}, \hat{\Omega}, E)$ is the external source, [17].

Unfortunately, this simplifying approximation added another unknown, J, which leaves one equation with two unknowns. In an attempt to eliminate one of these unknowns, Equation (A.1) is multiplied by $\hat{\Omega}$ and integrated over angle again to obtain the first angular moment:

$$\nabla \cdot \int d\hat{\Omega} \, \hat{\Omega}\hat{\Omega}\psi(\vec{r}, \hat{\Omega}, E) + \Sigma(\vec{r}, E)J(\vec{r}, E) = \int dE' \, \Sigma_{s1}(\vec{r}, E' \to E)J(\vec{r}, E') + \int d\hat{\Omega} \int d\hat{\Omega} \, \hat{\Omega}q_{ex} \; , \tag{A.2}$$

where $\Sigma_{s1} = \int d\hat{\Omega} \, \hat{\Omega}\Sigma_s$. The first angular moment form of the equation cannot be solved either because the streaming (first) term is still unknown.

To make Equation (A.2) solvable, the original assumption is modified to assert that the angular flux is weakly, in fact linearly, dependent on angle rather than independent of angle. To implement this assumption the angular flux is expanded in angle and only the first two terms are retained (see Equation (A.3)). The truncated angular flux is then inserted into the streaming term in Equation

(A.2), giving Equation (A.4).

$$\psi(\vec{r},\hat{\Omega},E) \cong \frac{1}{4\pi}\phi(\vec{r},E) + \frac{3}{4\pi}J(\vec{r},E)\cdot\hat{\Omega}\,, \qquad \text{which gives} \tag{A.3}$$

$$\nabla\cdot\int d\hat{\Omega}\;\hat{\Omega}\hat{\Omega}\psi(\vec{r},\hat{\Omega},E) \cong \frac{1}{3}\nabla\phi(\vec{r},E)\,. \tag{A.4}$$

Next, the scattering source term is simplified in angle and energy. To address the angular dependence define $\bar{\mu}_0$ as the average cosine of the scattering angle, which, temporarily suppressing energy, gives $\Sigma_{s1} = \bar{\mu}_0\Sigma_s$. For elastic scattering from stationary nuclei when s-wave scattering is present in the center of mass frame, $\bar{\mu}_0 = \frac{2}{3A}$ where A is atomic mass number. A common procedure to simplify the energy dependence is to neglect the anisotropic contribution to energy transfer in a scattering collision. Mathematically this means $\Sigma_{s1}(E' \to E) = \Sigma_{s1}(E)\delta(E' = E)$, giving $\int dE'\,\Sigma_{s1}(\vec{r},E' \to E)J(\vec{r},E') = \bar{\mu}_0\Sigma_s(\vec{r},E)J(\vec{r},E)$.

Finally, it is assumed that the external source is isotropic, $\int d\hat{\Omega}\int d\hat{\Omega}\;\hat{\Omega}q_{ex} = 0$.

If these approximations are all included and Equation (A.2) is solved for J, the result is Fick's Law:

$$J(\vec{r},E) \cong -\frac{1}{3(\Sigma(\vec{r},E) - \bar{\mu}_0\Sigma_s(\vec{r},E))}\nabla\phi(\vec{r},E) = -D(\vec{r},E)\nabla\phi(\vec{r},E)\,. \tag{A.5}$$

Fick's Law can be introduced back into Equation (A.1) to obtain the diffusion equation:

$$-\nabla\cdot D(\vec{r},E)\nabla\phi(\vec{r},E) + \Sigma(\vec{r},E)\phi(\vec{r},E) = \int dE'\,\Sigma_s(\vec{r},E' \to E)\phi(\vec{r},E') + q_{ex}(\vec{r},E)\,. \tag{A.6}$$

This equation now includes several assumptions which are valid when the solution is not near a void, boundary, source, or strong absorber. While these requirements can be quite restrictive, the diffusion equation has been used frequently for analysis of nuclear systems throughout the history of the nuclear industry.

Some terms from this section that are used when describing DSA in Chapter 4 are:

$$\Sigma_{tr} = \Sigma(\vec{r},E) - \bar{\mu}_0\Sigma_s(\vec{r},E) \qquad \text{is the transport cross section, and} \tag{A.7}$$

$$\mathbf{C} = -\nabla\cdot\frac{1}{3\Sigma_{tr}}\nabla + \Sigma(\vec{r},E) \qquad \text{is the diffusion operator.} \tag{A.8}$$

Appendix B: Krylov Methods

This appendix is intended to provide the mathematical details of Krylov methods, including information about how restarting Arnoldi and GMRES. While understanding this is not crucial to the developed methods, it is beneficial to be aware of how the underlying solvers work.

B.1 Arnoldi Method

A general issue with Krylov subspaces is that the columns of $\mathcal{K}_k(\mathbf{A}, v_1)$ become increasingly linearly dependent with increasing k. Trying to build solutions out of linear combinations of nearly linearly dependent vectors can fail. The columns of the subspace can be orthogonalized to solve this problem. However, direct orthogonalization will cause some information contained in the subspace to be lost, making more sophisticated factorization methods useful [61].

In general there are two factorization methods upon which many Krylov methods are based. Some use the Arnoldi method, which generates an orthonormal basis for the Krylov subspace for non-normal matrices, and others use the Lanczos method which creates non-orthogonal bases for normal matrices [36], [61]. Denovo has the option of using either restarted GMRES or BiCGSTAB, both of which use the Arnoldi method [18]. The remainder of this subsection will be devoted to developing an understanding of the Arnoldi method.

Fundamentally, Krylov methods are Galerkin or Galerkin-Petrov methods on a Krylov subspace. Galerkin's method uses a few fundamental concepts. One is that an inner product of two functions is zero when the functions are orthogonal: $< f(x), g(x) >= 0$ if $f(x)$ and $g(x)$ are orthogonal. Another is that any function, $f(x)$, in a subspace, $\mathcal{V}$, can be written as a linear combination of the vectors that make a basis for that function space. Let $\mathbf{V} = \{\phi_i(x)\}_{i=0}^{\infty}$ be the basis for $\mathcal{V}$; if $f(x) \in \mathcal{V}$ then $f(x) = \sum_{j=0}^{\infty} c_j \phi_j(x)$ for some scalar coefficients c_j [44].

A weighted residual method is a solution technique for solving some linear problem $\mathbf{A}u = b$ where $u = u_0 + \sum_{i=1}^{n} c_i \phi_i(x)$ is the approximate solution and u_0 is an initial guess. The solution is found by taking the inner product of some arbitrary weight function, $w(x)$, and the residual, $r(x) = b - \mathbf{A}u$, $r(x) \in \mathcal{V}$, such that $< w(x), r(x) >= 0$. The solution is the u satisfying this requirement. Galerkin's method is a weighted residual method where the weight function is chosen from the basis functions: $w(x)$ is selected from $\mathbf{V}$ [44]. In the Galerkin-Petrov method, the weight functions come from a subspace other than $\mathcal{V}$, that is $w(x) \in \mathcal{W}$ [59].

The weighted residual can also be thought of as a process to minimize the residual. There are a few ways to express this idea. If $\hat{u}$ is the exact minimizer of $r(x) = b - \mathbf{A}\hat{u}$, let $u' = \hat{u} + w(x)$ be a close approximation to $\hat{u}$ with $w(x) \in \mathcal{V}$. The residual is minimized if and only if $< w(x), r(x) >= 0$ for all $w(x) \in \mathcal{V}$, meeting the Galerkin condition just described. This can also be written as:

$$\text{find} \qquad u \in u_0 + \mathcal{V} \qquad \text{such that} \qquad r(x) \perp \mathcal{V}\,. \tag{B.1}$$

Further, if y is the solution to

$$\mathbf{V}^T\mathbf{A}\mathbf{V}y = \mathbf{V}^T r_0\,, \qquad \text{then} \qquad \hat{u} = u_0 + \mathbf{V}y\,. \tag{B.2}$$

This $\hat{u}$ minimizes the residual in the some measure of interest, where the measure is determined by the selection of y. In the Petrov-Galerkin formulation, $\mathbf{W}$ is a basis for subspace $\mathcal{W}$ and the idea is to find $u \in u_0 + \mathcal{V}$ such that $r(x) \perp \mathcal{W}$. Now $\mathbf{V}^T\mathbf{A}\mathbf{W}y = \mathbf{V}^T r_0$ [59].

Sorensen [60] and Stewart [61] use the Galerkin Method in combination with Ritz pairs to understand the Arnoldi method, and this viewpoint will be developed here. To maintain consistency with the main portion of this work, the subspace from which solutions are derived in the next paragraphs will be the Krylov subspace $\mathcal{K}_k(\mathbf{A}, v_1)$, and the equation will be switched to $\mathbf{A}x = b$.

A vector $z \in \mathcal{K}_k(\mathbf{A}, v_1)$ is defined as a Ritz vector with corresponding Ritz value, θ, if it satisfies the Galerkin condition $< w, \mathbf{A}z - \theta z >= 0$ for all $w \in \mathcal{K}_k(\mathbf{A}, v_1)$. To see why Ritz pairs can be important, define $\hat{p}(\mathbf{A})$ to be the minimum characteristic polynomial of $\mathbf{A}$ such that $||\hat{p}(\mathbf{A})v_1|| \leq ||q(\mathbf{A})v_1||$ for all monic polynomials $q \neq \hat{p}$ of degree k. If (θ, z) is a Ritz pair for $\mathbf{A}$, then $\mathbf{A}z - z\theta = \gamma\hat{p}(\mathbf{A})v_1 = g$ for some scalar γ. When $g = 0$, then the Ritz pair is an eigenpair. When g is small, the Ritz pair is likely a close approximation to an eigenpair of $\mathbf{A}$ [60].

These ideas can be assembled to understand why the Arnoldi method works. Let $\mathbf{V}$ be a basis for a Krylov subspace $\mathcal{K}_k(\mathbf{A}, v_1)$, and $\mathcal{X} \in \mathcal{K}_k(\mathbf{A}, v_1)$ be an eigenspace of $\mathbf{A}$. When solving $\mathbf{A}x = b$, the subspace $\mathcal{K}_k(\mathbf{A}, v_1)$ and the vector x must satisfy

1. $x \in \mathcal{K}_k(\mathbf{A}, v_1)$ and
2. $r = \mathbf{A}x - b \perp \mathcal{K}_k(\mathbf{A}, v_1)$.

Let $\mathbf{H} = \mathbf{V}^T\mathbf{A}\mathbf{V}$. There is an eigenpair (θ, x) of $\mathbf{H}$ such that $(\theta, \mathbf{V}x)$ is an eigenpair of $\mathbf{A}$. To reduce notational clutter let $z = \mathbf{V}x$, giving $\mathbf{A}z = \theta z$ [61].

If $\mathbf{V}$ is an orthonormal basis for $\mathcal{K}_k(A, v_1)$, then (θ, z) is a Ritz pair if and only if $x = \mathbf{V}y$ with $\mathbf{H}y = \theta y$ for some y. Noting the definition of $\mathbf{H}$, comparing to Equation (B.2), and doing some basic matrix manipulation, it is clear that this y minimizes the residual. As the residual tends toward zero, the Ritz pair converges to an approximate eigenpair of $\mathbf{A}$ [61]. Another way to state this is to revisit the polynomial identity expressing the minimum residual. It can be shown that $\mathbf{A}\mathbf{V}y - \mathbf{V}\mathbf{H}y = \gamma\hat{p}(\mathbf{A})v_1 = g$. As $g \rightarrow 0$, the Ritz pair approaches the eigenpair of $\mathbf{A}$ [60].

In summary, the eigenpairs of $\mathbf{A}$ are approximated by the eigenpairs of $\mathbf{H}$. These eigenvalues and/or eigenvectors are subsequently used in different ways by different Krylov methods to formulate the solution to $\mathbf{A}x = b$ to achieve specific goals, like minimizing the residual in a certain norm. The Galerkin condition is used to ensure the eigenpairs of $\mathbf{H}$ become increasingly good approximations to those of $\mathbf{A}$ as the size of the Krylov subspace increases.

The Arnoldi method is a process of establishing the $\mathbf{V}$ and $\mathbf{H}$ discussed above [61]. $\mathbf{H}$ is an orthogonal projection of $\mathbf{A}$ onto the basis $\mathbf{V}$ and is upper Hessenberg in form. The Gram-Schmidt method (or the modified Gram-Schmidt method) computes $\mathbf{V}$. The Arnoldi method generates a Ritz estimate for the Ritz pair at each iteration. Using these terms and ideas, the Arnoldi algorithm shown in Algorithm 7 can be understood [56], [60]:

The kth step of an Arnoldi factorization can be written as:

$$\mathbf{A}\mathbf{V}_k = \mathbf{V}_k\mathbf{H}_k + g_k e_k^T \ , \tag{B.3}$$

where e_k is the k^{th} column of the identity matrix and g is the residual. An alternate way to derive the Arnoldi method is as a truncation of the reduction of $\mathbf{A}$ to Hessenberg form using shifted QR-iteration [60].

Algorithm 7 Orthogonal Arnoldi Method

```
$r_0 = b - \mathbf{A}x_0$
$v_1 = \frac{r_0}{||r_0||}$
For $j = 1$ to $k$:
    $h_{i,j} = v_i \mathbf{A} v_j, i = 1, \ldots, j$
    $\hat{v}_{j+1} = \mathbf{A}v_j - \sum_{i=1}^{j} v_j h_{i,j}$
    $h_{j+1,j} = ||\hat{v}_{j+1}||$
    $v_{j+1} = \frac{\hat{v}_{j+1}}{h_{j+1,j}}$
End For
Form the solution $x_k = x_0 + \mathbf{V}_k y_k$, where $y_k = \mathbf{H}_k^{-1}||r_0||e_1$
```

B.2 Restarting Arnoldi

As mentioned above, Krylov methods can become untenable if the subspace is allowed to become large. Restart methods have been developed to handle this. After m iterations the Arnoldi process is halted and a new starting vector, $\hat{v}_1$, is constructed from information gained during the previous m iterations. This new vector is used as the starting vector for another round of the Arnoldi process, which is now using the subspace $\mathcal{K}_m(\mathbf{A}, \hat{v}_1)$ rather than $\mathcal{K}_m(\mathbf{A}, v_1)$. This process is repeated, finding new a starting vector every time the subspace reaches dimension m, until convergence. It is assumed that of the m eigenvalues available, only r of them are wanted. Two such methods, explicit and implicit, will be discussed here from the viewpoint of how they work and how they preserve the information of interest.

Explicit restarting applies a polynomial to the starting vector that is designed to damp unwanted components of the eigenvector expansion. There are a few different forms of explicit restarting, but the basic idea begins with splitting the Ritz values into a wanted set, Ω_w and an unwanted set Ω_u. In one form of explicit restarting, a combination of wanted eigenvalues are used and in another form a combination of unwanted eigenvectors are used. In both formulations a filter polynomial is used to create a $\hat{v}_1$ that minimizes the residual g by removing the unwanted information. The components of the new starting vector therefore become more aligned toward the directions of interest. Unfortunately, filter polynomials are expensive to apply directly and this motivates the need for a more sophisticated restart method [60], [61].

Implicit restarting is designed to be less expensive than explicit restarting. It uses an implicitly shifted QR mechanism to compress the factorization while retaining information of interest. The idea is to go from a factorization of length $m = p + r$ to one of length r by applying p shifts implicitly. For $j = 1, 2, \ldots, p$, a QR factorization is done with shifts μ_j where the shifts are derived from the spectrum of $\mathbf{H}$, $\sigma(\mathbf{H}) \equiv \{\lambda \in \mathbb{R} : rank(\mathbf{A} - \lambda\mathbf{I}) < n\}$. This gives an r-step Arnoldi factorization of the form $\mathbf{A}\mathbf{V}_r\mathbf{Q} = (\mathbf{V}_r\mathbf{Q})(\mathbf{Q}^T\mathbf{H}_r\mathbf{Q}) + g_k e_k^T$. $\mathbf{Q} = \mathbf{Q}_1\mathbf{Q}_2 \ldots \mathbf{Q}_p$ where $\mathbf{Q}_j$ is an orthogonal matrix corresponding to shift μ_j, and $g_r = (\mathbf{V}_r\mathbf{Q})e_{r+1}\beta_r + g_m\sigma(\mathbf{H})$ [60].

The shift selection strategy can be designed to create a minimal $\hat{v}_1$. The shifted QR factorization reduces the subspace size while eliminating the undesired components. In essence this is implicitly applying a filter polynomial while avoiding matrix-vector multiplications that would be required in the explicit method. Further, $\mathcal{K}_p(\mathbf{A}, v_1) \subset \mathcal{K}_m(\mathbf{A}, \hat{v}_1)$, which means the wanted Ritz vectors are still represented in the updated subspace. The implicit method is more efficient and more numerically stable than the explicit restart method [60], [61].

B.3 GMRES

One of the more popular Krylov methods, which uses the Arnoldi process, is the GMRES algorithm developed by Saad and Schultz [56], [36]. Recall that $\mathbf{V}_k$ is an orthonormal basis for the Krylov subspace $\mathcal{K}_k(\mathbf{A}, v_1)$ and that $\mathbf{H}_k$ is the representation of the part of $\mathbf{A}$ that is in this Krylov subspace formed from the basis $\mathbf{V}_k$. The notation $\bar{\mathbf{H}}^k$ is the $(k + 1) \times k$ upper Hessenberg matrix that includes the newest $h_{k+1,k}$ element generated in the Arnolodi process [56].

The distinguishing factor for different Krylov methods is the way in which they select the y_k that makes the solution $x_k = x_0 + \mathbf{V}_k y_k$. Let $z = \mathbf{V}_k y_k$. GMRES uses the least squares procedure to find the y_k that minimize the norm of the residual over z in $\mathcal{K}_k(\mathbf{A}, v_1)$. To accomplish this, the least squares problem

$$\min_{z \in \mathcal{K}_k} ||r_0 - \mathbf{A}z|| \tag{B.4}$$

is solved, giving the solution $z = \mathbf{V}_k y_k$. The y_k that is selected minimizes $||\beta e_1 - \bar{\mathbf{H}}^k y||$, where $\beta = ||r_0||$ [56].

GMRES picks the best solution within the Krylov subspace with respect to minimizing the residual [33]. This method cannot break down unless it has already converged, in which case the answer has been found. GMRES has all of the storage issues associated with Krylov methods mentioned before, but restarted GMRES can be used to mitigate this concern [56].

Appendix C: Reflecting Boundaries

Algorithm 3 works for problems with vacuum boundary conditions. However, with reflecting boundary conditions some modifications must be made to the calculation of the Rayleigh quotient. To compute the correct RQ the reflected boundary terms must be excluded when computing the dot products, but their contribution to the real moments must be included properly. It was not possible to do this using the solvers in Denovo when this work was begun. How Denovo handles reflecting boundary conditions and what was done to fix this problem is detailed here. The information in this section is derived from conversations with Tom Evans [] and a paper by Warsa et. al. [69].

Denovo stores the boundary condition information between the flux moments in the solution vector. For the following discussion let the moments for group g be held in ϕ_g, which is of length $f_g = t \times c \times u$ (these sizes are defined in Chapter 1). Let the reflected angular flux for that group be $\psi_{r,g}$, which is of length $m = n \times c_r \times u$ where c_r is the number of cells on the reflecting boundaries. For each group the solution vector and associated source are of length $f_g + m$ and look like

$$\Phi_g = \begin{pmatrix} \phi_g & \psi_{r,g} \end{pmatrix}^T , \qquad Q = \begin{pmatrix} q & 0 \end{pmatrix}^T . \tag{C.1}$$

To exclude the boundary terms when computing the dot products in the RQ, Φ_g is simply truncated to only include ϕ_g.

Understanding the next issue is aided by considering what the operator form of the transport equation for one group looks like with reflecting boundary conditions. First recall the transport equation for group g that *does not* include reflecting boundaries, where group indices are excluded for notational brevity:

$$\mathbf{L}\psi = \mathbf{MS}\phi + q \tag{C.2}$$

$$\big(\mathbf{I} - \mathbf{DL}^{-1}\mathbf{MS}\big)\phi = \mathbf{DL}^{-1}q \, . \tag{C.3}$$

To include reflecting boundaries, the operators are expanded in the appropriate dimensions from f_g to $f_g + m$ to act on Φ. The reflecting operators will be denoted by a tilde. The transport operator is split into volume (traditional) and reflected components: $\tilde{\mathbf{L}} = \mathbf{L}_v + \mathbf{L}_r$. The other terms become:

$$\tilde{\mathbf{I}} = \begin{pmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{I}_r \end{pmatrix}, \quad \tilde{\mathbf{D}} = \begin{pmatrix} \mathbf{D} & 0 \\ 0 & \mathbf{I}_r \end{pmatrix}, \quad \tilde{\mathbf{M}} = \begin{pmatrix} \mathbf{M} & 0 \\ 0 & \mathbf{I}_r \end{pmatrix}, \quad \tilde{\mathbf{S}} = \begin{pmatrix} \mathbf{S} & 0 \\ 0 & \mathbf{I}_r \end{pmatrix}.$$

The operators in Equation (C.3) can now be replaced by the reflecting operators, and then doing a transport solve will converge both the moments and the reflected terms at once.

When computing the Rayleigh quotient, however, the operator is applied but a full solve is not done. With the standard implementation this does not converge ψ_r and those terms therefore do not contribute to the solution properly.

To address this the reflecting boundaries must be converged first by solving each group separately while excluding within-group scattering. Explicitly multiplying the reflecting matrices illustrates what is happening here. The inverse of the reflecting transport operator is

$$\tilde{\mathbf{L}}^{-1} = \begin{pmatrix} \mathbf{I} \\ \mathbf{P}^T \end{pmatrix} \begin{pmatrix} \mathbf{L}_v^{-1} & -\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P} \end{pmatrix} = \begin{pmatrix} \mathbf{L}_v^{-1} & -\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P} \\ \mathbf{P}^T\mathbf{L}_v^{-1} & -\mathbf{P}^T\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P} \end{pmatrix}, \tag{C.4}$$

where $\mathbf{P}$ is a projection operator that projects $\psi_r \rightarrow \psi$. The multiplication makes the system:

$$\left[\begin{pmatrix} \mathbf{I} & 0 \\ 0 & \mathbf{I}_r \end{pmatrix} - \begin{pmatrix} \mathbf{D}\mathbf{L}_v^{-1}\mathbf{M}\mathbf{S} & -\mathbf{D}\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P} \\ \mathbf{P}^T\mathbf{L}_v^{-1}\mathbf{M}\mathbf{S} & -\mathbf{P}^T\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P} \end{pmatrix} \right] \begin{pmatrix} \phi \\ \psi_r \end{pmatrix} = \begin{pmatrix} \mathbf{D}\mathbf{L}_v^{-1} & -\mathbf{D}\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P} \\ \mathbf{P}^T\mathbf{L}_v^{-1} & -\mathbf{P}^T\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P} \end{pmatrix} \begin{pmatrix} q \\ 0 \end{pmatrix}. \tag{C.5}$$

Now there are two equations with two unknowns that can be examined for similarities.

$$\phi - \mathbf{D}\mathbf{L}_v^{-1}\mathbf{M}\mathbf{S}\phi + \mathbf{D}\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P}\psi_r = \mathbf{D}\mathbf{L}_v^{-1}q \tag{C.6}$$

$$\psi_r - \mathbf{P}^T\mathbf{L}_v^{-1}\mathbf{M}\mathbf{S}\phi + \mathbf{P}^T\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P}\psi_r = 0 \tag{C.7}$$

Comparing the reflecting case to the standard case, $\mathbf{P}^T$ replaces $\mathbf{D}$ and $-\mathbf{L}_r\mathbf{P}$ acts like $\mathbf{MS}$. Equation (C.7) can be rearranged and solved for ψ_r. In Denovo this amounts to solving each group separately to converge the reflecting flux where the group source is $\mathbf{P}^T\mathbf{L}_v^{-1}\mathbf{M}\mathbf{S}\phi$. The reflecting boundaries will then contribute properly when solving Equation (C.6) where the effective source becomes $q - \mathbf{D}\mathbf{L}_v^{-1}\mathbf{L}_r\mathbf{P}\psi_r$.

In practice, the process to apply $\mathbf{A}$ in Denovo when calculating the Rayleigh quotient is:

$$\tilde{\mathbf{L}}z = v \;, \qquad y = \tilde{\mathbf{D}}z \;. \tag{C.8}$$

Now Equation (C.8) looks just like Equation (C.2) where the within-group scattering source is set to zero ($\mathbf{S} = 0$) and $v = q$.

An alternative way to express the same idea using the traditional operators using the $\mathbf{L}_v$ and $\mathbf{L}_r$ terms can be informative as well. This is presented in source iteration notation, where the reflecting boundaries are lagged. This is exactly the same idea as above, though iteration indices were excluded.

$$\psi^{l+1} = \mathbf{L}_v^{-1}\big(\mathbf{MS}\phi^l - \mathbf{L}_r\mathbf{P}\psi_r^l\big) + \mathbf{D}q \tag{C.9}$$

$$\phi^{l+1} = \mathbf{D}\psi^{l+1} \;. \tag{C.10}$$

A new solver was written that solves each group separately to converge the reflecting boundaries. This solver is used to compute the Rayleigh quotient when there are reflecting boundaries. This has worked well, but has not been implemented for multisets because each group must be solved separately.

Appendix D: Supplementary Preconditioning Results

This appendix contains tables of the results for which there are only figures in Chapter 4. The tables are simply titled with monikers matching those in the Preconditioning Chapter.

Table D.1 Small Fixed Source Vacuum Boundary, Weight Variation Study

Weight	Relaxations	V-cycles	Iterations
0	0	0	10
1	1	1	7
1.1	1	1	6
1.2	1	1	6
1.3	1	1	6
1.4	1	1	7
1.5	1	1	8*
1.6	1	1	10*
1.7	1	1	13
1.8	1	1	20
1.9	1	1	265
2	1	1	7
2.1	1	1	1000†

* One group in one cell failed a test

† All tests failed

Table D.2 Small Fixed Source Vacuum Boundary, Relaxation Count and V-cycle Variation Study

Weight	Relaxations	V-cycles	Iterations
0	0	0	10
1	1	1	7
1	2	2	4
1	3	3	4
1	4	4	4
1	5	5	4
1	6	6	4
1	7	7	4
1	10	10	4

Table D.3 Small Fixed Source Reflecting Boundary, Weight Variation Study

Weight	Relaxations	V-cycles	Iterations
0	0	0	15
1	1	1	11
1.1	1	1	10
1.2	1	1	10
1.3	1	1	10
1.4	1	1	11
1.5	1	1	13
1.6	1	1	19
1.7	1	1	50
1.8	1	1	1000[†]

[†] All tests failed

Table D.4 Small Fixed Source Reflecting Boundary, Relaxation Count and V-cycle Variation Study

Weight	Relaxations	V-cycles	Iterations
0	0	0	15
1	1	1	11
1	2	2	5
1	3	3	3
1	4	4	2
1	5	5	2
1	6	6	2
1	7	7	2
1	10	10	1
1.3	2	2	4
1.3	3	3	2
1.3	4	4	2
1.3	7	7	2

Table D.5 2D C5G7 Benchmark Preconditioning With Power Iteration, Parameter Study

Weight	Relaxations	V-cycles	Krylov	PI	Time (s)
0	0	0	3,129	32	8.54×10^3
1	1	1	1,649	31	2.12×10^4
1.1	1	1	1,591	31	2.06×10^4
1.2	1	1	1,546	31	1.98×10^4
1.3	1	1	1,492	31	1.92×10^4
1.4	1	1	1,458	31	1.91×10^4
1.5	1	1	1,771	31	2.29×10^4

Table D.6 Multiset Study with Preconditioner for Iron Graphite Fixed Source Problem

# Sets	Precond T (s)	Regular T (s)	Rel Diff	P T Set N / Set 1	R T Set N / Set 1
1	3.64×10^{3}	8.00×10^{2}	3.56	1.00	1.00
2	1.81×10^{3}	4.26×10^{2}	3.24	4.96×10^{-1}	5.32×10^{-1}
3	1.25×10^{3}	2.99×10^{2}	3.18	3.43×10^{-1}	3.74×10^{-1}
4	8.84×10^{2}	2.42×10^{2}	2.65	2.42×10^{-1}	3.02×10^{-1}
5	7.70×10^{2}	2.18×10^{2}	2.54	2.11×10^{-1}	2.72×10^{-1}
6	6.93×10^{2}	1.89×10^{2}	2.67	1.90×10^{-1}	2.36×10^{-1}
7	5.59×10^{2}	1.64×10^{2}	2.41	1.53×10^{-1}	2.05×10^{-1}
8	4.33×10^{2}	1.62×10^{2}	1.67	1.19×10^{-1}	2.03×10^{-1}
9	3.66×10^{2}	1.39×10^{2}	1.64	1.01×10^{-1}	1.74×10^{-1}
10	3.63×10^{2}	1.37×10^{2}	1.65	9.95×10^{-2}	1.71×10^{-1}

Table D.7 Multiset Study with Preconditioner for Infinite Medium Eigenvalue Problem

# Sets	Precond T (s)	Regular T (s)	Rel Diff	P T Set N / Set 1	R T Set N / Set 1
1	1.43×10^{2}	5.69×10^{1}	1.51	1.00	1.00
2	7.14×10^{1}	2.89×10^{1}	1.47	4.99×10^{-1}	5.08×10^{-1}
3	4.49×10^{1}	1.91×10^{1}	1.35	3.14×10^{-1}	3.35×10^{-1}
4	2.91×10^{1}	1.51×10^{1}	0.93	2.03×10^{-1}	2.65×10^{-1}
5	2.27×10^{1}	1.26×10^{1}	0.80	1.59×10^{-1}	2.22×10^{-1}
6	2.00×10^{1}	1.01×10^{1}	0.99	1.40×10^{-1}	1.77×10^{-1}
7	1.46×10^{1}	8.54	0.71	1.02×10^{-1}	1.50×10^{-1}
8	1.15×10^{1}	7.86	0.46	8.05×10^{-2}	1.38×10^{-1}
9	1.10×10^{1}	1.05×10^{1}	0.05	7.73×10^{-2}	1.84×10^{-1}
10	1.11×10^{1}	7.32	0.52	7.77×10^{-2}	1.29×10^{-1}