NON-LINEAR S_2 ACCELERATION FOR MULTIDIMENSIONAL PROBLEMS WITH UNSTRUCTURED MESHES

A Dissertation

by

JOSHUA THOMAS HANOPHY

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
to meet requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Jim Morel
Committee Members,	Jean Ragusa
	Matthias Maier
	Marvin Adams
Head of Department,	Michael Nastasi

May 2022

Major Subject: Nuclear Engineering

ABSTRACT

The S_N transport equation is popularly used to describe the distribution of neutrons in many applications including nuclear reactors. The topic of this research is a non-linear acceleration method for accelerating convergence of the scalar flux when the S_N equation is solved iteratively. The S_N angular flux iterate is used to compute average direction cosines in each octant. These direction cosines define a vector in each octant that may not have a unit length. Nonetheless, these eight average directions are used to form an S_2-like equation that serves as the low-order equation in a nonlinear acceleration scheme. The acronym NL-S_2 will be used to denote this non-linear S_2-like equation. This method is investigated for use accelerating k-eigenvalue calculations and in this case, a k-eigenvalue can be converged on the low order system. NL-S_2 is simple to discretize consistently with the S_N equation and when this is done the scalar flux solution for the NL-S_2 equation is the same as that for the S_N equation.

A primary motivation for this investigation of NL-S_2 acceleration is that an S_N style sweeper might be effective for inverting the NL-S_2 "streaming plus collision" operator. However, the NL-S_2 system, while looking similar to an S_2 equation, has some significant differences. For any mesh other than one consisting entirely of rectangles or rectangular cuboids, the NL-S_2 system will have many cyclic dependencies coupling cells. The NL-S_2 method has been investigated in a number of other works, however all previous investigations focused either on one-dimensional problems or two-dimensional problems using a structured mesh. In this work, several methods for using an S_N style sweeper were investigated for the NL-S_2 system. It is found that modifications can be made to the NL-S_2 linear system that drastically reduce the amount of off-diagonal matrix coefficients. The modified NL-S_2 system is equivalent to the original at convergence of the scalar flux solution. An S_N style sweeper is shown to be effective for this modified NL-S_2 streaming plus collision operator. Acceleration of k-eigenvalue calculations is investigated for the well known two-dimensional C5G7 benchmark as well as a C5G7 like three-dimensional problem. A pincell problem containing a large void in the center is also investigated and NL-S_2 acceleration is found

to not be significantly impacted by the void. Our results indicate that NL-S_2 acceleration is an effective alternative to traditional diffusion-based methods.

ACKNOWLEDGMENTS

I would like to thank my graduate committee for taking time to provide feedback and particularly my advisor for many hours spent cumulatively in research meetings. I would also like to thank Jan Vermaak for a significant amount of guidance on using and modifying the Chi-Tech library and also Zach Hardy whose initial work on a k-eigenvalue solver was helpful in creating a new k-eigenvalue solver using NL-S_2.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of advisor Dr. Jim Morel and committee members Dr. Jean Ragusa and Dr. Marvin Adams all of the Department of Nuclear Engineering and Dr. Matthias Maier of the Department of Mathematics.

All work conducted for the thesis was completed by the student independently.

Funding Sources

This material is based upon work supported by INL Contract Number 214446. This research made use of the resources of the High Performance Computer Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND BACKGROUND

The topic of this dissertation is a non-linear acceleration method applicable to solving the S_N transport equation. The S_N transport equation is introduced in the next section. Acceleration is reviewed in general in Section 1.2. This review is not exhaustive, but meant simply to provide some basis for describing the benefits and drawbacks of the different methods and why the specific method investigated in this dissertation may be worth pursuing. The acceleration method investigated in this dissertation is a simple "weighted flux" method where angular integrals of the angular flux are performed over parts of the unit sphere. The specific name NL-S_2 is used for the method throughout this work. The NL-S_2 method has been investigated in a number of other works, however all previous investigations focused either on one-dimensional problems or two-dimensional problems using a structured mesh. The previous investigations are reviewed in Section 1.3.2.1.

1.1 Introduction to the S_N Transport Equation

The linear Boltzmann equation is a useful description for the transport of particles that do not interact with each other. For example, it is useful for describing neutrons traveling through a medium such as a reactor or a radiation shield. In this situation, it is possible that any one neutron may collide with another neutron, but this type of interaction is so unlikely to occur compared to the neutron either traveling out of the reactor or shield, or colliding with some atom in the reactor or shield, that neutron neutron scattering can be ignored. Additionally, the approximation is made that the neutron can be accurately modeled as a particle. This approximation is useful for many applications since the typical energy ranges of interest correspond to a small enough wave length that the particle approximation is appropriate. With these approximations made, the transport equation can be derived by constructing a conservation equation in phase space where the phase space is the number of neutrons traveling in a particular direction at a particular speed [1]. The derivation is not discussed further here, instead the integro-differential form of the transport equation is simply stated below.

The mono-energetic transport equation with isotropic cross-sections and a fixed source is shown below where $\mathcal{H}$ is the problem domain and $\partial\mathcal{H}^-$ is the portion of the problem domain such that $\vec{n}(\vec{r}) \cdot \vec{\Omega} < 0$ where $\vec{n}(\vec{r})$ is the outward surface normal for the surface of the domain. $\vec{\Omega}$ is the unit direction vector, $\psi(\vec{r}, \vec{\Omega})$ is the angular flux, $\sigma_t(\vec{r})$ is the total interaction cross section, $\sigma_s(\vec{r})$ is the scattering cross section, $q(\vec{r})$ is a volumetric source, and $g(\vec{r}, \vec{\Omega})$ is a specified flux value at the inflow portion of the boundary. The convention used here is that $\int_{4\pi} d\Omega = 4\pi$. Note the case of energy dependence and also anisotropic scattering will be discussed later.

$$\begin{aligned} \vec{\Omega} \cdot \nabla\psi(\vec{r}, \vec{\Omega}) + \sigma_t(\vec{r})\psi(\vec{r}, \vec{\Omega}) &= \frac{\sigma_s(\vec{r})}{4\pi}\phi(\vec{r}) + \frac{q(\vec{r})}{4\pi} \quad \text{in } \mathcal{H} \notin \partial\mathcal{H}^- \\ \psi(\vec{r}, \vec{\Omega}) &= g(\vec{r}, \vec{\Omega}) \qquad\qquad \text{on } \partial\mathcal{H}^- \\ \phi(\vec{r}) &= \int_{4\pi} \psi(\vec{r}, \vec{\Omega}) d\Omega \end{aligned} \tag{1.1}$$

There are several methods to create an equation discrete in direction which can be solved numerically to approximate the solution of Eq. 1.1. The S_N or discrete ordinates approximation will be discussed. An angular quadrature rule is selected and the scalar flux is defined as shown in Eq. 1.2. The S_N equations are formed by replacing the continuous angular flux in Eq. 1.1 with the value at the discrete directions which leads to Eq. 1.3 The operator $\mathcal{L}_m$ introduced in Eq. 1.3 will be referred to as the "streaming plus collision" operator. Additionally, $\psi(\vec{r}, \vec{\Omega}_m)$ will subsequently be written simply as ψ_m.

$$\begin{aligned} \phi &= \sum_{m=1}^{M} w_m \psi_m \\ \psi_m &= \psi(\vec{\Omega}_m) \end{aligned} \tag{1.2}$$

$$\begin{aligned} \mathcal{L}_m \, \psi_m &= \frac{\sigma_s}{4\pi}\phi + \frac{q}{4\pi} \\ \mathcal{L}_m &= \vec{\Omega}_m \cdot \nabla + \sigma_t \end{aligned} \tag{1.3}$$

1.2 Solving the S_N Transport Equation Iteratively

In practice, solving Eq. 1.3 is difficult due to the coupling of the equations through the scattering source written on the rhs of the equation which depends on the scalar flux. The linear system to be solved for any particular spatial discretization of Eq. 1.3 grows as more directions are used in the angular quadrature set and so the linear system can be very large for many practical problems. There are many techniques for solving large linear systems iteratively with computers and many of these methods can be implemented in parallel on a large number of processors. However, the character of the S_N transport equation makes finding a generally effective linear solver difficult. When there is no scattering or the amount of scattering is low, the equation has a hyperbolic character and becomes like many decoupled advection-reaction equations and when in addition to this, as the total cross section becomes small, the equations become purely advective. As the scattering cross section becomes large, the importance of the advective character of the equation decreases and the equation instead becomes diffusive.

Many practical problems will consist of heterogeneous materials and the system may contain some regions where streaming of neutrons is important, and other regions where the scattering source dominates any streaming. As an example, multigrid methods are commonly used for a variety of problem types, have very good parallel scalability, and research effort has been made towards applying these techniques to the transport equation. For example, in [2] a multigrid method was developed and applied to two dimensional problems with uniform meshes using the corner balance discretization. The method was shown to be effective for certain problems, but showed degraded effectiveness for heterogeneous problems. The difficulties in finding a generally effective solver for Eq. 1.3 in addition to the potentially very large size of the linear system mean that some additional technique is generally required beyond simply trying to solve the equation as written with some iterative linear solver. Two such techniques will be discussed next.

The technique referred to as source iteration iteration will be discussed first. Source iteration is a technique commonly used to break the dependency between the right and left hand side of Eq. 1.3. This technique is shown in Eq. 1.4 for the fixed source problem where n is an iteration index. The

scattering source is simply lagged and this technique is actually just Richardson iteration. To update the scalar flux iterate from ϕ^n to ϕ^{n+1}, M independent equations of the form shown in Eq. 1.4 are solved. Source iteration is also relevant when solving the k-eigenvalue equation as will be discussed later. The M independent equations that are solved to compute ϕ^{n+1} from ϕ^n are simply advection equations where the constant $\vec{\Omega}_m$ takes the place of a flow velocity vector and ψ_m takes the place of a quantity that is advected in the flow field. There are many different effective and scalable solvers for these types of equations. A method referred to as sweeping will be used in this dissertation for solving Eq. 1.4 as discussed in Section 2.2. This same type of solver is investigated for use solving the NL-S_2 equations as discussed in Section 2.3.1.

$$\begin{aligned} \mathcal{L}_m \psi_m^{n+1} &= \frac{\sigma_s}{4\pi}\phi^n + \frac{q}{4\pi} \\ \phi^{n+1} &= \sum_{m=1}^{M} w_m \psi_m^{n+1} \end{aligned} \tag{1.4}$$

The number of iterations required to converge source iteration increases as the ratio $c = \frac{\sigma_s}{\sigma_t}$, called the scattering ratio, increases. To see the reason for this, note that the scalar flux solution iterate ϕ^n can be thought of as the solution considering all particles which have scattered up to n times and so it is clear that in an infinite medium where particles cannot be lost due to leakage, as $c \to 1$ the number of iterations to converge source iteration goes to infinity as particles can scatter infinitely many times. Techniques meant to address this issue as well as generally accelerate convergence are discussed in the next section.

More sophisticated techniques for solving Eq. 1.3 based on Krylov subspace methods are frequently used instead of source iteration. Such methods involve finding an approximate solution $\tilde{x}$ of $Ax = b$ within the n dimensional space $span\{b, Ab, A^2b, ..., A^{n-1}b\}$ where n is generally much less than the dimension of the matrix, otherwise storing too many vectors requires too much computer memory for the method to be practical. GMRES is one such method [3]. There are several ways GMRES could be applied to the transport equation, but only one method, which is common today, will be discussed here. The method described here was first discussed in [4] and further

analyzed in [5]. For the purpose of introducing the specific GMRES method, assume that some spatial discretization has been applied. Spatial discretization of Eq. 1.3 is discussed in detail in the next chapter. Consider $L = \{L_0, L_1, ..., L_m\}$ which is a row vector of the operators L_m which are the spatially discrete versions of the operator $\mathcal{L}_m$ shown in Eq. 1.3. Also consider an operator P which operates on the angular flux vector of unknowns and integrates the vector to produce the scalar flux and S being the operator which acts on the scalar flux and outputs the scattering source. Then the spatially discrete S_N equation can be written as $(L - SP)\Psi = q_v$ where Ψ is by

$$\Psi = \begin{bmatrix} [\psi_1] \\ [\psi_2] \\ \vdots \\ [\psi_M] \end{bmatrix}$$

with $[\psi^m]$ being the column unknowns for angular flux $\psi(\vec{\Omega}_m)$. For simplicity, q_v is written as a source vector for angular flux instead of an isotropic source. Now applying PL^{-1} to the S_N transport equation results in

$$(I - SPL^{-1})\phi = PL^{-1}q_v \tag{1.5}$$

Because only the action of matrices are required as part of GMRES, L^{-1} does not need to be constructed, instead only the action of L^{-1} is needed. An efficient matrix free way of computing L^{-1} is discussed in Section 2.2. GMRES can be thought of as an acceleration method as discussed in [5] where each scalar flux iterate is computed by considering information from all previous scalar flux iterates compared with simple source iteration where the next scalar flux iterate is based only on the previous iterate. In general, preconditioning is still desirable and used to accelerate convergence of GMRES. An additional benefit to GMRES is that some preconditioning methods that become ineffective when used with source iteration remain effective when used as part of a krylov method as discussed in Section 1.3.1. GMRES will also be used to solve the NL-S_2 equation as discussed in Section 2.4.

To make the discussion in the section more complete, it is noted that there are other forms for the transport equation besides that shown in Eq. 1.1 which are more commonly solved using algorithms such as multigrid methods. These forms involve second order derivatives, see for example [6] for the so called self-adjoint angular flux equation. Eq. 1.1 is called the first order transport equation and the primary reason it is the subject of this dissertation is that there are spatial discretizations for which the first order form of the transport equation can be solved using matrix free algorithms. One such specific type of discretization which will be used throughout this dissertation is described in the next chapter. For more information on recent work related to the development scalable efficient solvers for the second order form of the transport equation see [7].

1.3 Accelerating Convergence of the Scattering Source

There are many linear and non-linear techniques to increase the effectiveness of source iteration. A good review is given by Adams and Larsen [8]. Essentially all of these techniques involve projecting the original problem into a subspace either with dependence on fewer directions or no angular dependence. This is referred to as the low-order problem. The linear acceleration methods involve calculating a correction which is an estimate for the scalar flux error after one or more source iterations and then adding this correction to the scalar flux. At convergence and assuming compatible discretization between the transport solve and the low order solve, the correction calculated is zero and the resulting scalar flux solution is the same as that calculated by transport.

With non-linear acceleration methods, a scalar flux is calculated from the low order problem. The low order equation contains terms which depend non-linearly on the transport equation. For some non-linear acceleration methods, the scalar flux solution from the low order system will be different from the accelerated S_N transport solution and these may both be different from the unaccelerated S_N transport solution. However, for the fully consistent NL-S_2 acceleration investigated here, the low order scalar flux solution will be the same as the high-order scalar flux solution and these are both the same as the unaccelerated solution. The NL-S_2 acceleration method is discussed further in Section 1.3.2.1. Although the subject of this dissertation is a non-linear method, popular linear and non-linear methods are reviewed next for completeness.

1.3.1 Linear Acceleration Methods

Two linear techniques will be reviewed briefly in this section. The first method discussed Diffusion Synthetic Acceleration (DSA). This method is specifically reviewed because of its popularity and effectiveness for certain problems. Transport Synthetic Acceleration (TSA) is discussed next primarily because this method has some of the same potential benefits that NL-S_2 has, specially it is simple to discretize the low order equation consistently with the S_N system and also the same linear solver that is used to invert the S_N streaming plus collision operator can be used for the low order system.

An important fact relevant to the development of DSA is that when the angular flux is linearly anisotropic, that is of the form $\psi(\vec{r},\vec{\Omega}) = a(\vec{r}) + \vec{b}(\vec{r})\cdot\vec{\Omega}$, the scalar flux solution of the transport equation is given by Eq. 1.6 which is clearly a diffusion equation.

$$\begin{aligned} \nabla\cdot\vec{J}(\vec{r}) + \sigma_a(\vec{r})\phi(\vec{r}) &= q_v(\vec{r}) \\ \vec{J}(\vec{r}) &= -\frac{1}{3\sigma_t}\nabla\phi \end{aligned} \tag{1.6}$$

As for what this means physically, it can be shown through asymptotic analysis that the solution to the transport equation is approximately the solution to the diffusion equation when the total cross section and the scattering ratio (σ_s/σ_t) are large [9]. The importance of this is that these are the conditions under which source iteration will converge slowly. There are many techniques for solving a diffusion equation and these can be employed to speed up convergence of the scalar flux using DSA. The historical development of DSA is covered in the review by Adams and Larsen [8] and several implementations are possible. One specific implementation is reviewed here and some items relevant to the overall performance of DSA are briefly discussed.

For simplicity, consider the linear system $Ax = b$. The residual equation $Ae = r$ where e is the error and $r = b - A\tilde{x}$ is the residual where $\tilde{x}$ is some guess for the solution vector x. Starting with the guess $\tilde{x}$, the residual equation can be solved for the error to give $x = \tilde{x} + e$. Of course solving the residual equation is as difficult as solving the original linear system, however, if some

other linear system existed which was close to A but easier to invert, then an approximate residual equation could be solved to get an approximate error and this could be used to update a solution iterate like $x^{n+1} = x^n + \tilde{e}$ where $B\tilde{e} = r$ with B being close to A in some sense but easier to invert. The idea can be applied towards accelerating convergence of source iteration. Starting with a scalar flux iterate ϕ^n, a new scalar flux iterate can be computed $\phi^{n+1/2}$. The residual in this case is simply $r^{n+1/2} = \phi^{n+1/2} - \phi^n$. An approximate residual equation is solved using the diffusion approximation $D\tilde{e}^{n+1/2} = r^{n+1/2}$ where D is the diffusion operator. The next scalar flux iterate is then computed $\phi^{n+1} = \tilde{e}^{n+1/2} + \phi^{n+1/2}$.

An important aspect of the historical development of DSA reviewed by Adams and Larsen [8] was the realization that consistent spatial discretization between the S_N system and diffusion system was important for the overall stability and effectiveness of the acceleration scheme. To summarize why this is notable in one sentence, this fact adds some level of complexity to developing DSA because the discontinuous spatial discretizations commonly used for the first order S_N equation are often not easily applied to diffusion equations. Adams and Larsen [8] review the historical development of consistent discretizations and also nearly consistent discretizations, the motivation for the later being that a DSA spatially discretized in a manner close to consistent may be only marginally less effective than the fully consistent scheme in terms of spectral radius, but may be much easier to solve numerically. The first work to derive a fully consistent DSA scheme for discontinuous discretization on a general three-dimensional mesh of tetrahedrals [10] resulted in a low order system having 16 unknowns on each tetrahedral and was relatively complex to solve. In that work, they solved a real world problem and compared the fully consistent method derived with some partially consistent methods and found that one of the partially consistent methods became unstable while a different partially consistent method required more iterations overall to converge, but was easier to solve and thus faster overall for the specific problem investigated. The use of discontinuous finite elements for elliptic problems has been investigated in various works and more recently these ideas were applied towards developing a partially consistent DSA method which involves solving a symmetric positive definite linear system and was shown be be uncondi-

tionally stable for triangular meshes [11] and this was extended to meshes of arbitrary polyhedrons in [12].

When voids are present, that is $\sigma_t = 0$, a problem with Eq. 1.6 becomes apparent, namely the diffusion coefficient becomes large as $\sigma_t \to 0$ and is undefined for a void. A linear DSA scheme compatible with voids that involves a modified diffusion coefficient is presented in [13] and the references therein include past work to develop methods with modified diffusion coefficients in or near void regions. In [14] the approach is taken to only apply DSA to optically thick regions and thus this method also does not suffer issues related to an ill defined diffusion coefficient in void regions. More generally, it has been documented that DSA can become divergent for strongly heterogeneous problems in multiple dimensions when using DSA with source iteration. However, this problems can be largely mitigated by using DSA as a preconditioner for krylov iteration instead of source iteration as described in [15].

The purpose of the previous discussions are to highlight that while DSA is effective for accelerating convergence in many situations, there are some complexities related to consistent discretization and handling voids. As discussed in the next chapter, consistently discretizing the NL-S_2 equations is simple. Additionally, voids do not cause any numerical difficulty, however, if strongly heterogeneous material configurations couple with thick cells leads to negative angular fluxes, then some treatment is required otherwise then NL-S_2 systems may be ill posed. This is discussed in Section 2.5. Additionally, it is shown in Section 3.1.2 that NL-S_2 acceleration can suffer convergence issues for strongly heterogeneous problems although a simple solution is shown in that section.

The previous discussion in this section regarding the residual equation is also applicable to Synthetic Transport Acceleration (TSA). Instead of approximately solving the residual equation using a diffusion approximation, the first order S_N equation can be used but with fewer directions in the angular quadrature set than is used when computing the residual. This method is specially discussed in this work because TSA has several potential benefits in common with NL-S_2. First the same matrix free linear solver discussed in Section 2.2 which is commonly used for the S_N

equation can be used both for TSA and potentially for NL-S_2 acceleration. Also, as discussed previously, it can be difficult to implement a spatial discretization when using DSA that is consistent with the spatial discretization used for the S_N equation. However, it is trivial to consistently discretize the TSA equation since it is an S_N equation and, as is discussed in Section 2.3, consistently discretizing the NL-S_2 system is also trivial. TSA is discussed in [16] where conjugate gradient is employed to solve the lower order S_N equation.

1.3.2 Non-linear Acceleration Methods

Non-linear methods of accelerating the convergence of iterative solutions to the S_N have in common with linear methods that they rely on an "low order" equation with dependence on fewer directions that the S_N equation or no angular dependence. These methods include in the low order system a term with a non-linear dependence on the angular flux S_N equation and this non-linear term typically encompasses information about the shape of the scalar flux solution which could otherwise not be included in the low order equation. Whereas linear methods generally require that the low order system be consistently or close to consistently discretized with the S_N spatial discretization, it is simpler to use a low order system that is not consistently discretized when using nonlinear methods. The scalar flux is computed from the low order system, and so, for example, one might want to use a more accurate spatial discretization for the low order system and use a less accurate discretization for the S_N system since it is only used as input to the non-linear terms in the low order equation. In this work, the low order system investigated is discretized consistently with the S_N equation.

Some fundamental ideas underlying the nonlinear methods relevant to this dissertation are presented by Anistratov and Gol'din [17]. The ideas presented in that work are relevant to this dissertation because the method investigated here can be thought of as belonging to a family of non-linear acceleration methods that involve a low order equation where the nonlinear terms depending on the transport equation are developed by taking moments of the transport equation. What is referred to as NL-S_2 acceleration in this dissertation involves taking the zeroth moments of the transport equation over different parts of the angular domain. NL-S_2 is discussed in more detail in

the next section. This method was first developed for acceleration of a time dependent transport calculation in one-dimensional spherical coordinates [18] and also in [19] where a more general form of the method was used for steady state transport in slab geometry.

NL-S_2 and closely related methods will be discussed specifically in the next section. Before that discussion, non-linear diffusion like methods are summarized to compare them with the DSA discussion in the previous section. Quasi-diffusion is most closely related to the method discussed in the next section. Another method commonly referred to as non-linear diffusion acceleration (NDA) can be summarized as a method using a drift-diffusion equation where a drift vector has been added so that the scalar flux from the low order NDA equation is the same as the S_N solution. A review of NDA development as well as a formulation for discontinuous discretization is included in [20]. Again, these diffusion based methods require some special treatment for problems with voids. For NDA, a method similar to that mentioned previously for linear DSA in [13] was presented in [21].

1.3.2.1 NL-S_2 Acceleration

The NL-S_2 method investigated is derived by first partitioning the unit sphere into octants. $\int_{\Omega_k} d\Omega$ is the integral over the k^{th} octant and the partial range scalar flux will be written using Φ instead of ϕ to avoid confusion with scalar flux. The partial range scalar flux over octant k is defined as $\Phi_{\Omega_k}(\vec{r}) = \int_{\Omega_k} \psi(\vec{\Omega}, \vec{r}) d\Omega$. The general scheme for NL-S_2 acceleration of source iteration is shown in Eq 1.7 where, for simplicity, the equations below are written with continuous angular dependence where $\mathcal{L} = \vec{\Omega} \cdot \nabla + \sigma_t$. In the scheme shown, one transport solve is performed and then ϕ^{n+1} is computed from the NL-S_2 equation.

$$
\begin{aligned}
\mathcal{L}\psi^{n+1/2} &= \frac{q}{4\pi} + \frac{\sigma_s}{4\pi}\phi^n \\
\vec{M}_{\Omega_k}^{n+1/2} &= \frac{\int_{\Omega_k} \vec{\Omega}\psi^{n+1/2} d\Omega}{\int_{\Omega_k} \psi^{n+1/2} d\Omega} \\
(\nabla \cdot \vec{M}_{\Omega_k}^{n+1/2} + \sigma_t)\Phi_{\Omega_k}^{n+1} &= \frac{2}{\pi}q + \frac{2}{\pi}\sigma_s\phi^{n+1} \\
\phi^{n+1} &= \sum_k^8 \Phi_{\Omega_k}^{n+1}
\end{aligned}
\tag{1.7}
$$

Solving for ϕ^{n+1} obviously involves solving for the eight partial range scalar flux values. As written in Eq. 1.7, the lhs and rhs of the NL-S_2 equation are coupled. Solving the coupled equation will generally be difficult for multidimensional problems. The difficulty is not as severe as for the S_N equation since there are only eight average directions instead of however many directions are in the S_N quadrature set (many more than eight in general), but to make the computation of ϕ^{n+1} practical, source iteration must be used on the NL-S_2 equation. As the scattering ratio increases close to one, source iteration on the lower order system will converge slowly. However, a lower order iteration can still be much faster than an S_N iteration and thus even if the total number of iterations required to converge to a scalar flux is not reduced, the calculation may still be accelerated if the bulk of the iterations can be shifted to the low order system.

Both source iteration and GMRES are investigated for solving Eq. 1.7. Source iteration is discussed in this section for simplicity. Solving the NL-S_2 equation is discussed in detail in sections Section 2.3.1. The term "streaming plus collision operator" will be used again to refer to the left hand side of the NL-S_2 source iteration equation shown in Eq. 1.8. Whether the term "streaming plus collision" operator is meant to refer to the S_N equations or the NL-S_2 equations, that is $\vec{\Omega} \cdot \nabla + \sigma_t$ or $\nabla \cdot \vec{M}_{\Omega_k} + \sigma_t$, will be specified if there is any ambiguity. Eq. 1.8 is obviously similar to the S_N source iteration equation. Several different options exist for picking an initial guess for the source iteration shown in Eq. 1.8. One obvious choice would be to use $\phi^{n+1/2}$ from the initial S_N iteration shown in Eq. 1.7. This generally is a good choice for an initial guess. One potential complication involves the situation where negative angular fluxes result from the S_N solve which is discussed in detail in Section 2.5.

$$
\begin{aligned}
(\nabla \cdot \vec{M}_{\Omega_k}^{n+1/2} + \sigma_t)\Phi_{\Omega_k}^{j+1} &= \frac{2}{\pi} q + \frac{2}{\pi}\sigma_s \phi^j \\
\phi^{j+1} &= \sum_k^8 \Phi_{\Omega_k}^{j+1}
\end{aligned}
\tag{1.8}
$$

The introduction in this section thus far has not involved spatial discretization. In the next chapter, the spatial discretization used in the S_N equation will be introduced. Following this, the specific NL-S_2 formulation investigated is also introduced in the next chapter. The primary idea of the formulation is summarized here. The coefficients for the spatially discretized S_N equations are simply averaged and these averages become the coefficients for the NL-S_2 equations. Thus, there is no need to discuss a spatial discretization of Eq. 1.7 since it follows directly from the formulation. A very similar methodology was used by Adams [22] where in addition to taking moments over direction, a spatial moment was used and the resulting non-linear S_2 like equation had unknowns only on cell edges. Although the method presented in [22] was easily generalized to multiple dimensions, multidimensional problems and unstructured meshes were not investigated. The NL-S_2 formulation used in this work only uses moments in angle and so as shown in Section 2.3, average coefficients must be stored for all volumetric terms as well as face terms. This formulation is in some ways simpler. But the primary novelty of the work presented in this dissertation is an investigation into the effectiveness of using an S_N style sweeper for solving the NL-S_2 equation for complicated two-dimensional and three dimensional unstructured meshes and the simple technique discussed in Section 2.3.2 to increase the efficiency of the sweep solver.

Eq. 1.7 actually shows a specific example of a more general method where the average direction $\vec{M}^{k,n+1/2}$ could be computed using some weight function. As shown in the equation, this weight function is simply one. Use of different weight factors are discussed in [23] where slab geometry is investigated and a weight of $|\mu|^\alpha$ is investigated with $\alpha \in [0, 1]$ and μ being the direction cosine and this idea was expanded upon in [24] with a weight factor of $1 + \beta|\mu|^\alpha$. The idea of generally weighted moment methods was investigated further for two-dimensional problems using a uniform mesh in [25]. In this work, the low order equation was discretized independently of the

transport equation using a lumped bilinear discontinuous finite element discretization while the needed transport equation input was approximated using the method of short characteristics with parabolic interpolation.

1.4 k-Eigenvalue Calculations

The mono-energetic k-eigenvalue equation with isotropic cross-sections is shown in Eq. 1.9. Power iteration can be used to calculate the k-eigenvalue. Given iteration values for ϕ^m and k^m, power iteration can be summarized by Eq. 1.10. To solve for ψ^{m+1} in the first part of this equation, source iteration would generally be used. Each power iteration requires solving the transport equation to obtain an updated scalar flux iterate.

$$\mathcal{L}\psi = \left(\frac{1}{k}\frac{1}{4\pi}\nu\sigma_f + \frac{\sigma_s}{4\pi}\right)\phi \tag{1.9}$$

$$\begin{aligned} \mathcal{L}\psi^{n+1} - \frac{\sigma_s}{4\pi}\phi^{n+1} &= \frac{1}{4\pi}\frac{\nu\sigma_f}{k^n}\phi^n \\ k^{n+1} &= \frac{\int_V \phi^{n+1}dV}{\int_V \phi^n dV}k^n \end{aligned} \tag{1.10}$$

Instead of attaining a scalar flux solution iterate from the S_N equations, the scalar flux and k-eigenvalue can be converged on the low order system as shown in Eq. 1.11. Note that in this equation, the iteration indices $n+1$ are written on the scalar flux and the k-eigenvalue meaning that a k-eigenvalue iterate is solved for using the low order equation.

$$\begin{aligned} \mathcal{L}\psi^{n+1/2} &= \left(\frac{1}{4\pi}\frac{\nu\sigma_f}{k^n} + \frac{\sigma_s}{4\pi}\right)\phi^n \\ \nabla\cdot\left(\vec{M}_{\Omega_k}^{n+1/2} + \sigma_t\right)\Phi_{\Omega_k}^{n+1} &= \left(\frac{1}{4\pi}\frac{\nu\sigma_f}{k^{n+1}} + \frac{\sigma_s}{4\pi}\right)\phi^{n+1} \end{aligned} \tag{1.11}$$

The low order system can be solved with power iterations. Source iteration would also generally be employed as an inner iteration on the streaming plus collision operator when solving the low order system. The scheme shown in Eq. 1.11 is applicable to solving energy dependent transport eigenvalue problems. In that case, source iteration can be used as a inner iteration to converge

the within group low order equation while lagging k and the scattering source from other groups.

1.4.1 Energy Dependence

The energy dependent transport equation is of interest to this work as the k-eigenvalue problems investigated in chapter 5 will use the multigroup transport equation. The energy dependent transport equation is shown below in the fixed source form for simplicity.

$$\begin{aligned} \vec{\Omega} \cdot \nabla \psi(\vec{r}, \vec{\Omega}, E) + \sigma_t(\vec{r}, E)\psi(\vec{r}, \vec{\Omega}, E) &= \frac{1}{4\pi} \int_0^\infty \sigma_s(\vec{r}, E' \rightarrow E)\phi(\vec{r}, E')dE' + \frac{q(\vec{r}, E)}{4\pi} \\ \phi(\vec{r}, E) &= \int_{4\pi} \psi(\vec{r}, \vec{\Omega}, E)d\vec{\Omega} \end{aligned} \tag{1.12}$$

A common technique for discretizing the energy dependence of this equation is the multigroup treatment where the energy domain of interest for the calculation is divided into G intervals. A common scheme used is to number the intervals starting from 0 at the highest energy. For any energy interval E_g to E_{g-1}

$$\begin{aligned} \psi_g(\vec{r}, \vec{\Omega}) &= \int_{E_g}^{E_{g-1}} \psi(\vec{r}, \vec{\Omega}, E)dE \\ \phi_g(\vec{r}) &= \int_{E_g}^{E_{g-1}} \phi(\vec{r}, E)dE \end{aligned}$$

The multigroup cross sections are defined through use of a weighting spectrum $f(E)$, which ideally is equal to the energy dependent scalar flux or angular flux

$$\sigma_{x,g}(\vec{r}) = \frac{\int_{E_g}^{E_{g-1}} \sigma_x(\vec{r}, E)f(E)dE}{\int_{E_g}^{E_{g-1}} f(E)dE}$$

The multigroup equation for group g is

$$\mathcal{L}_{m,g}\psi_{m,g}(\vec{r}) = \sum_{g'=1}^{G} \frac{1}{4\pi}\sigma_{s,g'\to g}\phi_{g'}(\vec{r}) + \frac{q_g(\vec{r})}{4\pi}$$

The scattering source now couples all directions and energy groups. Source iteration is relevant to this energy dependent case

$$\mathcal{L}_{m,g}\psi_{m,g}^{n+1}(\vec{r}) = \frac{\sigma_{s,g}(\vec{r})}{4\pi}\phi_g^n(\vec{r}) + q_{s,g}(\vec{r}) + \frac{q_g(\vec{r})}{4\pi} \tag{1.13}$$

where $q_{s,g}(\vec{r})$ is a source from scattering of particles from any energy g' into group g. If $q_{s,g}(\vec{r})$ is also lagged, then Eq. 1.13 is analogous to the monoenergetic transport equation and can be solved for ϕ_g^{n+1} as shown in Eq. 1.4. After converging the inner within group iteration, the value of $q_{s,g}(\vec{r})$ for each g can be calculated and this process repeated until convergence.

1.5 Anisotropic Scattering

A more general anisotropic scattering source includes a scattering cross section with a dependence on direction. Consider the monoenergetic case, then a general scattering source for $\psi(\vec{\Omega})$ can be written as

$$q_{general}(\vec{\Omega}) = \int_{4\pi} \psi(\vec{\Omega}')\sigma_s\left(\vec{\Omega}' \to \vec{\Omega}\right) d\Omega'$$

However, the scattering cross section in practice depends only on $\vec{\Omega}' \cdot \vec{\Omega}$ and this feature allows for the anisotropic scattering cross section to be represented by an expansion of spherical harmonics as follows where Y_ℓ^m is the spherical-harmonic function of degree ℓ and order m

$$q_{general}(\vec{\Omega}) = \sum_{\ell=0}^{\infty}\sum_{m=-\ell}^{+\ell} \frac{2\ell+1}{4\pi}\sigma_\ell\phi_\ell^m Y_\ell^m(\vec{\Omega})$$

$$\phi_\ell^m = \int_{4\pi} \psi(\vec{\Omega})Y_\ell^m(\vec{\Omega})d\Omega$$

and σ_ℓ is shown below where $\xi_s \equiv \vec{\Omega}' \rightarrow \vec{\Omega}$ in the scattering frame. ξ_s is related to the cosine with the z-axis, ξ, in the lab frame as well as the lab frame azimuthal angle, ω, by the following $\xi_s = \xi\xi' + \sqrt{(1-\xi^2)(1-\xi'^2)}\cos(\omega' - \omega)$

$$\sigma_\ell = 2\pi \int_{-1}^{+1} \sigma_s(\xi_s) P_\ell^0(\xi_s)$$

Problems with anisotropic scattering will be investigated as part of the proposed work. NL-S_2 acceleration for these problems will be implemented by only accelerating the scalar flux. Higher moments are simply lagged and treated as an additional source when performing source iteration to converge a NL-S_2 solution. This is shown below where $\phi_\ell^{m,n}$ is the scalar flux moment m or order ℓ at iteration index n. Results exploring the effectiveness of this are presented in Section 4.3. In general, this will become less effective as the anisotropic scattering becomes more significant.

$$\begin{aligned}
\mathcal{L}\psi^{n+1/2} &= \frac{q}{4\pi} + \sum_{\ell=0}^{L} \sum_{m=-\ell}^{+\ell} \frac{2\ell+1}{4\pi} \sigma_\ell \phi_\ell^{m,n} Y_\ell^m(\vec{\Omega}) \\
\vec{M}^{k,n+1/2} &= \frac{\int_{\Omega^k} \vec{\Omega}\psi^{n+1/2} d\Omega}{\int_{\Omega^k} \psi^{n+1/2} d\Omega} \\
\nabla \cdot \left(\vec{M}^{k,n+1/2}\phi^{k,n+1}\right) + \sigma_t \phi^{k,m+1} &= \frac{2}{\pi} q + \\
&\frac{2}{\pi}\sigma_s \phi^{n+1} + \sum_{\ell=1}^{L} \sum_{m=-\ell}^{+\ell} \frac{2\ell+1}{4\pi} \sigma_\ell \phi_\ell^{m,n+1/2} Y_\ell^m(\vec{\Omega})
\end{aligned} \tag{1.14}$$

2. NL-S_2 FOR DG DISCRETIZED S_N TRANSPORT

To this point, the equations presented have been continuous in space. This chapter introduces the spatially discrete S_N equation. The spatial discretization of the S_N equation is important since the NL-S_2 equation investigated in this work will be consistently discretized. The spatially discrete NL-S_2 equation will be introduced after the S_N equation.

This chapter also discusses solving the spatially discretized equations. The idea of a sweep solver is introduced and its use for the S_N equation is discussed. Its potential use for the NL-S_2 equation is discussed as well. A method is introduced in this chapter to modify the NL-S_2 system to make the sweep solver more effective. Since this section introduces the consistently discretized NL-S_2 equations, this chapter also includes a discussion of a situation where the consistently discretized NL-S_2 equations may not be solvable, that is when negative angular flux solutions are present in which case an inconsistent NL-S_2 system is used.

2.1 DG Discretization

The upwind discontinuous Galerkin (DG) finite element method (FEM) will be used for spatial discretization in this work. This method was introduced for the purpose of solving the S_N transport equation in [26] and initially analyzed in [27]. This first analysis arrived at an error estimate of h^k on general meshes and h^{k+1} for structured meshes although it was noted that the numerical results showed better convergence for general meshes than that arrived at from the analysis. The error estimate was improved further for $h^{k+1/2}$ in [28] for general meshes. The is a commonly used linear discretization.

The DG Discretization will be introduced in this section using the spatially discretized version of Eq. 1.4 which shows the source iteration method for solving the S_N equation. Consider that the problem domain $\mathcal{H}$ is covered by a triangulation $\mathcal{H} = \sum_h^H T_h$ where T_h is a mesh element. The term triangulation and triangles are used here and subsequently to refer to a general mesh of elements covering the domain, the mesh elements may be triangles or other shapes. Consider that

$\psi_m^{n+1}(\vec{r})$ is approximated by a function $\tilde{\psi}_m^{n+1}(\vec{r})$ in the space $V(T_h)$, such as the space of polynomials of degree less than k, which are continuous within a triangle T_h. No continuity condition is enforced between triangles. The weak form of Eq. 1.1 is formulated by multiplying the equation by a test function $v \in V(T_h)$ and then integrating over the problem domain. Integration by parts is used resulting in Eq. 2.1. The solution $\tilde{\psi}_m^{n+1}(\vec{r})$ is determined by finding the function which satisfies Eq. 2.1 for all $v(\vec{r}) \in V(T_h)$. Note that the scattering source contribution dependent on the previous iterate of the scalar flux ϕ^n as shown in Eq. 1.1 is written below in Eq. 2.1 simply as q_t^n so this term is meant to included the scattering source as well as any volumetric fixed source. Also note that the same equation applies for problems with anisotropic scattering or with a lagged fission source simply by changing q_t^n to account for these in which case the source may become a function of angle in addition to just space as written below.

$$\sum_{T_h}^{H} \Big\{ \int_h \Big[\sigma_t v(\vec{r})\tilde{\psi}_m^{n+1}(\vec{r}) - \tilde{\psi}_m^{n+1}(\vec{r})\vec{\Omega}_m \cdot \nabla v(\vec{r})\Big]\, dh + \int_{\partial h} \hat{\psi}_m^{n+1}(\vec{r})\vec{\Omega}_m \cdot \vec{n}(\vec{r}) v(\vec{r}) ds \Big\} = \int_H q_t^n(\vec{r}) v_j(\vec{r}) dh \tag{2.1}$$

The above equation is applicable in two or three spatial dimensions with $\int_h dh$ being an integral over the volume or area of the triangle in two or three dimensions and $\int_{\partial h} ds$ is an integral over the surface area or perimeter of the triangle with $\vec{n}(\vec{r})$ being the outward pointing normal. The term including $\int_H dh$ is an integral over the whole domain covered by the triangulation. The term $\int_{\partial h} \hat{\psi}_m(\vec{r})\vec{\Omega}_m \cdot \vec{n}(\vec{r}) v(\vec{r}) ds$ is not straight forward due to the discontinuity at the the border of each triangle, the treatment of this integral is described succinctly in [29] for the closely related problem of linear advection (identical in form to the S_N equation except that $\vec{\Omega}_m$ is replaced by a vector specifying the velocity of the unknown at each point). Formally applying integration by parts leads to $\hat{\psi}$ being equal to the average value of $\tilde{\psi}$ at the discontinuity, that is $\frac{1}{2}\{\tilde{\psi}_m(\vec{r}_{\partial h}+\epsilon)+\tilde{\psi}_m(\vec{r}_{\partial h}-\epsilon)\}$ where ϵ is a small positive number. This definition leads to a method which is only stable in a $L^2(H)$ norm. Additionally, using this definition for $\hat{\psi}$ means that the solution for each triangle in the mesh depends on all of its neighbors and so to solve the equation requires inverting a matrix

using some general method like Gaussian elimination. For a large mesh, Gaussian elimination is not computationally practical and this linear system would need to be solved iteratively. The "upwind" flux can be used for $\hat{\psi}$ instead which leads to a method that is L^∞ stable and also block lower triangular. This means the equation can be solved in a matrix free method, by visiting each triangle in the mesh in a specific order and inverting a small dense matrix to solve Eq. 2.2 in each triangle. This is simply inverting the block diagonals followed by forward substitution. The upwind DG discretized S_N equation is shown below

$$\begin{aligned}\int_h \left\{\sigma_t v(\vec{r})\tilde{\psi}_m^{n+1}(\vec{r}) - \tilde{\psi}_m^{n+1}(\vec{r})\vec{\Omega}_m \cdot \nabla v(\vec{r})\right\} dh + \int_{\partial h} \tilde{\psi}_m^{*,n+1}(\vec{r})\vec{\Omega}_m \cdot \vec{n} v(\vec{r}) ds \\ = \int_h q_t^n(\vec{r}) v_j(\vec{r}) dh\end{aligned} \tag{2.2}$$

where the upwind flux $\tilde{\psi}_m^*(\vec{r})$ is shown below. In this definition, consider a face belonging to cell T_1 with outward normal $\vec{n}_1(\vec{r})$ that either boarders another cell T_2 with outward normal, that is an internal face, or is part of the problem domain and thus an external face. Then the upwind unknown is defined as follows

$$\tilde{\psi}_m^*(\vec{r}) = \begin{cases} \tilde{\psi}_m^1(\vec{r}), & \text{if } \vec{\Omega} \cdot \vec{n}_1(\vec{r}) > 0 \\ \tilde{\psi}_m^2(\vec{r}), & \text{if } \vec{\Omega} \cdot \vec{n}_1(\vec{r}) < 0 \text{ and internal face} \\ g_m(\vec{r}), & \text{if } \vec{\Omega} \cdot \vec{n}_1(\vec{r}) < 0 \text{ and external face} \end{cases}$$

A second formulation equivalent to Eq. 2.2 is shown in Eq. 2.3 and is sometimes presented in literature. This second formulation can be found from applying integration by parts again to Eq. 2.2, that is use integration by parts to arrive at Eq. 2.1, apply the upwind flux stabilization to this, and then use integration by parts again to arrive at Eq. 2.3

$$\int_h \Big(\vec{\Omega}_m \cdot \nabla \tilde{\psi}_m(\vec{r}) + \sigma_t \tilde{\psi}_m(\vec{r}) - \frac{q(\vec{r})}{4\pi}\Big) v(\vec{r}) dh + \int_{\partial h^- \notin \partial H^-} \vec{n} \cdot \vec{\Omega}_m (\tilde{\psi}_m(\vec{r}) - \tilde{\psi}^*_m(\vec{r})) v(\vec{r}) ds + \int_{\partial h^- \in \partial H^-} \vec{n} \cdot \vec{\Omega}_m (\tilde{\psi}_m(\vec{r}) - g_m(\vec{r})) v(\vec{r}) ds = 0 \tag{2.3}$$

where ∂h^- is defined to be the portion of the cell boundary such that $\vec{n}(\vec{r}) \cdot \vec{\Omega}_m < 0$ where $\vec{n}(\vec{r})$ is the outward normal of the face and $\tilde{\psi}^*_m(\vec{r})$ is the upwind flux as defined earlier.

To generate an equation that can be solved practically, the standard finite element procedure is followed which is to consider only a subspace $v_h \in V(T_h)$ instead of considering all functions $v \in V(T_h)$. Additionally, these functions $v_h \in V(T_h)$ will be chosen so that they are non-zero only over individual triangles so that the linear system representing the equation to be solved will be sparse. The standard Galerkin procedure is that the solution is also approximated in this subspace, $\tilde{\psi}_m(\vec{r}) = \sum_i^I \psi_{m,i} v_i(\vec{r})$, where $\tilde{\psi}_m(\vec{r})$ is the approximate solution. $\psi_{m,i}$ are the coefficients to be solved for by enforcing Eq. 2.4 on every triangle where the upwind flux is defined in the same manner discussed above. There are many possible choices for the test and basis function space v_h. Some specific examples are discussed in the next section.

$$\int_h \Big\{\sigma_t v_i(\vec{r}) \psi^{n+1}_{m,j} u_j(\vec{r}) - \psi^{n+1}_{m,j} u_j(\vec{r}) \vec{\Omega}_m \cdot \nabla v_i(\vec{r})\Big\} dh + \int_{\partial h} \psi^{n+1}_{m,*} u_*(\vec{r}) \vec{n} \cdot \vec{\Omega}_m v_i(\vec{r}) ds = \int_h q^n_v(\vec{r}) v_j(\vec{r}) dh \quad :\forall\ v_i, u_j \tag{2.4}$$

Some specific examples of DG discretizations are introduced next to facilitate discussion of the DG NL-S_2 acceleration implemented in this dissertation. The specific example of two dimensional problems with linear functions defined on triangular mesh elements are discussed first for simplicity. The basis functions used in this dissertation are actually the so called PWLD basis functions [38] and these will be investigated on quadrilaterals and triangles in two dimensions as well as hexahedrals in three dimensions, but the ideas presented more simply for linear basis function on triangles below are generally applicable. The basis functions are defined on a reference right trian-

gle in a natural coordinate system, defined by coordinates (ζ, η), where the two legs of the triangle have length one. The linear basis functions $N_1(\zeta, \eta)$, $N_2(\zeta, \eta)$, $N_3(\zeta, \eta)$ are written below and also plotted in Figure 2.1.

$$\begin{aligned} N_1 &= 1 - \zeta - \eta \\ N_2 &= \zeta \\ N_3 &= \eta \end{aligned} \tag{2.5}$$

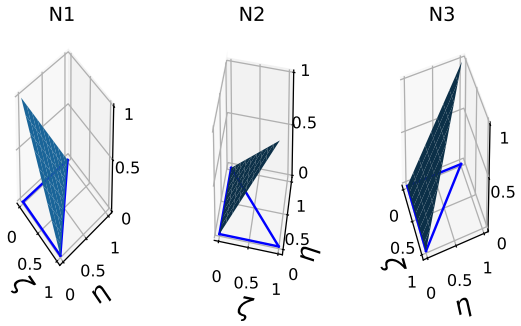


Figure 2.1: Transformation from the normal x,y coordinate system to the reference coordinate system

The triangles used to create a mesh can be of a general shape. The triangular elements can be transformed from the Cartesian coordinate system used for the mesh to a reference coordinate system as shown in Eq 2.6 so that the integrals required to build the finite element system can be written in terms of the reference coordinate system. This is illustrated in Figure 2.2

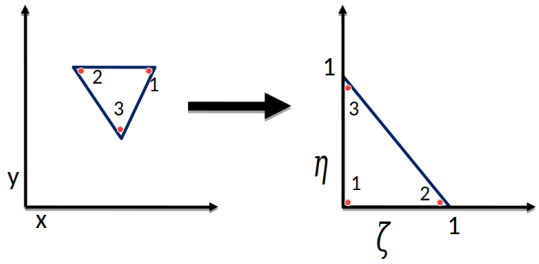


Figure 2.2: Transformation from the Cartesian x,y coordinate system to the reference coordinate system

The finite element method being described here is isoparametric meaning that the location of the unknowns corresponds with the geometric points used to describe the geometry, that is the vertices of the triangle. The nodes of each triangle in the domain are numbered in a counter-clockwise sense and then the Cartesian x, y coordinates for element e can be written in terms of the reference basis as shown in Eq. 2.6 below.

$$\begin{aligned} x &= x_1^e + c_3\zeta - c_2\eta \\ y &= y_1^e - b_3\zeta + b_2\eta \end{aligned} \tag{2.6}$$

The integrals in the weak formulation of the problem are transformed and evaluated on the reference coordinate system. The integral over the volume $\int_h dh$ is transformed to reference element $\int_h dh = \int_{\hat{h}} |J| d\hat{h}$ where $|J|$ is the determinant of the Jacobian matrix, dh is the differential volume of the original element, and $d\hat{h}$ is the differential volume of the reference element. The Jacobian matrix for the specific example being discussed here in two dimensions is given by

$$J = \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \zeta} & \frac{\partial x}{\partial \eta} \end{bmatrix} \tag{2.7}$$

The term "Jacobian" is usually used to refer to the determinant of the Jacobian matrix, written here as $|J|$. For the specific example of linear basis functions on triangles, transforming the integral of

a function over a triangular element h to the reference element becomes:

$$\int\int_h f(x,y)dxdy = \int_0^1 \int_0^{1-\eta} f(x(\zeta,\eta), y(\zeta,\eta))|J|d\zeta d\eta$$

The divergence operator applied to the test function must also be transformed to the reference element. For the specific example of triangular elements discussed here, that is given by $\nabla_{x,y} v(x,y) = J^{-1}\nabla_{\zeta,\eta}\hat{v}(\zeta,\eta)$ where the gradient operator is a column vector

$$\nabla_{\zeta,\eta} = \begin{bmatrix} \frac{\partial}{\partial\zeta} \\ \frac{\partial}{\partial\eta} \end{bmatrix}$$

and $\hat{v}$ will be used to denote the test or basis functions in the reference coordinate system.

With an appropriate Jacobian matrix, the same procedure applies for transformation of x, y, z to a reference coordinate system $\hat{x}, \hat{y}, \hat{z}$. From Eq. 2.6 and based on the definition of the Jacobian matrix show in Eq. 2.7, it follows that for the simple example case of linear basis functions on triangular elements the coefficients of the Jacobian matrix are constants for each triangular element. In general, the Jacobian matrix can be a non-constant function of the reference coordinates and Cartesian coordinates.

Surface integrals are also needed on each element. The following equalities [30] relate coordinates between ζ and η to x and y.

$$d\vec{\zeta} = \begin{bmatrix} \frac{\partial x}{\partial\zeta} \\ \frac{\partial y}{\partial\zeta} \end{bmatrix} d\zeta \qquad d\vec{\eta} = \begin{bmatrix} \frac{\partial x}{\partial\eta} \\ \frac{\partial y}{\partial\eta} \end{bmatrix} d\eta$$

This can be used for the simple example case of triangle with linear elements discussed here. For

example on the face where η is constant

$$\vec{n}_{1,2} d\ell_{1\to 2} = \begin{bmatrix} \frac{\partial x}{\partial \zeta} \\ \frac{\partial y}{\partial \zeta} \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} d\zeta$$

where $\vec{n}_{1,2}$ is the outward normal and $d\ell_{1\to 2}$ is a differential length of side 1,2 of the triangle (Figure 2.2). In this case, the result is simply $\vec{n}_{1,2} d\ell_{1\to 2} = \vec{n}_{1,2} L_{1,2} d\zeta$ where $L_{1,2}$ is the length of side 1,2 and the same procedure leads to a similar equality for the two other sides. Additionally, the same procedure applies to other basis functions and in three dimensions.

The spatially discretized version of Eq. 2.2 can be written in terms of matrix elements. The specific formulation of the integrals for triangles can be determined from the definitions provided previously, but the integrals are written in a form below that is generally applicable to different basis functions and in different dimensions with the appropriate interpretation of the integrals.

$$\begin{aligned} M_{i,j} &= \int_{\hat{h}} \hat{u}_j \hat{v}_i |J| d\hat{h} \\ S_{i,j} &= \int_{\hat{h}} -\hat{u}_j \nabla^T_{\hat{x},\hat{y},\hat{z}} \hat{v}_i (J^{-1})^T |J| d\hat{h} \\ F^{\hat{h},k}_{i,j} &= \int_{F^k_{\hat{h}}} \hat{u}_j \hat{v}_i d\mathbf{s}_k \end{aligned} \tag{2.8}$$

In Eq. 2.8 $\mathbf{s}_k$ is the outward normal for reference element face number k. $F^k_{\hat{h}}$ is meant to denote face k of triangle $\hat{h}$.

2.1.1 Precomputed Local FEM Matrices and Their Important to Efficient Solves

For each triangle in the mesh, the coefficients in the matrices M, S, and F defined in Eq. 2.8 can be computed and stored as part of an initial step before beginning the solve for any angular flux values. In practice, the coefficients of these matrices are computed using a quadrature formula. Even when the form of the integrals is simple, using quadrature has the advantage that a generic piece of program logic can compute the value of these coefficients for a variety of different basis functions instead of having to program in analytical results for different possible basis functions

that might be used. In this dissertation, only elements with straight faces are considered, that is elements where the surface normal is constant and so the form of F of interest here is $F_{i,j}^{\hat{h},k} = \int_{F_{\hat{h}}^k} \vec{n}_k \gamma \hat{u}_j \hat{v}_i ds$ where ds is a differential length or area along the face k of the reference element, $\vec{n}_k$ is the normal for the face of the element in the actual coordinate system, and γ is some constant. With these definitions Eq. 2.4, which is to be enforced within each triangle in the mesh, can be written as $A^m \Psi = q_{(t,m)}$ where Ψ is the column vector consisting of the $\tilde{\psi}_{m,j}^{n+1}$ values for the element. The superscript m is not meant to mean a power but instead denotes that these coefficients are defined for the S_N angular flux direction $\vec{\Omega}_m$. The source $q_{(t,m)}$ is written simply as a source of angular flux. The coefficients of A^m and values of the rhs vector $q_{(t,m)}$ are defined as follows:

$$\begin{aligned} A_{i,j}^m &= \sigma_t M_{i,j} + \vec{\Omega}_m \cdot S_{i,j} + \sum_{\vec{n}_k \cdot \vec{\Omega}_m > 0} \vec{\Omega}_m \cdot F_{i,j}^k \\ q_{t,i,m} &= \sum_j (q_{v,j,m} + q_{s,j,m}^n) M_{i,j} - \sum_{\vec{n}_k \cdot \vec{\Omega}_m < 0} \sum_{j*} \vec{\Omega}_m \cdot F_{i,j*}^k \psi_{j*}^{n+1} \end{aligned} \tag{2.9}$$

In any finite element computation, the values of M, S, and F obviously need to be computed. In the case of solving the DG discretized S_N equation with source iteration, the equation is solved in a matrix free way element by element as described in the previous section. Therefore, the values of M, S, and F could be computed each time they are needed to solve for the angular flux within an element. As discussed in Section 1.2, solving the S_N equation in multiple dimensions typically involves a large number of unknowns. Since M, S, and F do not depend on $\vec{\Omega}$ or energy in an energy dependent calculation, the M, S, and F matrices can be computed once at the start of the calculation for each cell, stored, and then used for each S_N direction. This is a balance between computational speed and memory footprint. Obviously even more time could be saved by example storing $\vec{\Omega}_m \cdot S_{i,j}$ for each element and each m instead of just S, but for anything more than a small number of directions, this usually is prohibited by memory limits. A further efficiency can be realized for multi-group calculations. In that case, $\vec{\Omega}_m \cdot S_{i,j}$ and $\sum_{\vec{n}_k \cdot \vec{\Omega}_m > 0} \vec{\Omega}_m \cdot F_{i,j}^k$ can be computed once for each direction and then these contributions to the FEM matrices used for each energy group without having to recompute them. This amortizes the cost of these two

computations over all of the energy groups and lowers the computational cost per unknown. These two techniques are used by the S_N sweeper in Chi-Tech [31].

2.2 Solving the DG Discretized S_N Equation with Sweeping

The linear systems arising from discretizing the streaming plus collision operator (the left hand side) of Eq. 1.4 are logically block lower triangular for many commonly used spatial discretizations including DG discretizations. Logically block lower triangular as used here means that the matrix for the linear system, if formed, may not be block lower triangular, but that there is a permutation such that the matrix is block lower triangular. For example, the upwind DG finite element discretization discussed in the previous section on a two-dimensional mesh with no concave cells would result in a logically block lower triangular matrix for each of the S_N equations with any angular quadrature set.

Sweeping refers to a matrix free method for directly inverting the logically lower triangular systems with forward substitution. It directly inverts the streaming plus collision operator. The key to inverting the streaming plus collision operator in a matrix free manner is that the cells in the mesh must be visited in the order which leads to a block lower triangular system. This idea is illustrated in Figure 2.3 for a uniform mesh of 9 cells and one particular direction. This ordering of cells will be referred to as the sweep ordering. The yellow cells in the figure have all upwind information they need to compute their angular flux solutions. Once the angular flux solution is computed in the yellow cells, their downwind neighbors can compute their solution in the next stage. As seen in this figure, the yellow cells appear to "sweep" across the mesh hence the term sweep is generally used to describe solving the S_N equations in this manner. The term "sweep" in this dissertation will generally be used to mean solving all the directions in the S_N quadrature set, not just a single direction. The ordering of cells leading to a block lower triangular system can be different for each direction when the mesh is unstructured. When the mesh is uniform, all directions pointing in the same octant will have the same sweep ordering, but directions pointing in different octants will have different orderings.

There are multiple reasons to implement sweeping for solving the S_N transport equation. As

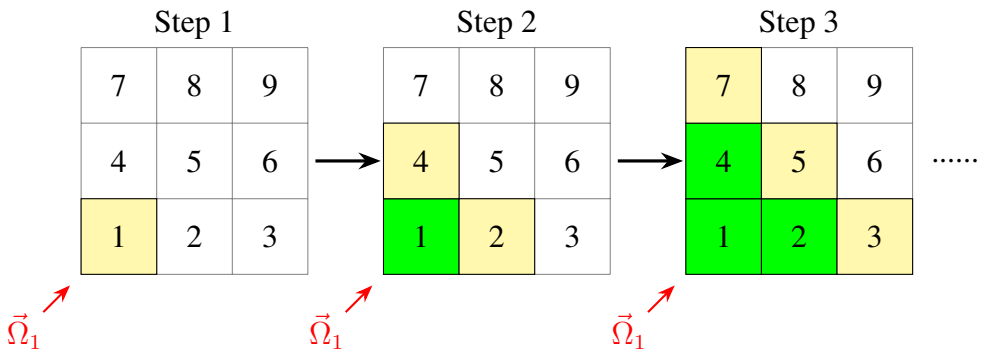


Figure 2.3: Depiction of the first three stages in solving for a particular flux direction $\vec{\Omega}_1$ on a uniform mesh with "upwind" discretization. The flux in the yellow cells are being solved for in the shown stage and the solution in the green cells has already been computed. The process is continued until all cells are green. The same idea applies to any other direction $\vec{\Omega}_m$ although the order that cells are solved in may be different.

mentioned above, the method is matrix free. Secondly, if you have a lower triangular system, forward substitution will generally be the most efficient way to solve the system. Lastly, although inversion of the block lower triangular system is generally a serial process, the sparse nature of the linear systems arising from most discontinuous FEM spatial discretizations used for the S_N equation allows for parallelism of the forward substitution. Parallel implementation of sweepers is discussed in more detail in the next section. Sweeping is discussed specifically in this section because this algorithm will be used to solve the NL-S_2 system in addition to the S_N system. As mentioned in the introduction, a significant potential advantage of NL-S_2 acceleration over other techniques is that the NL-S_2 equations can potentially be solved using sweeps, eliminating the need to use a different type of linear solver for the low order system.

The sweep ordering required among the cells to invert the streaming plus collision operator can be represented as a directed graph. The vertices of the graph represent the cells and arrows point from a cell that has outgoing angular flux to the cell that depends on that flux. The graphs are useful because there are many common algorithms for representing and manipulating directed graphs using computers and so they are important for actually computing the proper sweeper ordering. This idea is explained through several specific examples.

Figure 2.4 shows a uniform mesh consisting of a 3x3 grid of square elements. Consider the example of the three directions $\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3$ shown in the figure which all point in the same octant. The angular flux solution in each cells depends on its neighbor cells solution through the upwind fluxes and this information is represented as a directed graph in Figure 2.5. An example graph for directions $\vec{\Omega}_{10}, \vec{\Omega}_{11}, \vec{\Omega}_{12}$ is shown in Figure 2.6. Some simple ideas about how this type of graph can be constructed and used to determine a sweep ordering will be discussed next. This is only a brief overview since the use of these algorithms are common and they were already implemented in Chi-Tech for solving the S_N equation. Considerations specific to using these algorithms as part of solving the NL-S_2 equations are discussed in Section 3.3.

To construct the directed graph graph for any particular angular flux direction, a data structure can first be created that has a vertex for each element in the mesh and includes the ability for each vertex to store which other vertices it is connected to through an outgoing arrow and also those it is connected to through an incoming arrow. The directed graph can then be constructed by iterating through the entire mesh cell by cell in the following way. For each cell, each face is considered and a dot product is taken between the outward face normal and the angular flux direction of interest. If the dot product is positive, then the neighboring element is added to a list of elements or vertices in the graph to which the current vertex is connect by an outgoing arrow. If the the dot product is negative, then the neighboring element is added to a list of vertices to which the current vertex is connect by an incoming arrow. In the case of a uniform mesh like the simple example shown, the directed graph created in this way will be acyclic meaning there are no cycles in the graph. Additionally, for all unstructured meshes investigated in this work, for example that two dimensional mesh shown in Figure 3.1 or the three dimensional extruded mesh used in Section 4.2, the directed graph used to compute a sweep ordering will be acyclic. Cycles in the directed graph are discussed further in Section 2.2.2. For the purposes of the discussion in this section, the graph being acyclic simply means it can be used to determine a sweep ordering without additional steps being required to resolve cyclic dependencies in the graph.

The directed graph once constructed can be used to determine a sweep ordering in the following

way. First an empty list is initialized to which vertices will be added in a specific order. This will be called the sweep ordering list. Next, the vertices of the graph are all visited and any vertex which has zero vertices in its list of vertices on which it has an incoming dependency is added to the sweep ordering list of vertices. In the simple case of the uniform mesh shown, this search will obviously find only a corner element, for example, in the directed graph constructed for $\vec{\Omega}_1, \vec{\Omega}_2,$ or $\vec{\Omega}_3$, only element C1 will be added to the sweep ordering list in this first step. More than one element may in general be added to the sweep ordering list in this first step when an unstructured mesh is used and/or the domain being meshed in not a simple rectangular or cubic shape. For each vertex that was added to sweep ordering list, the following procedure is performed. Each vertex that is connected to a vertex in the sweep ordering list and is connected by an arrow going from the vertex in the sweep ordering list to the connected vertex is visited. The count of incoming dependencies for this connected vertex is decremented by one. After decrementing, if the total number of incoming dependencies is zero, this connected vertex is added to the sweep ordering list. For the specific example of the directed graphs for either $\vec{\Omega}_1, \vec{\Omega}_2,$ or $\vec{\Omega}_3$, vertices C2 and C4 will be visited based on vertex C1 initially being the sweep ordering list. Both C2 and C4 have one incoming dependency so when this total is decremented, the new total is zero and both of these vertices will be added to the sweep ordering list. Note that multiple vertices can be added to the sweep ordering list during the step where connected vertices are visited. So a number is stored which is the current position in the sweep ordering list, that is the position of the current vertex in the list for which its connected vertices are being visited. For the example of an acyclic graph being discussed here, the sweep ordering list will always contain new vertices for which to visit other connected vertices up until all vertices have been added to the sweep ordering list.

The next few steps of the algorithm are followed for the example shown below of the uniform mesh for completeness. At this point, the sweep ordering list contains vertices C1, C4, and C2. The order of vertices C2 and C4 in the list is arbitrary other than that they both come after C1. Say C4 was added to the list before C2. Then the current position of vertices in the sweep ordering list will be increased by one (operations are complete for vertex C1) and C4 will be the next vertex.

Note the current position is the second vertex in the list, but there are three total vertices in the list (C1, C4, C2). The vertices connected to C4 will now be visited and their total number of incoming dependencies will be decremented by one. This leads to C7 being added to the sweep ordering list. C5 is not added to the list as its total number of incoming dependencies after decrementing is still one. The current position in the sweep ordering list is then increased by one to the next vertex in the list, C2. Its two connected vertices are visited. These include C5 which already had one value decremented from its total number of incoming dependencies and after this total is decremented again, the new total is zero and so C5 is added to the sweep ordering list. C3 is also visisted and it only has one incoming dependency so after decrementing this, C3 is added to the sweep ordering list. The current position in the sweep ordering list is increased and this processes continues until all vertices in the graph have been added to the sweep ordering list. The sweep ordering list then contains a list of mesh elements in the order in which they will be visited during a sweep to invert the S_N streaming plus collision operator. The algorithm discussed here is a specific usage of the "breadth first search" algorithm [32] which is a common graph algorithm with many applications.

2.2.1 Parallel Sweeping

Parallel implementation of the sweeping algorithm is discussed in this section for several reasons. First, the parallel scalability of sweeps is an important reason for implementing the algorithm to solve the S_N equations. Secondly, although parallel scalability of sweeping applied to the NL-S_2 equation is not a specific topic of this dissertation, the three-dimensional problem investigated in Section 4.2 is too large to be run on a single processor and so some discussion of the parallel implementation is required. Some basic ideas are introduced in this section and specific parallel implementations relevant to this work are discussed.

For the multigroup S_N transport equation, many possible parallel algorithms are possible to try and make use of multiple processors to speed solution time or to run a larger problem in the same amount of time as a smaller problem. A good review of early research into solving the transport equation in parallel is given by Zerr [33]. Some parallel paradigms involve each processor having a copy of the mesh and being able to solve for unknowns in the whole domain simultaneously.

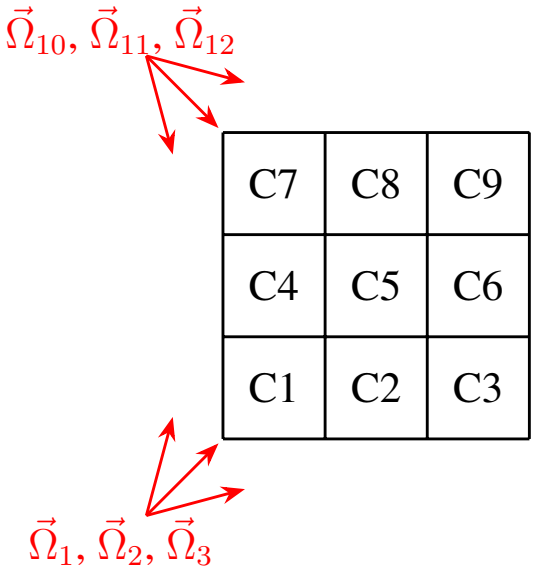


Figure 2.4: The grid depicts a 2D domain discretized with a structured mesh of 9 cells

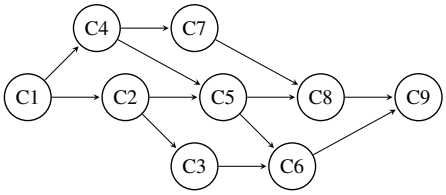


Figure 2.5: Dependencies among cells shown to the left for directions $\vec{\Omega}_1, \vec{\Omega}_2, \vec{\Omega}_3$

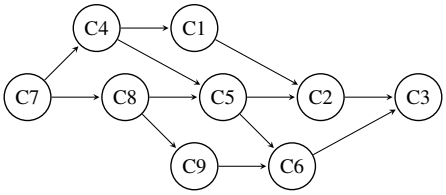


Figure 2.6: Dependencies among cells shown above for directions $\vec{\Omega}_{10}, \vec{\Omega}_{11}, \vec{\Omega}_{12}$

For example, in a multigroup problem, one could use the same number of processors as there are energy groups and have each processor perform a sweep solve with the intergroup scattering source lagged. A limitation of this methodology and similar ideas is that there are typically many more processors available than energy groups and that the memory requirement for each processor to have an full copy of the spatial mesh is too large to be practical for many three dimensional problems of interest. The solution to this is that the spatial mesh and associated unknowns must be distributed across the processors such that each processor owns only a portion of the entire mesh.

A simple example of a distributed mesh is depicted in Figure 2.7. The grid of larger black squares with Px written in the center with x being an integer represent the processor domain for processor Px and the lighter grid of squares within these larger squares represents the problem mesh owned by processor Px. Processor Px does not know about the mesh or associated unknowns for processor Py since each processor has its own distinct memory other processor doe not have access to. For one processor to know about the unknowns that another processor owns, the two processors must pass messages back and forth. Obviously, many meshes of interest are unstructured. The potential complexity introduced by unstructured meshes is briefly discussed in the next section, however the structured mesh and structured mesh partitioning shown in Figure 2.7 is sufficient to discuss the concepts of parallel sweeps that are relevant to this dissertation.

An important reason to implement a sweep solver for the solving the S_N equations is that in addition to being efficient, the algorithm exhibits excellent parallel scalability in practice. This is not necessarily obvious since performing forward substitution of a lower triangular matrix is in general a serial process. An additional complexity of parallel sweeps involves sweep front collisions which are described in [34]. To summarize here with a specific example, consider again the distributed mesh shown in Figure 2.7. First note that the order in which processors must work to invert the streaming plus collision operator for a specific direction, that is the sweep ordering of the processors, can be represented again as a directed graph. The graph can be constructed and used in a manner similar to the described in the previous section. An example directed graph is shown in Figure 2.8. When using a parallel sweep solver, the four processors owning a corner

domain; processors P1, P3, P7, and P9, can start sweeping while the other five processors must wait for these four to finish solving a direction and communicate to them their flux solution. After P1 and P7 finish solving for whichever direction they started with, as an example, P4 will get messages from both these processors and will have to pick from two directions. After processors P4, P2, P6, and P8 each pick a direction to solve for, processor P5 will get messages from four processors and must pick some order in which to work on the four directions. As processor P5 works on directions a queue builds up of work.

The first work to show that sweeps could be scalable on a large scale was [35] but the issue of sweep front collisions were not explicitly studied. The issue of sweep front collision was explicitly considered in [34] and this work showed they are not a problem in practice preventing sweeps from being a scalable algorithm. Parallel sweeps show a theoretical growth in solve time for weak scaling that behaves like O(Pˆ1/d +M) where P is the number of processors, M the number of directions, and d the spatial dimension of the problem. The theoretical growth in solve time growing polynomialy with the number of processors is not ideal. Other popular methods for solving linear systems such as multigrid methods have scaling laws where the solution time grows logarithmicaly with the number or processors. Despite this, parallel sweeps are in practice highly scalable. The theoretical scaling law shows that when M is large, that is there are many directions in the angular quadrature set, increasing the number of processors will have a small impact on parallel efficiency initially. For structured meshes, parallel sweeps have been shown to have good efficiency out to hundreds-of-thousands of processors [36] and are among the fastest methods available.

Other parallel algorithms that still allow for the use a distributed mesh are also possible. One that is relevant to this work is a block-jacobi like iteration. The blocks in this case are the processor domains, that is the entire linear system accounted for by the section of mesh owned by a processor minus only the incoming angular fluxes which are owned by a neighboring processor. Instead of determining a processor ordering in which to solve so that the streaming plus collision operator is inverted, all of the processors can begin working simultaneously where the information each processor needs from its neighbor is simply lagged first at an initial guess and then after all processors

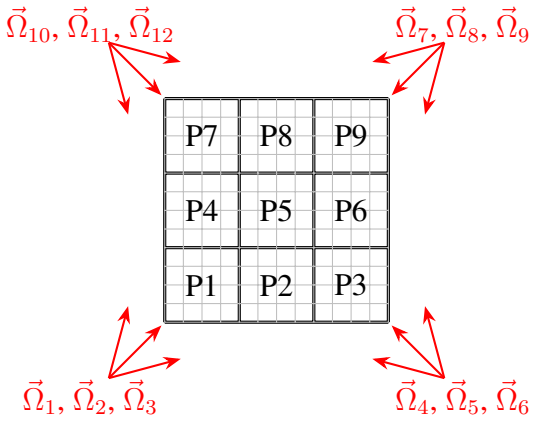


Figure 2.7: The grid depicts a problem domain distributed over 9 processors. In this example, there are 12 angular flux unknowns, three directions in each quadrant.

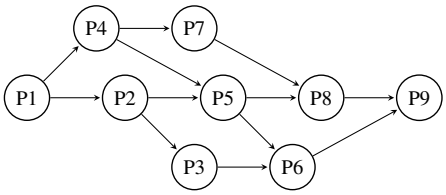


Figure 2.8: The dependencies among cells shown to the left for directions $\vec{\Omega}_1$, $\vec{\Omega}_2$, $\vec{\Omega}_3$ can be represented by the directed graph shown here

have completed one solve and communicated, at the next iterate. This is iteratively solving a block lower triangular system in the example covered in this section, but the reason to do this instead of forward substitution is the increased parallel efficiency (all of the processors can begin work immediately with no waiting) and also the ease of implementation since solves do not have to be scheduled in a particular order. Despite these benefits, block-jacobi like parallel implementations have the same problem as simple jacobi iteration, including degraded effectiveness as the linear system grows in size and poorer convergence rates as the diagonal dominance of the linear system decreases. For an analysis of this method for the case of transport, see [37]. A block-jacobi like parallel implementation is tested in Section 4.2.

2.2.2 Cyclic Dependencies

The directed graph shown in Figure 2.8 representing the dependencies among the processors in Figure 2.7 for solving the S_N equations for $\vec{\Omega}_1$, $\vec{\Omega}_2$, $\vec{\Omega}_3$ is acyclic. The graphs shown in Figure 2.5 and Figure 2.6 are also acyclic. This means that starting at any vertex, there are no paths that lead back to the vertex. A cycle is a path that contains at least one edge and starts and ends with the same vertex. A path is a sequence of vertices in the order in which they can be reached form start

to finish by following connecting edges. A graph with no cycles is acyclic. A example of a graph with a cycle is shown in Figure 2.10 where the cycle is {C1,C2,C1}. This graph represents the dependencies among the four cells shown in Figure 2.10 for the direction shown in the figure. In this figure, cells C1 and C3 consist of 5 sides each sharing two of these sides. For the angular flux direction shown, cell C1 has its two exterior faces as inflow boundaries, these are given by the boundary condition, but one of the internal faces shared with cell C3 is also an inflow boundary. The other internal face shared between these two cells is is an inflow boundary for cell C3. Thus cell C3 depends on cell C1, but cell C1 depends on cell C3 and this is presented as a cycle in the graph shown. This is a very simple example, but it is similar to the type of cyclic dependency that will be encountered when examining the NL-S_2 equations on unstructured meshes.

The idea of cycles is also relevant to parallel sweeps. The simple example partitioning shown in Figure 2.7 is obviously easily applied to a structure mesh. But if a mesh is unstructured, the partitioning among processors, even if a regular portioning grid is used, will have jagged edges. This means that a graph with a cycle like that shown in Figure 2.10 might express the dependencies among the processors as the domain partitions may look similar to the cell shapes shown in Figure 2.9.

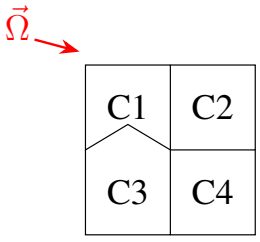


Figure 2.9: Schematic of discretization which leads to a parallel "cycle" for the direction shown

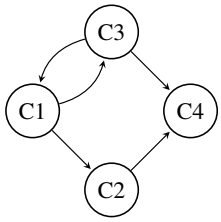


Figure 2.10: The directed graph related to mesh shown

2.3 DG Discretization NL-S_2

The main idea of the NL-S_2 formulation investigated in this dissertation is that the coefficients for the S_N linear systems are simply integrated and averaged over part of the unit sphere. This is discussed generally in Section 1.3.2.1 where the averages are defined, that is the NL-S_2 equation will consist of eight partial range scalar fluxes that must be solved for and the scalar flux is simply the sum of these eight partial range values. Since the NL-S_2 formulation is rather simple, there is not much additional information to state beyond the overview of the DG discretized S_N system already provided. Despite the relative simplicity, the averaging process is discussed first to show clearly that the NL-S_2 scalar flux solution is the same as the S_N solution. The NL-S_2 linear system is then examined further with some specific examples that illustrate some points important to the use of an S_N sweeper for solving the NL-S_2 equations.

As discussed in the previous sections, the S_N streaming plus collision operators are usually inverted in a matrix free way by visiting each element in mesh in the proper order and solving the linear system shown in Eq. 2.4 in each element. But the DG discretized NL-S_2 system will first be written as one large linear system to make it clear that when angular flux is converged to the solution of the S_N equation, the scalar flux from the NL-S_2 system is the same as the scalar flux from the transport equation. Consider a matrix B^m with coefficients defined by Eq. 2.9 plus the face flux terms shown on the rhs of Eq. 2.9. That is $A^m_{i,j}$ and $\sum^K_{\vec{n}_k\cdot\vec{\Omega}_m<0}\vec{\Omega}_m\cdot F^k_{i,j*}$ where $\sum^K_{\vec{n}_k\cdot\vec{\Omega}_m<0}$ is the sum over all inflow faces for each mesh element and is the term that encompasses coupling between the unknowns in each cell. Matrix B is a block diagonal matrix with blocks defined by $diag\{B^0, B^1, ..., B^M\}$ where the block B^m is the matrix for S_N direction m just defined.

Next, consider the vector Ψ where the unknowns are written such that $\{\Psi_m\}$ is a column vector of the spatial unknowns for angular flux direction m and $\{q_m\}$ is the column vector of fixed source values plus scattering source for direction m for each spatial point. Since the correct angular flux solution is assumed to be known, the source as written simply includes the correct scalar flux and associated scattering source. Note that as before, q is written as a source of angular flux instead of

scalar flux for convenience.

$$\Psi = \begin{bmatrix} \{\Psi_1\} \\ \{\Psi_2\} \\ \vdots \\ \{\Psi_m\} \\ \vdots \\ \{\Psi_M\} \end{bmatrix}, q = \begin{bmatrix} \{q_1\} \\ \{q_2\} \\ \vdots \\ \{q_m\} \\ \vdots \\ \{q_M\} \end{bmatrix}$$

Finally consider a matrix $W = diag\{W_{\Omega_1}, W_{\Omega_2}, ..., W_{\Omega_8}\}$ where W_{Ω_n} is a matrix of size I rows and $I * N_m$ columns where N_m is the number of directions from the angular quadrature set that point in octant N and I is the total number of spatial unknowns. This matrix W multiplies the angular flux vector of unknowns Ψ and this produces a vector Φ which is defined to be the partial range scalar fluxes for each of the eight octants as shown below where $\{\Phi_n\}$ is a column vector of the unknowns for the partial range scalar flux in octant n.

$$\Phi = \begin{bmatrix} \{\Phi_1\} \\ \{\Phi_2\} \\ \vdots \\ \{\Phi_8\} \end{bmatrix}$$

Next, a matrix $B^n(\Psi)$ is defined by the following non-linear coefficients where $B^m_{i,j}$ are the coefficients from the linear system defined above

$$B^n_{i,j}(\Psi) = \frac{\sum_m^{\vec{\Omega}_m|\vec{\Omega}_m \in \Omega_n} w_m B^m_{i,j} \psi_{j,m}}{\sum_m^{\vec{\Omega}_m|\vec{\Omega}_m \in \Omega_n} w_m \psi_{j,m}}$$

where $\sum_m^{\vec{\Omega}_m|\vec{\Omega}_m \in \Omega_n}$ is a summation over all directions m from the S_N angular quadrature set that point in octant n denoted with the symbol Ω_n. Note that $B\Psi$ will be used to mean matrix B multiplied with vector Ψ while $B(\Psi)$ will be used to denote a matrix B with coeffi-

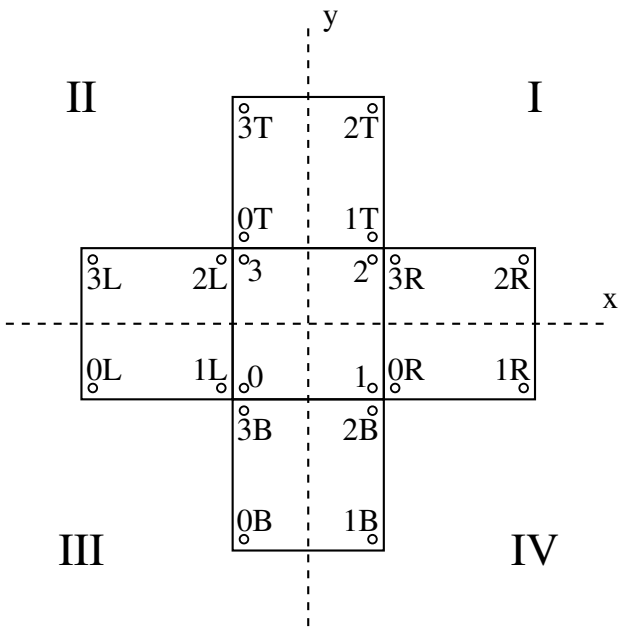


Figure 2.11: Four structured quadrature elements with the quadrants as referenced by the NL-S_2 system number with roman numerals

cients that have a non-linear dependence on Ψ. Consider a block diagonal matrix $B(\Psi)c = diag\{B^1(\Psi), B^2(\Psi), ..., B^8(\Psi)\}$ where the matrices making up the blocks are those just defined. Then the NL-S_2 can be written as $B(\Psi)\Phi = Wq$ where $\Phi = W\psi$. This is obviously the exact same linear system as $WB\Psi = Wq$.

2.3.1 Sweeping the NL-S_2 Equations

Some features of the NL-S_2 linear system relevant to the use of a S_N style sweeper as a solver are introduced next by two specific examples. Both examples are two dimensional structured meshes, one structured quadrilaterals and the other structured triangles. It will be apparent that in the case of structured quadrilaterals, the NL-S_2 linear system have an obvious similarity to the S_N equation. However, for any type of mesh other than this, as seen in the example of structured triangles, couplings between cells exist in the NL-S_2 equation which do not exist in S_N system and this has implications for the performance of a sweep solver.

Consider the mesh shown in Figure 2.11 with five square mesh cells. To show the NL-S_2 system for this type of mesh can be solved in same way that the S_N equations are, the linear

systems associated with solving for the partial range scalar flux in quadrant 1 are written below. The first linear system written below is for the left cell, denoted with nodes numbered with L. In this system, the coefficients $A_{i,j}^m$ are those defined in Eq. 2.9. The $F_{i,j}^{k,m}$ coefficients are defined in Eq. 2.8 where the m superscript in this case is meant to indicate that the coefficient is from a system where the outward normal is constant and so the contribution to the finite element system of the flow area is written as $F_{i,j}^{k,m} = F_{i,j}^k \cdot \vec{\Omega}_m$. The k subscript indicates the relevant outward normal where k is written as the two nodes that makeup the face. The equations are written for the first source iteration after the S_N solver has computed the iterate $\Psi^{n+1/2}$ and the initial guess for the scattering source is based on this iterate hence the total source is written simply as $q_t^{n+1/2}$. Note that $\sum_j q_{(t,j,m)} M_{i,j}$ from Eq. 2.9 will simply be written as $q_{(t,i,m)}^{n+1/2}$. After the next iterate of partial range scalar flux values are computed, the scattering source is obviously updated. The boundary condition is written simply as some source of angular flux and this is the g^m value written below. The summation $\sum_m^{\Omega_1}$ is a summation of all directions m where $\vec{\Omega}_m$ points in quadrant 1.

The important point in writing the first linear system is that $\vec{\Omega}_m \cdot \vec{n}^k > 0$ for all directions pointing in octant 1 for faces 1L2L and 2L3L. This system can be solved as written to determine the next iterate values for the partial range scalar fluxes in quadrant 1, that is $\Phi_{0L,\Omega_1}^{n+1}, \Phi_{1L,\Omega_1}^{n+1}, \Phi_{2L,\Omega_1}^{n+1}, \Phi_{3L,\Omega_1}^{n+1}$. The linear system to be solved for the bottom square in Figure 2.11 (nodes labeled with B) is not written below, but is obviously similar to that written for the left mesh square. After the unknowns in the left and bottom cell are solved for, the unknowns in the center cell can be determined by solving the second linear system written below. The unknowns for the cells below and to the left of the center cell appear on the right hand side and have already been solved for. This is analogous to how an S_N solve would progress through a structured mesh.

$$
\begin{bmatrix}
\frac{\sum_m^{\Omega_1} w_m A^m_{0L,0L}\psi^{n+1/2}_{0L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{0L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{0L,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{0L,2L}\psi^{n+1/2}_{2L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{0L,3L}\psi^{n+1/2}_{3L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3L,m}} \\
\frac{\sum_m^{\Omega_1} w_m A^m_{1L,0L}\psi^{n+1/2}_{0L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{0L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{1L,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{1L,2L}\psi^{n+1/2}_{2L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{1L,3L}\psi^{n+1/2}_{3L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3L,m}} \\
\frac{\sum_m^{\Omega_1} w_m A^m_{2L,0L}\psi^{n+1/2}_{0L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{0L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{2L,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{2L,2L}\psi^{m}_{2L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{2L,3L}\psi^{n+1/2}_{3L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3L,m}} \\
\frac{\sum_m^{\Omega_1} w_m A^m_{3L,0L}\psi^{n+1/2}_{0L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{0L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{3L,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{3L,2L}\psi^{n+1/2}_{2L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{3L,3L}\psi^{n+1/2}_{3L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3L,m}}
\end{bmatrix}
\begin{bmatrix}
\Phi^{n+1}_{0L,\Omega_1} \\ \Phi^{n+1}_{1L,\Omega_1} \\ \Phi^{n+1}_{2L,\Omega_1} \\ \Phi^{n+1}_{3L,\Omega_1}
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\sum_m^{\Omega_1} w_m \left(q^{n+1/2}_{t,0L,m} - \left(F^{0L3L,m}_{0L,0L} + F^{0L1L,m}_{0L,0L} \right) g^m_{0L} - F^m_{0L,1L} g^m_{1L} - F^m_{0L,3L} g^m_{3L} \right) \\
\sum_m^{\Omega_1} w_m \left(q^{n+1/2}_{t,1L,m} - F^m_{0L,1L} g^m_{0L} - F^{0L1L,m}_{1L,1L} g^m_{1L} \right) \\
\sum_m^{\Omega_1} w_m q^{n+1/2}_{t,2L,m} \\
\sum_m^{\Omega_1} w_m \left(q^{n+1/2}_{t,3L,m} - F^m_{0L,3L} g^m_{0L} - F^{0L3L,m}_{3L,3L} g^m_{3L} \right)
\end{bmatrix}
\tag{2.10}
$$

$$
\begin{bmatrix}
\frac{\sum_m^{\Omega_1} w_m A^m_{0,0}\psi^{n+1/2}_{0,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{0,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{0,1}\psi^{n+1/2}_{1,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{0,2}\psi^{n+1/2}_{2,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{0,3}\psi^{n+1/2}_{3,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3,m}} \\
\frac{\sum_m^{\Omega_1} w_m A^m_{1,0}\psi^{n+1/2}_{0,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{0,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{1,1}\psi^{n+1/2}_{1,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{1,2}\psi^{n+1/2}_{2,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{1,3}\psi^{n+1/2}_{3,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3,m}} \\
\frac{\sum_m^{\Omega_1} w_m A^m_{2,0}\psi^{n+1/2}_{0,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{0,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{2,1}\psi^{n+1/2}_{1,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{2,2}\psi^{m}_{2,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{2,3}\psi^{n+1/2}_{3,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3,m}} \\
\frac{\sum_m^{\Omega_1} w_m A^m_{3,0}\psi^{n+1/2}_{0,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{0,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{3,1}\psi^{n+1/2}_{1,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{3,2}\psi^{n+1/2}_{2,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{3,3}\psi^{n+1/2}_{3,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3,m}}
\end{bmatrix}
\begin{bmatrix}
\Phi^{n+1}_{0,\Omega_1} \\ \Phi^{n+1}_{1,\Omega_1} \\ \Phi^{n+1}_{2,\Omega_1} \\ \Phi^{n+1}_{3,\Omega_1}
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\sum_m^{\Omega_1} w_m q^{n+1/2}_{t,0,m} - \frac{\sum_m^{\Omega_1} w_m F^{30,m}_{0,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1L,m}}\Phi^{n+1}_{1L,\Omega_1} - \frac{\sum_m^{\Omega_1} w_m F^m_{0,2L}\psi^{n+1/2}_{2L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2L,m}}\Phi^{n+1}_{2L,\Omega_1} \\
\sum_m^{\Omega_1} w_m q^{n+1/2}_{t,1,m} - \frac{\sum_m^{\Omega_1} w_m F^m_{1,3B}\psi^{n+1/2}_{3B,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3B,m}}\Phi^{n+1}_{3B,\Omega_1} - \frac{\sum_m^{\Omega_1} w_m F^{01}_{1,2B}\psi^{n+1/2}_{2B,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2B,m}}\Phi^{n+1}_{2B,\Omega_1} \\
\sum_m^{\Omega_1} w_m q^{n+1/2}_{t,2,m} \\
\sum_m^{\Omega_1} w_m q^{n+1/2}_{t,3,m} - \frac{\sum_m^{\Omega_1} w_m F^{03,m}_{3,2L}\psi^{n+1/2}_{2L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2L,m}}\Phi^{n+1}_{2L,\Omega_1} - \frac{\sum_m^{\Omega_1} w_m F^m_{3,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{1L,m}}\Phi^{n+1}_{1L,\Omega_1}
\end{bmatrix}
\tag{2.11}
$$

$$
- \frac{\sum_m^{\Omega_1} w_m F^{01,m}_{0,3B}\psi^{n+1/2}_{3B,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{3B,m}}\Phi^{n+1}_{3B,\Omega_1} - \frac{\sum_m^{\Omega_1} w_m F^m_{0,2B}\psi^{n+1/2}_{2B,m}}{\sum_m^{\Omega_1} w_m\psi^{n+1/2}_{2B,m}}\Phi^{n+1}_{2B,\Omega_1}
\end{bmatrix}
$$

Next consider the mesh shown in Figure 2.12 which includes three triangles. This could represent a structured mesh consisting of triangles or part of an unstructured mesh. The important difference between the previous example is that the faces between cells are not aligned with the coordinate axes. For example, again considering quadrant 1, for a general S_N quadrature set, the value of $\vec{\Omega}_m \cdot \vec{n}^k$ for face 01 will be positive for some directions pointing in quadrant 1 and negative for other directions pointing in this quadrant. To continue this specific example, the linear system for the NL-S_2 system for quadrant 1 is shown below. This system can be solved for the partial range scalar fluxes in quadrant 1, that is $\Phi^{n+1}_{0L,\Omega_1}, \Phi^{n+1}_{1L,\Omega_1}, \Phi^{n+1}_{2L,\Omega_1}$. The linear system for the next two cells, C2 and C3, is written as the second linear system below. The linear system for these two cells is not block triangular as it would be for the S_N equation. To solve this linear system directly, cells C2 and C3 must be solved together. If using the sweep solver as it would be used for the S_N equation, only one cell would be solved at a time and so either cell C2 or cell C3 would be solved first with the required flux solution from the other cell lagged at a previous value. Then the other cell could be solved and the solution iterate updated.

$$
\begin{bmatrix}
\frac{\sum_m^{\Omega_1} w_m A^m_{0L,0L}\psi^{n+1/2}_{0L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{0L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{0L,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{1L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{0L,2L}\psi^{n+1/2}_{2L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{2L,m}} \\
\frac{\sum_m^{\Omega_1} w_m A^m_{1L,0L}\psi^{n+1/2}_{0L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{0L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{1L,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{1L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{1L,2L}\psi^{n+1/2}_{2L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{2L,m}} \\
\frac{\sum_m^{\Omega_1} w_m A^m_{2L,0L}\psi^{n+1/2}_{0L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{0L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{2L,1L}\psi^{n+1/2}_{1L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{1L,m}} & \frac{\sum_m^{\Omega_1} w_m A^m_{2L,2L}\psi^{m}_{2L,m}}{\sum_m^{\Omega_1} w_m \psi^{n+1/2}_{2L,m}}
\end{bmatrix}
\begin{bmatrix}
\Phi^{n+1}_{0L,\Omega_1} \\ \Phi^{n+1}_{1L,\Omega_1} \\ \Phi^{n+1}_{2L,\Omega_1}
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\sum_{\vec{\Omega}_m \cdot \vec{n}_{0L2L} < 0}^{\Omega_1} w_m \left(-F^{0L2L,m}_{0L,0L} g^m_{0L} - F^m_{0L,2L} g^m_{2L} \right) \\
\sum_m^{\Omega_1} w_m \left(q^{n+1/2}_{t,1L,m} - F^m_{1L,0L} g^m_{0L} - F^{1L2L,m}_{1L,1L} g^m_{1L} \right) \\
\sum_m^{\Omega_1} w_m q^{n+1/2}_{t,2L,m} + \sum_{\vec{\Omega}_m \cdot \vec{n}_{0L2L} < 0}^{\Omega_1} w_m \left(-F^{0L2L,m}_{0L,0L} g^m_{0L} - F^m_{0L,2L} g^m_{2L} \right) \\
+ \sum_m^{\Omega_1} w_m \left(q^{n+1/2}_{t,0L,m} - F^m_{0L,1L} g^m_{1L} - F^{0L1L,m}_{0L,0L} g^m_{0L} \right)
\end{bmatrix}
\tag{2.12}
$$

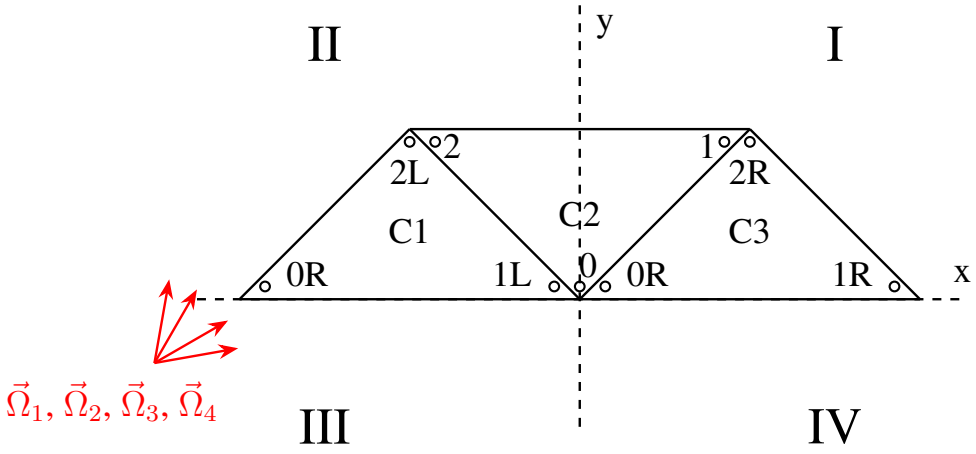


Figure 2.12: The grid depicts a 2D domain discretized with 3 triangular cells; C1, C2, and C3. The unknowns applicable to an example discontinuous discretization are represented in the figure by labeled open circles

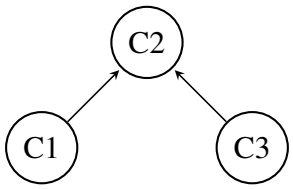


Figure 2.13: The dependencies among cells shown above for directions $\vec{\Omega}_1$, $\vec{\Omega}_2$ can be represented by the directed graph shown here

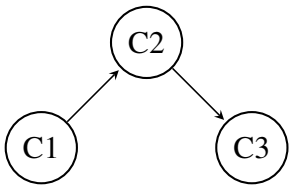


Figure 2.14: The dependencies among cells shown above for directions $\vec{\Omega}_3$, $\vec{\Omega}_4$ can be represented by the directed graph shown here

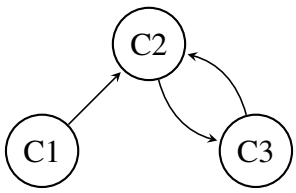


Figure 2.15: The directed graph showing cell dependencies for the NL-S_2 equation of octant one, with reference to Figure 2.12, which includes directions $\vec{\Omega}_1$, $\vec{\Omega}_2$, $\vec{\Omega}_3$, $\vec{\Omega}_4$

$$
\begin{bmatrix}
\frac{\sum_m^{\Omega_1} w_m A_{0,0}^m \psi_{0,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{0,1}^m \psi_{1,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{0,2}^m \psi_{2,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2,m}^{n+1/2}} \\
\frac{\sum_m^{\Omega_1} w_m A_{1,0}^m \psi_{0,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{1,1}^m \psi_{1,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{1,2}^m \psi_{2,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2,m}^{n+1/2}} \\
\frac{\sum_m^{\Omega_1} w_m A_{2,0}^m \psi_{0,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{2,1}^m \psi_{1,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{2,2}^m \psi_{2,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2,m}^{n+1/2}} \\
\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{0R2R}<0}^{\Omega_1} w_m F_{0R,0}^{0R2R,m} \psi_{0,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0,m}^{n+1/2}} & \frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{0R2R}<0}^{\Omega_1} w_m F_{0R,1}^{0R2R,m} \psi_{1,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1,m}^{n+1/2}} & 0 \\
0 & 0 & 0 \\
\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{0R2R}<0}^{\Omega_1} w_m F_{2R,0}^{0R2R,m} \psi_{0,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0,m}^{n+1/2}} & \frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{0R2R}<0}^{\Omega_1} w_m F_{2R,1}^{0R2R,m} \psi_{1,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1,m}^{n+1/2}} & 0
\end{bmatrix}
\begin{bmatrix}
\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01}<0}^{\Omega_1} w_m F_{0,0R}^{01,m} \psi_{0R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}^{n+1/2}} & 0 & \frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01}<0}^{\Omega_1} w_m F_{0,2R}^{01,m} \psi_{2R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2R,m}^{n+1/2}} \\
\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01}<0}^{\Omega_1} w_m F_{1,0R}^{01,m} \psi_{0R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}^{n+1/2}} & 0 & \frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01}<0}^{\Omega_1} w_m F_{1,2R}^{01,m} \psi_{2R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2R,m}^{n+1/2}} \\
0 & 0 & 0 \\
\frac{\sum_m^{\Omega_1} w_m A_{0R,0R}^m \psi_{0R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{0R,1R}^m \psi_{1R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1r,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{0R,2R}^m \psi_{2R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2R,m}^{n+1/2}} \\
\frac{\sum_m^{\Omega_1} w_m A_{0R,0R}^m \psi_{0R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{0R,1R}^m \psi_{1R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1R,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{0R,2R}^m \psi_{2R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2R,m}^{n+1/2}} \\
\frac{\sum_m^{\Omega_1} w_m A_{0R,0R}^m \psi_{0R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{0R,1R}^m \psi_{1R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1R,m}^{n+1/2}} & \frac{\sum_m^{\Omega_1} w_m A_{0R,2R}^m \psi_{2R,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2R,m}^{n+1/2}}
\end{bmatrix}
\begin{bmatrix}
\Phi_{0,\Omega_1}^{n+1} \\
\Phi_{1,\Omega_1}^{n+1} \\
\Phi_{2,\Omega_1}^{n+1} \\
\Phi_{0R,\Omega_1}^{n+1} \\
\Phi_{1R,\Omega_1}^{n+1} \\
\Phi_{2R,\Omega_1}^{n+1}
\end{bmatrix}
$$

$$
= \begin{bmatrix}
\sum_m^{\Omega_1} w_m q_{t,0,m}^{n+1/2} - \frac{\sum_m^{\Omega_1} w_m F_{0,1L}^{02,m} \psi_{1L,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1L,m}^{n+1/2}} \Phi_{1L,\Omega_1}^{n+1} - \frac{\sum_m^{\Omega_1} w_m F_{0,2L}^{02,m} \psi_{2L,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2L,m}^{n+1/2}} \Phi_{2L,\Omega_1}^{n+1} \\
\sum_m^{\Omega_1} w_m q_{t,1,m}^{n+1/2} \\
\sum_m^{\Omega_1} w_m q_{t,2,m}^{n+1/2} - \frac{\sum_m^{\Omega_1} w_m F_{2,1L}^{02,m} \psi_{1L,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{1L,m}^{n+1/2}} \Phi_{1L,\Omega_1}^{n+1} - \frac{\sum_m^{\Omega_1} w_m F_{2,2L}^{02,m} \psi_{2L,m}^{n+1/2}}{\sum_m^{\Omega_1} w_m \psi_{2L,m}^{n+1/2}} \Phi_{2L,\Omega_1}^{n+1} \\
\sum_m^{\Omega_1} w_m q_{t,0R,m}^{n+1/2} \\
\sum_m^{\Omega_1} w_m q_{t,1R,m}^{n+1/2} \\
\sum_m^{\Omega_1} w_m q_{t,2R,m}^{n+1/2}
\end{bmatrix} \tag{2.13}
$$

For the mesh shown in Figure 2.12, cells C2 and C3 are coupled for four of the octants in the NL-S_2 system, but will have no coupling in the other four octants, but cells C1 and C2 are coupled in these other four octants but not the four that C2 and C3 are coupled in. So for this mesh of structured triangles and likely for a generally unstructured mesh, the cyclic dependencies connecting cells will be spread out among the NL-S_2 directions with faces that are coupled in some

octants necessarily not being couples in others. However, it is pointed out here that it is possible to consider a mesh where all the faces may be connected in cyclic dependencies in two quadrants and have no cyclic dependencies in the other quadrants. Such a mesh is depicted in Figure 2.16. Figures 2.17 and 2.18 show the directed graphs resprenting the cell dependencies for the S_N directions in the caption and Figure 2.19 shows the possible dependencies among cells in the NL-S_2 equations for quadrants 1 or 3. It is possible that a sweeper solver will exhibit significantly different convergence properties when used for a mesh like that shown in Figure 2.12 compared to when used for a generally unstructured mesh. This is investigated briefly in Section 3.1.2.3 and at least for that sample problem, the sweep solver is still effective for a skewed mesh like that shown.

Based on the previous discussion, it is clear that for most meshes of interest, a sweeper will not exactly invert the NL-S_2 streaming plus collision operator like it would for the S_N streaming plus collision operator, that is the NL-S_2 streaming plus collision operator is not block lower triangular while the S_N streaming plus collision operator is. As discussed in Section 2.2.2, the S_N streaming plus collision operator may not be block lower triangular in general, but for all of the meshes investigated in this dissertation, it is. Section 3.1 shows an example of a relatively simple unstructured mesh. Even for the simple mesh investigated in that section, the NL-S_2 system has many cells connected to their neighbor in cyclic dependencies. Two different methods for using an S_N style sweeper with the NL-S_2 system are investigated in this dissertation as documented in Section 3.1.2. These two different methods of sweeping the NL-S_2 system are described next.

When the NL-S_2 streaming plus collision operator is not lower triangular, the sweep solver will be like a guass-seidel iteration and so some portion of the unknowns must be lagged. This is true for either of the two methods for using a sweep solver next described. The first method is to form a directed graph representing the dependencies among the cells. As discussed above and shown in Section 2.2.2 for an example problem, this graph will contain many cycles for most meshes but the cycles will mostly contain two vertices. These cycles could be broken simply by removing one of the two edges in the cycle and the edge that is removed could be chosen randomly or by following some rule like the edge pointing to the vertex with the lower numbering is removed. A simple

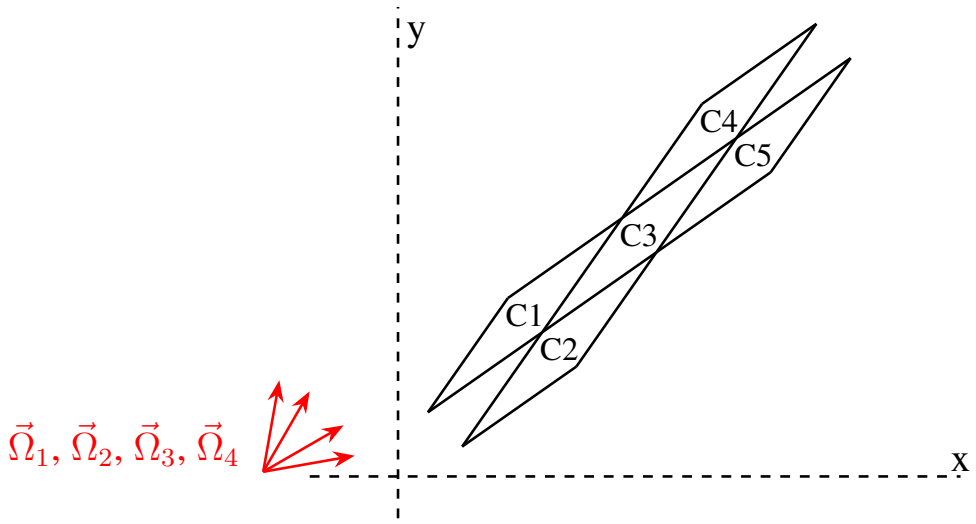


Figure 2.16: The grid depicts a 2D domain discretized with 5 diamond shaped cells; C1, C2, C3, C4, and C5. Also shown for discussion purposes are four unit directions from any arbitrary angular quadrature set.

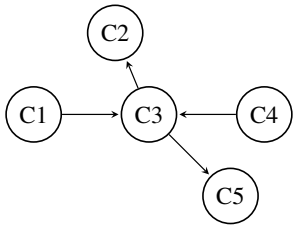


Figure 2.17: The dependencies among cells shown above for directions $\vec{\Omega}_1$, $\vec{\Omega}_2$ can be represented by the directed graph shown here

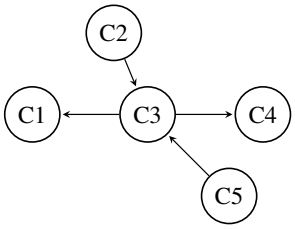


Figure 2.18: The dependencies among cells shown above for directions $\vec{\Omega}_3$, $\vec{\Omega}_4$ can be represented by the directed graph shown here

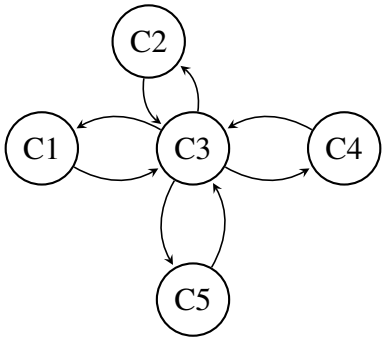


Figure 2.19: The directed graph showing cell dependencies for the NL-S_2 equation of quadrant one, with reference to Figure 2.16, which includes directions $\vec{\Omega}_1$, $\vec{\Omega}_2$, $\vec{\Omega}_3$, $\vec{\Omega}_4$

method for breaking these two vertex cycles is used in this dissertation. The edges are broken based on the effective flow area in the NL-S_2 system across the face in either direction. This is explained by a specific example. Consider again the example of the triangular mesh in Figure 2.12 for which the directed graph for the NL-S_2 streaming plus collision operator for quadrant 1 would generally by that shown in Figure 2.15. The cycle {C2,C3,C2} must be resolved, that is either cell C2 or C3 must be solved first lagging information from the other cell. The following weights for the two edges are computed and the edge with the lesser weight is removed. For example, if $w_{C2\rightarrow C3} < w_{C3\rightarrow C2}$ then cell C3 is solved before cell C2 with the required flux information from cell C2 lagged. Then cell C2 is solved for. $w_{C2\rightarrow C3}$ and $w_{C3\rightarrow C2}$ are defined below.

$$w_{C2\rightarrow C3} = \sum_{\vec{\Omega}_m\cdot\vec{n}_{01}>0}^{\Omega_1} w_m\vec{\Omega}_m\cdot\vec{n}_{01}\left(\psi_{0,m}^{n+1/2}+\psi_{1,m}^{n+1/2}\right) \tag{2.14}$$

$$w_{C3\rightarrow C2} = -\sum_{\vec{\Omega}_m\cdot\vec{n}_{01}<0}^{\Omega_1} w_m\vec{\Omega}_m\cdot\vec{n}_{01}\left(\psi_{0R,m}^{n+1/2}+\psi_{2R,m}^{n+1/2}\right) \tag{2.15}$$

The second methodology investigated for picking a sweep ordering is to simply use the sweep paths for an S_2 quadrature set in the S_N equation. This methodology is obviously simpler than the previously described method. However, computing sweep paths is not usually an expensive part of the simulation compared to performing the sweeps and converging the scalar flux so the impact of using a S_2 sweep path versus trying to compute the sweep path by consider the actual average NL-S_2 direction on the overal solve time should be small. More importantly however, is that when solving a multigroup problem, the NL-S_2 average directions for each of the energy groups will generally be different and this will lead, in general, to different sweep paths for each energy group. This causes problems when considering the general computing performance notion of data locality. If there is a different sweep path for each energy group, the entire mesh must be traversed from the starting cell to the final cell for group one and this then repeated for each energy group. This involves the CPU loading things relevant to each cell like total cross section and geometric

information the number of times there are energy groups. If an S_2 sweep path was used instead, the entire mesh is traversed only once and thus the CPU loads relevant cell information only once. The results in Section 3.1.2 show that there is little difference between the two sweeping methods in terms of iterative convergence and so for the multigroup k-eigenvalue calculations presented in Chapter 4, the simpler and more performant method of using a linear S_2 sweep path is used. Additionally, the idea presented in the next section for reducing coupling between cells, which is similar to the cycle breaking methodology described in this section, has a much greater impact on iterative convergence rates.

2.3.2 Reducing Cyclic Dependencies in the NL-S_2 Equations

As discussed in the previous section and further investigated in Section 3.1, the NL-S_2 system generated from simply averaging the DG discretized S_N system will, for everything but meshes consisting only of rectangles or rectangular parallelepiped shapes, in general be different from the S_N system in that it won't be block lower triangular. This section discusses a modification to the NL-S_2 system that can be made with the goal of reducing coupling between neighboring cells, or said a different way, removing some of the off-diagonal coefficients in the NL-S_2 linear system. There are several reasons to do this. First, when there is a cyclic dependency between cells, as discussed in the previous section, some of the unknowns in the NL-S_2 system must be lagged. The use of GMRES for solving the NL-S_2 system is discussed in the next section and this technique is used exclusively for the k-eigenvalue calculations presented in Chapter 4. As discussed in the next section, the amount of information which must be lagged has an impact on the memory requirements for the GMRES solver and it is generally desirable to have less lagged information. Beyond the memory requirements, the amount of lagged information also impacts the convergence rates of both GMRES and simple source iteration.

The technique described in this section is simple and when used for the NL-S_2 system, still results in a NL-S_2 system that produces the same scalar flux solution as the S_N system. However, until convergence of the flux, that is when the average directions used in the NL-S_2 system are not based on the exact angular flux solution, the linear system resulting from using this technique

will be different from the original. First consider the NL-S_2 system written in Eq. 2.13 when the partial range currents are the correct solution. After matrix vector multiplication, the lhs of the first equation can be written as shown in Eq. 2.16 where again the coefficients $A_{i,j}$ and $F^k_{i,j}$ are defined in Eq. 2.9 and the coefficients denoted $C_{i,j}$ are simply the portion of $A_{i,j}$ equal to $\sigma_t M_{i,j} + \vec{\Omega}_m \cdot S_{i,j}$. In the equation below, $F^k_{i,j}$ can be written as the surface normal $\vec{n}_{01}$ times some constant, that is $F^k_{i,j} = F_{i,j}\vec{n}_{01}$, since only faces with a constant surface normal are being considered here. In Eq 2.16, this constant is the same for several couplings, $F_{0,0} = F_{0,0R}$ and $F_{0,1} = F_{0,2R}$ which should be apparent considering linear basis functions on the triangles shown in Figure 2.15. These coefficients being equal to each other is written in the final equality where $F_{(0,0)=(0,0R)}$ and $F_{(0,1)=(0,2R)}$ are factored out. The point of writing this equation is to show clearly that the net flow between cells is what appears in the NL-S_2 equation when the solution is converged as seen by the $\sum^{\Omega_1}_{\vec{\Omega}_m\cdot\vec{n}_{01}<0} w_m\vec{\Omega}_m \cdot \vec{n}_{01}\psi_{2R,m} + \sum^{\Omega_1}_{\vec{\Omega}_m\cdot\vec{n}_{01}>0} w_m\vec{\Omega}_m \cdot \vec{n}_{01}\psi_{1,m}$ term and the $\sum^{\Omega_1}_{\vec{\Omega}_m\cdot\vec{n}_{01}<0} w_m\vec{\Omega}_m \cdot \vec{n}_{01}\psi_{0R,m} + \sum^{\Omega_1}_{\vec{\Omega}_m\cdot\vec{n}_{01}>0} w_m\vec{\Omega}_m \cdot \vec{n}_{01}\psi_{0,m}$ term.

$$
\begin{aligned}
&\frac{\sum_m^{\Omega_1} w_m A_{0,0}^m \psi_{0,m}}{\sum_m^{\Omega_1} w_m \psi_{0,m}} \Phi_{0,\Omega_1} + \frac{\sum_m^{\Omega_1} w_m A_{0,1}^m \psi_{1,m}}{\sum_m^{\Omega_1} w_m \psi_{1,m}} \Phi_{1,\Omega_1} + \frac{\sum_m^{\Omega_1} w_m A_{0,2}^m \psi_{2,m}}{\sum_m^{\Omega_1} w_m \psi_{2,m}} \Phi_{2,\Omega_1} + \\
&\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m F_{0,0R}^{01,m} \psi_{0R,m}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}} \Phi_{0R,\Omega_1} + \frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m F_{0,2R}^{01,m} \psi_{2R,m}}{\sum_m^{\Omega_1} w_m \psi_{2R,m}} \Phi_{2R,\Omega_1} = \\
&\sum_m^{\Omega_1} w_m A_{0,0}^m \psi_{0,m} + \sum_m^{\Omega_1} w_m A_{0,1}^m \psi_{1,m} + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m F_{0,0R}^{01,m} \psi_{0R,m} + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m F_{0,2R}^{01,m} \psi_{2R,m} \\
&= \sum_m^{\Omega_1} w_m C_{0,0}^m \psi_{0,m} + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} > 0}^{\Omega_1} w_m \vec{\Omega}_m \cdot F_{0,0}^{01} \psi_{0,m} + \sum_m^{\Omega_1} w_m C_{0,1}^m \psi_{1,m} + \\
&\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} > 0}^{\Omega_1} w_m \vec{\Omega}_m \cdot F_{0,1}^{01} \psi_{1,m} + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m \vec{\Omega}_m \cdot F_{0,0R}^{01} \psi_{0R,m} + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m \vec{\Omega}_m \cdot F_{0,2R}^{01} \psi_{2R,m} \\
&= \sum_m^{\Omega_1} w_m C_{0,0}^m \psi_{0,m} + \sum_m^{\Omega_1} w_m C_{0,1}^m \psi_{1,m} + \\
&F_{(0,0)=(0,0R)} \left(\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m \vec{\Omega}_m \cdot \vec{n}_{01} \psi_{0R,m} + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} > 0}^{\Omega_1} w_m \vec{\Omega}_m \cdot \vec{n}_{01} \psi_{0,m} \right) + \\
&F_{(0,1)=(0,2R)} \left(\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m \vec{\Omega}_m \cdot \vec{n}_{01} \psi_{2R,m} + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} > 0}^{\Omega_1} w_m \vec{\Omega}_m \cdot \vec{n}_{01} \psi_{1,m} \right)
\end{aligned}
\tag{2.16}
$$

It should be clear that the following manipulation can be made which will results in an equation equivalent to Eq 2.16. Net quantities are computed as shown in Eq. 2.17. If $J_{0/0R,net}^{01,\Omega_1} \cdot \vec{n}_{01} > 0$, then the coefficient in the NL-S_2 system $\frac{\sum_m^{\Omega_1} w_m A_{0,0}^m \psi_{0,m}}{\sum_m^{\Omega_1} w_m \psi_{0,m}}$ is replaced with $\frac{\sum_m^{\Omega_1} w_m C_{0,0}^m \psi_{0,m}}{\sum_m^{\Omega_1} w_m \psi_{0,m}} + \frac{F_{0,0} J_{0/0R,net}^{01,\Omega_1} \cdot \vec{n}_{01}}{\sum_m^{\Omega_1} w_m \psi_{0,m}}$ and $\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m F_{0,0R}^{01,m} \psi_{0R,m}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}}$ is replaced with zero. Or if $J_{0/0R,net}^{01,\Omega_1} \cdot \vec{n}_{01} < 0$, then the coefficient in the NL-S_2 system $\frac{\sum_m^{\Omega_1} w_m A_{0,0}^m \psi_{0,m}}{\sum_m^{\Omega_1} w_m \psi_{0,m}}$ is replaced with $\frac{\sum_m^{\Omega_1} w_m C_{0,0}^m \psi_{0,m}}{\sum_m^{\Omega_1} w_m \psi_{0,m}}$ and $\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m F_{0,0R}^{01,m} \psi_{0R,m}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}}$ is replaced with $\frac{F_{0,0R} J_{0/0R,net}^{01,\Omega_1} \cdot \vec{n}_{01}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}}$. Similarly, if $J_{1/2R,net}^{01,\Omega_1} \cdot \vec{n}_{01} > 0$, then the coefficient in the NL-S_2 system $\frac{\sum_m^{\Omega_1} w_m A_{0,1}^m \psi_{1,m}}{\sum_m^{\Omega_1} w_m \psi_{1,m}}$ is replaced with $\frac{\sum_m^{\Omega_1} w_m C_{0,1}^m \psi_{1,m}}{\sum_m^{\Omega_1} w_m \psi_{1,m}} + \frac{F_{0,1} J_{1/2R,net}^{01,\Omega_1} \cdot \vec{n}_{01}}{\sum_m^{\Omega_1} w_m \psi_{1,m}}$ and $\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m F_{0,2R}^{01,m} \psi_{2R,m}}{\sum_m^{\Omega_1} w_m \psi_{2R,m}}$ is replaced with zero. Or if $J_{1/2R,net}^{01,\Omega_1} \cdot \vec{n}_{01} < 0$, then the coefficient in the NL-S_2 system $\frac{\sum_m^{\Omega_1} w_m A_{0,1}^m \psi_{1,m}}{\sum_m^{\Omega_1} w_m \psi_{1,m}}$ is replaced with $\frac{\sum_m^{\Omega_1} w_m C_{0,1}^m \psi_{1,m}}{\sum_m^{\Omega_1} w_m \psi_{1,m}}$ and $\frac{\sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m F_{0,2R}^{01,m} \psi_{2R,m}}{\sum_m^{\Omega_1} w_m \psi_{2R,m}}$ is replaced with $\frac{F_{0,2R} J_{1/2R,net}^{01,\Omega_1} \cdot \vec{n}_{01}}{\sum_m^{\Omega_1} w_m \psi_{0R,m}}$.

$$
\begin{aligned}
J^{01,\Omega_1}_{0/0R,net} &= \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} > 0}^{\Omega_1} w_m \psi_{0,m} \vec{\Omega}_m + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m \psi_{0R,m} \vec{\Omega}_m \\
J^{01,\Omega_1}_{1/2R,net} &= \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} > 0}^{\Omega_1} w_m \psi_{1,m} \vec{\Omega}_m + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m \psi_{2R,m} \vec{\Omega}_m
\end{aligned}
\tag{2.17}
$$

If both $J^{01,\Omega_1}_{0/0R,net} \cdot \vec{n}_{01}$ and $J^{01,\Omega_1}_{1/2R,net} \cdot \vec{n}_{01}$ have the same sign, then the linear system for the unknowns in cells C2 and C3 will be block lower triangular. Otherwise, the situation which exists is depicted Figure 2.20 where one arrow could represent $J^{01,\Omega_1}_{0/0R,net}$ and the other arrow $J^{01,\Omega_1}_{1/2R,net}$. The reason to use this treatment on the NL-S_2 system is to reduce coupling between neighboring cells. If all of these substitutions resulted in block triangular systems, then the NL-S_2 system could be swept in the same way the S_N equations are swept, that is the streaming plus collision operator for this modified system will be inverted by the sweep. Even when coupling remains between cells, in general the coupling should be small since the situation depicted in Figure 2.20 should exist in practice when the net flow is near zero, changing from a small positive value to small negative value with respect to one of the cells. However, the treatment discussed above was written assuming the solution is known, in general it is of course not since the scalar flux is being solved for using the NL-S_2 system for acceleration. When only an angular flux iterate is known, net flows can still be computed like shown in Eq. 2.18. The same procedure can be used with these net flows, that is based on the sign of the net flow with the face normal, coefficients in the NL-S_2 linear system are replaced as described above. However, the linear system is now different after the coefficients are changed and the partial range scalar flux solution will be different. But again, once the solution is converged, the solution from the linear system after making the substitutions described above is the same as that from the system with no substitutions made. So there is a potential balance between substituting coefficients in the linear system to improve the convergence rate of GMRES or source iteration by increasing the effectiveness of the sweeper, versus potentially causing poorer acceleration because of the use of an inconsistent linear system (until convergence).

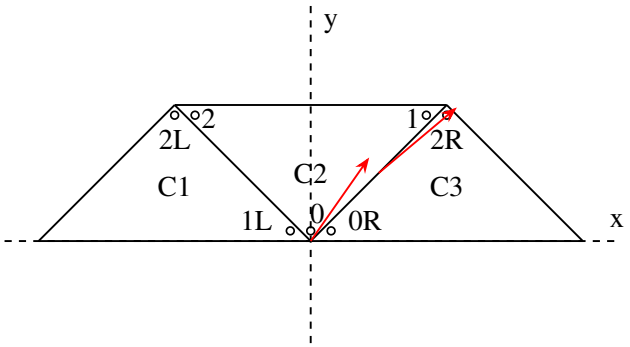


Figure 2.20: This figure shows the situation which could exist after computing a net flow with reference to the sample problem discussed in this section and the four S_N directions shown in Figure 2.12. With the red arrows representing the net flow, at one spatial location, the net flow is into cell C2 and at the other location it is into cell C3.

This is investigated further in Sections 3.1 and 3.1.2 where it is found that for several different problems run in two and three dimensions, computing net flows and substituting coefficients in the linear system significantly improves overall acceleration and the impact of using an inconsistent linear system is very small. That is the inconsistency for the problems investigated in the next chapter appear always to be small.

$$\begin{aligned} J^{01,\Omega_1}_{0/0R,net} &= \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} > 0}^{\Omega_1} w_m \psi^{n+1/2}_{0,m} \vec{\Omega}_m + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m \psi^{n+1/2}_{0R,m} \vec{\Omega}_m \\ J^{01,\Omega_1}_{1/2R,net} &= \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} > 0}^{\Omega_1} w_m \psi^{n+1/2}_{1,m} \vec{\Omega}_m + \sum_{\vec{\Omega}_m \cdot \vec{n}_{01} < 0}^{\Omega_1} w_m \psi^{n+1/2}_{2R,m} \vec{\Omega}_m \end{aligned} \tag{2.18}$$

2.4 GMRES for Solving the NL-S_2 System

The use of GMRES was introduced in Section 1.2. In that section, the operator L^{-1} was used and GMRES requires the action of this operator. For all of the meshes investigated in this work, there are no cyclic dependencies among cells for the S_N system. Some situations where cyclic dependencies will be present in the S_N equations are reviewed in Section 2.2.2. However, as discussed above, the NL-S_2 system will generally have many cyclic dependencies among neighboring

cells and even after the technique of using net flows between cells is implemented, there will still generally be some cycles. Ignoring the complexity of cycles for the moment, for example if the NL-S_2 system is used on a uniform mesh of rectangles or right rectangular prisms, then the discussion about using GMRES for solving the S_N equations can easily be generalized to the NL-S_2 system. Analogous operators to P, L, S used in Eq. 1.5 can be defined for the NL-S_2 equations and then Eq. 1.5 can be used for solving the NL-S_2 system.

When some solution data must be lagged, then one sweep solve no longer equates to L^{-1} since L^{-1} is inverting the streaming plus collision operator, but when data is lagged, the operator is not inverted, the sweeper is instead like one guass-seidel iteration. Consider the system $L\Psi^{n+1} = b^n$ where b is the rhs of the source iteration equation, that is the vector including the lagged scattering source. This equation could apply to the S_N equations or the NL-S_2 equation depending on the definition of the streaming plus collision operator L and the definition of b. When there are cyclic dependencies in the streaming plus collision operator, then the following is relevant $L = L_{bl} + U$ where L_{bl} is the block lower triangular portion of L and U is the portion of the matrix above the block lower triangular portion. The action of a sweep can be written as:

$$\Psi^{\ell+1} = L_{bl}^{-1}(b^n - U\Psi^{\ell}) \tag{2.19}$$

A new iteration index ℓ is introduced because the streaming plus collision operator is no longer directed inverted but must be solved iteratively. The converged solution of Eq. 2.19 is Ψ^{n+1}.

In Eq. 1.5, GMRES was formulated around the scalar flux vector, however it can obviously be formulated around the angular flux vector. For the specific case being discussed here where the streaming plus collision operator is not block lower triangular, the following equation can be written which incorporates L_{bl}^{-1}

$$(I - L_{bl}^{-1}(SP - U))\Psi = L_{bl}^{-1}q_v \tag{2.20}$$

The point of this is that the action of L_{bl}^{-1} can easily be computed by a sweep. If this formulation

were used for the S_N system, storing multiple krylov vectors that have a size equal to the unknown vector containing all angular flux directions would require large amounts of memory. If this formulation is used for solving the NL-S_2 equations, the memory impact is not as great since the size of the NL-S_2 solution vector is equivalent to an S_N sytem with only eight directions. Because the NL-S_2 system may have many cyclic dependencies, the GMRES formulation shown in Eq. 2.20 is simpler to implement and this is done in this dissertation, that is the entire NL-S_2 solution vector is used.

2.5 Negative Angular Flux Values

The angular flux solution for a situation where flux is incident into a region with absorption cross section, ignoring scattering or other sources, is exponential attenuation. If the interaction cross section is large, the exponential attenuation can be rapid enough that it is difficult to accurately capture the solution using linear finite element. The mesh must be highly refined. If the mesh is not highly refined where the flux solution has rapid exponential attenuation, the angular flux solution from the finite element system will result in some negative solution values. These negative solution values are not necessarily a problem, in one sense they simply indicate the mesh is not refined enough to accurately capture the flux solution in regions where these negative values are present. Additionally, it is possible that only some of the angular flux solutions will be negative and that the scalar flux solution will still be positive. However, negative angular flux solutions can lead to an ill-posed NL-S_2 linear system which can't be solved.

Revisiting the definition of the average NL-S_2 direction for octant k for the angular flux iterate $n + 1/2$, shown below, it is clear that if all the angular flux values averaged are positive, then $\vec{M}^{k,n+1/2}$ points in octant k unless the angular flux is everywhere zero in the octant in which case the average direction is simply the null vector.

$$\vec{M}_{\Omega_k}^{n+1/2} = \frac{\int_{\Omega_k} \vec{\Omega}\psi^{n+1/2} d\Omega}{\int_{\Omega_k} \psi^{n+1/2} d\Omega}$$

If there are negative angular flux values in the octant, then the magnitude of $\vec{M}^{k,n+1/2}$ is not

bounded. Additionally, $\vec{M}^{k,n+1/2}$ may not actually point in octant k. This can lead to a poorly conditioned NL-S_2 system which can't be solved.

There are many possible techniques which might be employed to resolve the issues caused by negative angular flux solutions. A simple technique will be used in this dissertation. The absolute value of angular flux solutions, abs(ψ^m), is always used when computing average directions. This technique may not be optimal, but is simple to implement and leads to a NL-S_2 system which is always well posed. When this simple technique is used, the scalar flux solution from the NL-S_2 system will be different from the S_N scalar flux. It is noted that techniques could be developed whose aim would be to replicate the S_N solution in the NL-S_2 system. However, since the S_N solution has negative angular fluxes, it's not clear what the value is of trying to replicate the S_N solution since the negative angular fluxes indicate that the mesh is not refined sufficiently. For the S_N solution to be meaningfully accurate, any negative solution values should be small and thus there should be little difference between the consistent NL-S_2 system and the system constructed using abs(ψ^m) other than that the system constructed using absolute values will be solvable. If the NL-S_2 solution is very different from the S_N solution, that might indicate that the negative angular flux solutions are large and thus the solution is not accurate. It is possible that some other technique for handling negative angular flux solutions other than simply using the absolute value could lead to a NL-S_2 acceleration scheme that is more stable. Again however, if negative angular flux solutions are large enough to cause instability in the overall acceleration scheme, the mesh likely needs to be refined further to get a meaningfully accurate answer. More investigation into handling negative angular fluxes with NL-S_2 acceleration is certainly an area for continued research. Since the converged scalar flux solution from the NL-S_2 system will not be the same as that from the S_N system when negative angular fluxes are present and absolute values of the angular flux are used, the criteria for determining convergence will be based on the NL-S_2 scalar flux.

2.6 NL-S_2 Program Implementation

A program using the Chi-Tech library [31] was created as part of this work to investigate the effectiveness of NL-S_2. As mentioned previously, the spatial discretization used throughout this work is the so called Piece-wise Linear Discontinuous (PWLD) [38] basis functions and the results presented should be applicable for other DG discretizations with properties similar to PWLD, for example muli-linear elements on quadrilaterals or hexehedrals.

As discussed in Section 2.1.1, the S_N solver can be implemented such that the solve becomes more efficient as more energy groups are added. In addition to this, product quadrature sets such as the type used exclusively in this work will have groups of directions that all have the same polar angle. For a two-dimensional mesh or extruded three-dimensional mesh, all directions with the same polar angle will have the same sweep path. Thus, a sweep path can be computed for each of the azimuthal angles used in the product quadrature set and then when each cell is solved for in the sweep path, all the directions with the same polar angle can be solved for. This does not reduce the amount of flops needed in the computation, but it can improve performance with better memory access patterns. When each cell is visited, data such as the total cross section for the cell and the precomputed parts of the local finite element system need to only be accessed from memory once to compute the solution for all angular flux unknowns in the cell that have the same polar angle. The point of discussing these items it to emphasize that the total time required to solve the S_N equations does not generally increase linearly with the number of energy groups or directions being solved for. The increases in efficiency as more energy groups are solve and the possibility of solving multiple directions using the same sweep path mean the solve can become more efficient and thus the increase in time is sublinear.

For the NL-S_2 equations, the average directions will generally be different for each energy groups and so computing the NL-S_2 analogy of S outside of a loop over energy groups is not possible. The NL-S_2 equations could be constructed in several different ways and two different implementations will be discussed next, these two implementations demonstrating some performance considerations important to this dissertation. First, the average directions could be stored

at each unknown location as well as the average outflow direction on each face with any outflow for a particular octant. These average directions could then be used with the precomputed parts for the local FEM system discussed in Section 2.1.1 to compute the local FEM system solved for the partial range scalar flux unknowns. However, this will generally lead to poorer real acceleration results than one would expect simply by comparing the number of unknowns in the NL-S_2 system versus the S_N system. This is primarily because the contribution from S times the average direction to the local FEM system must be computed for each energy group compared to $\vec{\Omega}_m \cdot S$ only needing to be computed once and so if average directions are stored and used to build the local FEM matrix each time a cell is visited, more flops are required for a NL-S_2 multigroup calculation than would be required for an S_2 calculation to build the local FEM matrices.

Instead of just storing average directions and average outflows and using these to compute the NL-S_2 local FEM matrix each time a cell is visited, the local FEM matrices can be built and stored beforehand. This obviously requires more memory than simply storing the average directions, however, since there are only eight partial range scalar fluxes for each energy group, the memory requirements may still be low enough that this method is useful. If the local FEM matrices are precomputed and stored, then they can also be factored with LU factorization once after the average directions have been computed and these LU factored matrices used with either source iteration or GMRES to converge the scattering source (and a k-eigenvalue iterate for eigenvalue problems). This was implemented code the code used in Chapter 4.

3. NL-S_2 FIXED SOURCE ACCELERATION PERFORMANCE RESULTS

A major potential benefit to NL-S_2 acceleration is that the same liner solver used to invert the S_N streaming plus collision operator may be effective for inverting the NL-S_2 streaming plus collision operator as well. For uniform meshes, it is clear as discussed in Section 2.3.1 that a sweeper is an effective method for solving the NL-S_2 equations since the NL-S_2 has the same block lower triangular structure as the S_N equations, but for any other type of mesh, the presence of many simple cycles connecting neighboring cells may render a sweep solver ineffective. The topic of this chapter is NL-S_2 acceleration of fixed source problems with unstructured and structured meshes in two and three dimensions. The investigation into fixed source acceleration is in some ways a prerequisite for the investigation of accelerating k-eigenvalue calculations which is documented in the next chapter. If there were fundamental issues preventing NL-S_2 from being useful for fixed source acceleration with unstructured meshes, it's hard to imagine how it would be useful for accelerating k-eigenvalue calculations on unstructured meshes. Additionally, many of the results presented in this section are for tests performed on simpler meshes than the C5G7 benchmark mesh investigated in the next chapter, which allowed for more simulations to be performed.

In addition to focusing on unstructured meshes, results are also presented in this chapter for problems with significantly inhomogeneous material configurations, particularly problems with cross sections rapidly changing from opaque to vacuum. As discussed in Section 1.2, one potential motivation for using NL-S_2 acceleration instead of other methods for certain problems is that NL-S_2 acceleration might be robust for problems with vacuum while implementing a robust diffusion based acceleration method may be more difficult for such problems. The results in this section show that NL-S_2 can fail to converge for some problems, although this issue can be resolved by performing more than one S_N sweep on the outer iteration.

Note that the simple technique for handling negative angular fluxes discussed in Section 2.5 is used for the heterogeneous problems investigated in this section. Lastly note that all results presented in this section use a guass-chebeychev product quadrature for the S_N equations. The

results presented in this chapter show that NL-S_2 acceleration is in general robust and that the sweeper is an effective linear solver for inverting the NL-S_2 streaming plus collision operator for a variety of problems.

3.1 Reducing Cyclic Dependencies in the NL-S_2 Equations

As discussed in Section 2.3.1, for non-orthogonal meshes, the NL-S_2 system will have cyclic dependencies among mesh cells that must be handled before the streaming plus collision operator can be inverted, either directly or as part of an iterative process, by a transport sweep. Reducing the number of cyclic dependencies may be useful for several reasons. The convergence of the iterative solve may improve and secondly, the computational costs may be reduced by decreasing the amount of computations at cell faces and also decreasing the amount of NL-S_2 flux data which must be stored. A potentially useful method to reduce the number of cyclic dependencies is to compute a net flow as described in Section 3.1. To summarize the method presented in that section, faces that connect cells in a cyclic dependency have both inflow and outflow with respect to one of the faces. These flows are replaced by the net flow. Doing this results in the same linear system when the partial range scalar flux values are exact, otherwise this new linear system using net flows across faces is different.

The extent of cyclic dependencies for a specific problem with an unstructured mesh are visualized in this section along with the effect of using net flows to reduce cyclic dependencies. In general, this technique does not remove all cyclic dependencies and it is even possible there would be no decrease in cyclic dependencies although such a case seems unlikely for any practical problem. The mesh shown in Figure 3.1 is discussed first with a homogeneous material configuration, that is the cross section for both region zero and one are set to the same values. The mesh is used with reflecting boundaries along the left and bottom axes, that is the x=0 and y=0 axes. Figure 3.3 depicts the cyclic dependencies in the NL-S_2 streaming plus collision equation by lines connecting points where the points mark cells and are placed at the cell centroids and the blue lines represent a cyclic connection across the face, that is both inflow and outflow with respect to one of the cells in the simple cycle. The cyclic dependencies are shown for quadrant 1 and quadrant 2. This example

demonstrates that even for a relatively simple homogeneous problem, the use of an unstructured mesh will lead to a considerable amount of cyclic dependencies. Note this figure was generated by examining the face flows after the first acceleration iteration, that is one S_N sweep has been performed, the scattering source on the NL-S_2 was converged, a second S_N sweep was performed, and then the face flows were examined. The amount of cyclic dependencies are similar for other iterations. After replacing flows across faces with net flows, the number of cycles is reduced to zero for this particular problem.

Figure 3.4 shows the same result discussed in the previous paragraph, but this time for the inhomogeneous material configuration described in the caption of the figure. In this figure, in addition to the blue lines representing the cyclic dependencies before replacing flows with net flow, thicker red lines mark in cycles that remain after this procedure. As opposed to the homogeneous problem, some cycles remain after replacing all flows with net flows. The purpose of Figure 3.3 and Figure 3.4 was to provide a visual demonstration of the potential usefulness of the net face flow technique for reducing the amount of cyclic dependencies. Clearly, the amount of cyclic dependencies is reduced, in the next sections, the impact this technique has on convergence and performance is investigated in detail.

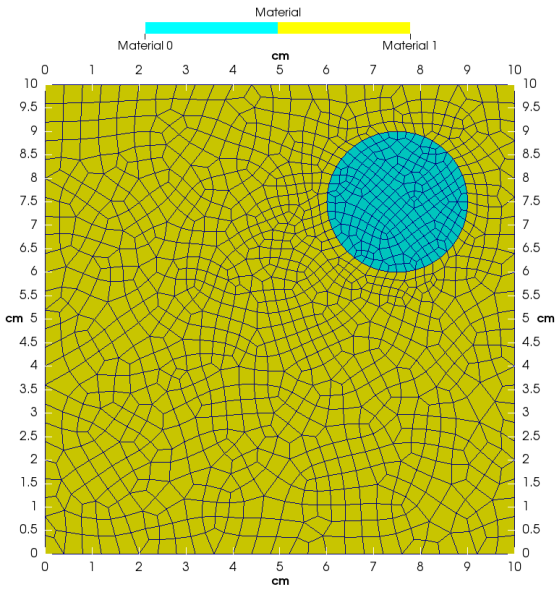


Figure 3.1: Unstructured mesh with two materials. The blue-green region inside circle in the upper right portion of the square is region 0 and the yellow region outside the circle will be referred to as region 1

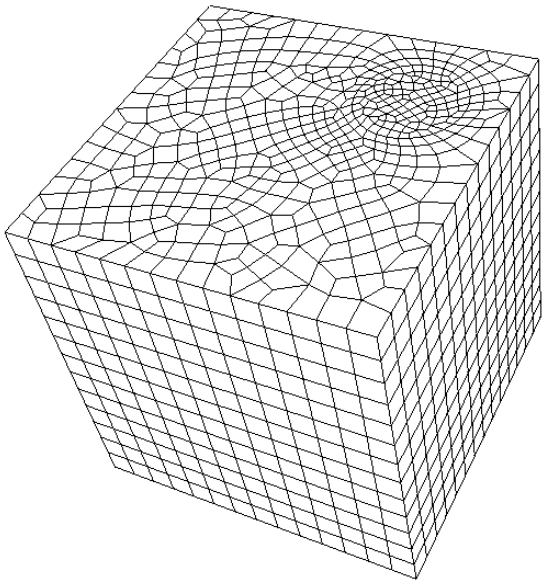


Figure 3.2: Extruded unstructured mesh

3.1.1 Directly Inverting the Streaming Plus Collision Operator

Using net flows across faces to reduce cyclic dependencies could affect convergence of the source iteration in several ways. If the number of cyclic dependencies are decreased significantly, the convergence of source iteration may improve because less unknowns must be lagged during a sweep. The convergence could also suffer because the linear system after using net flows is not consistent with the original linear system until the partial range currents are completely converged to the solution. Because cyclic dependencies are present, the streaming plus collision operator will be inverted iteratively if a sweep solve is used. The additional complication of sweeping the NL-S_2 system is discussed in the next section. In this section a direct solver is used to invert the streaming plus collision operator and the impact of solving the NL-S_2 streaming plus collision operator without any treatment versus solving the system after canceling flows across faces to compute a net flow is investigated for source iteration problems. The source iteration procedure is to start

with an initial scalar flux guess of zero, invert the S_N streaming plus collision operator to compute angular fluxes and use these to update the scalar flux iterate and compute average directions for

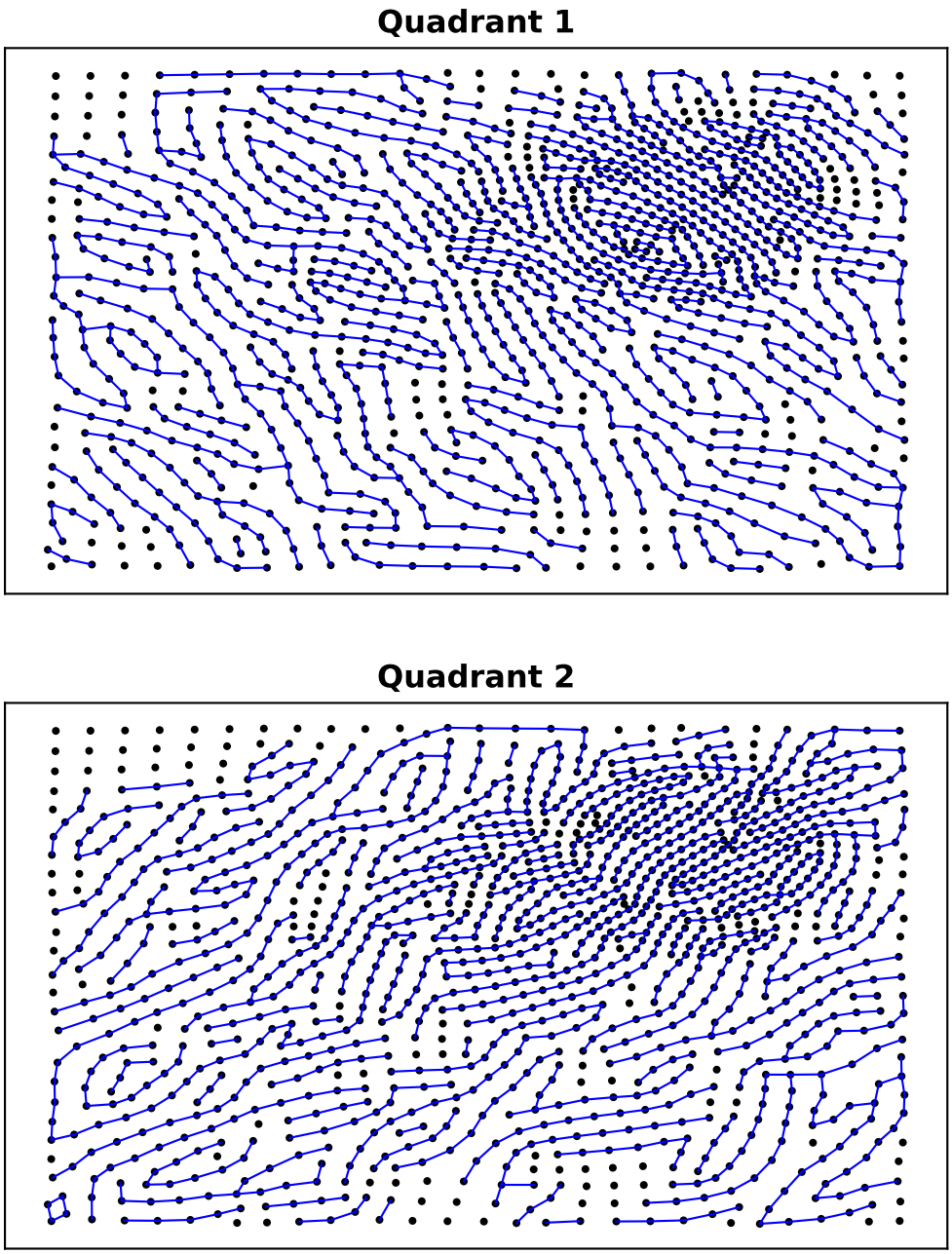


Figure 3.3: Mesh cell centroids from Figure 3.1 are marked with black dots. The cells connected by simple cycles are connected in this figure with a blue line and simple cycles remaining after computing a net flux are connected with a thicker red line. Results generated for a fixed source problem with $\sigma_t = 0.1$ and scattering ratio of 0.1

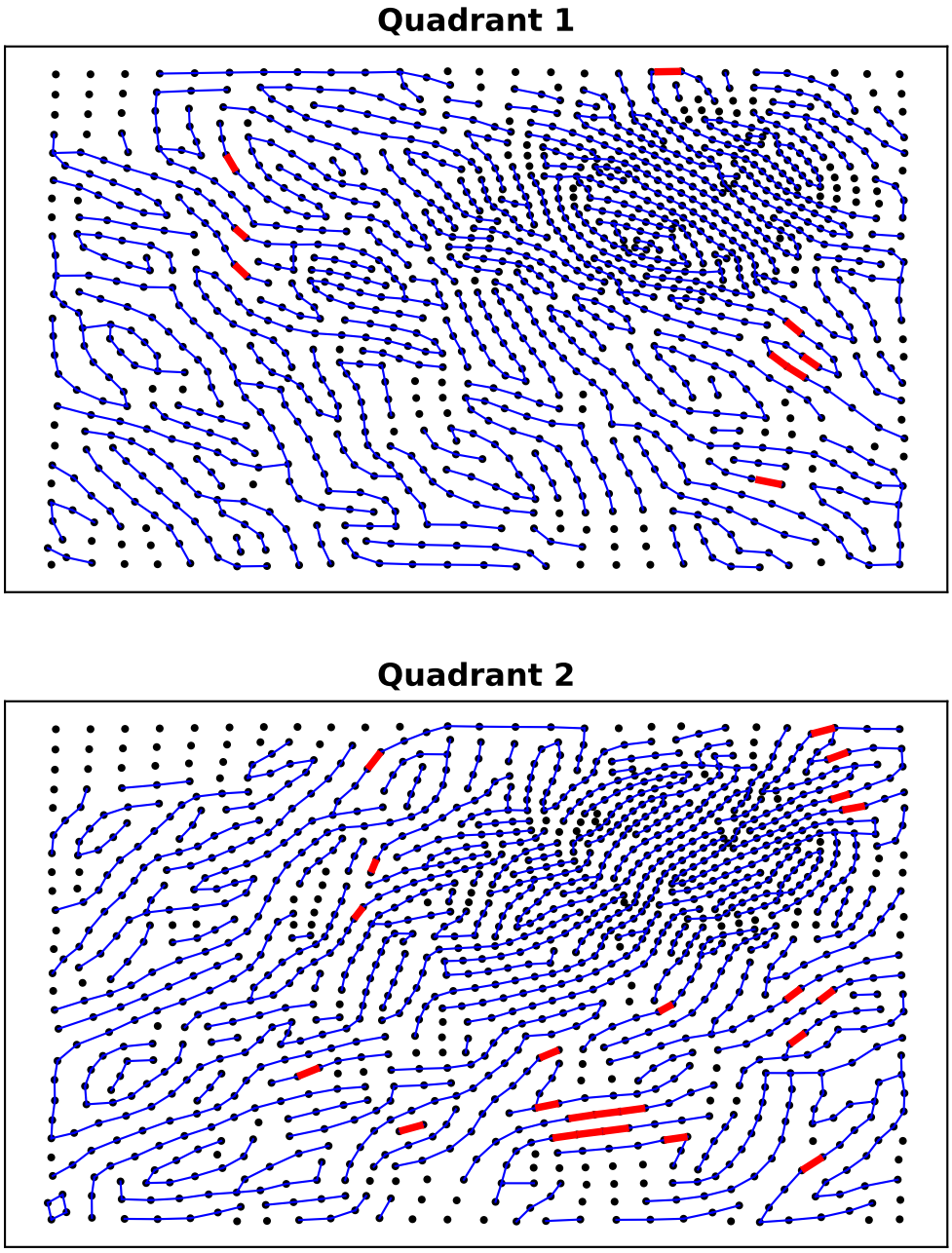


Figure 3.4: Mesh cell centroids from Figure 3.1 are marked with black dots. The cells connected by simple cycles are connected in this figure with a blue line and simple cycles remaining after computing a net flux are connected with a thicker red line. Results generated for a fixed source problem with $\sigma_{t0} = 0.1$ with a scattering ratio of 0.1, $q_0 = 1.0$ and $\sigma_{t1} = 1.0$ with a scattering ratio of 0.5, $q_1 = 0.0$

the NL-S_2 equation followed by converging the NL-S_2 scalar flux to a relative maximum relative difference of 1e-6 and then begin another iteration inverting the S_N streaming plus collision operator using the NL-S_2 scalar flux for the scattering source. This is repeated until convergence of the scalar flux measured by the maximum relative pointwise change. Inverting the S_N streaming plus collision operator to compute average directions for the NL-S_2 equation will be referred to as an outer iteration and the inner iterations are the iterations to converge to a scalar flux solution using the NL-S_2 equation.

Table 3.1 shows the impact of canceling face flows for the homogeneous problem on the total number of outer iterations (total number of S_N streaming plus collision operator inversions) and the total number of inner iterations (total number of NL-S_2 streaming plus collision operator inversions) to converge a scalar flux value. The result shown in the table is for a homogeneous cross section of $\sigma_t = 5$ cm^{-1} and a uniform fixed source and reflecting boundaries along the x=0 and y=0 axes. Values of $\sigma_t = 0.1$ cm^{-1} and $\sigma_t = 1.0$ cm^{-1} were also tested and for these values, there was no difference for any of the scattering ratios tested between using the consistent NL-S_2 system or replacing flows across faces with net flows. For $\sigma_t = 5$ cm^{-1}, there is very little difference in convergence characteristics for high scattering ratios and the convergence characteristics are the same for smaller scattering ratios. An additional test was performed by refining the mesh shown in Figure 3.1 uniformly twice and then repeating the test. In that case, there was no difference when using net flows for any of the cross section values tested. A problem with inhomogeneous cross sections was also tested. With reference to Figure 3.1, region 0 is set to a void with no source and total cross section values of 0.1 and 1.0 are tested in region 1 with scattering ratios of 0.1, 0.2, 0.4, 0.8, and 0.9. These values were tested for the mesh shown in the figure and for a refined once uniformly and then refined again uniformly. For all of these heterogeneous tests, there was no difference in total iteration counts.

A three dimensional problem could be created from Figure 3.1 by extruding it. The nature of cyclic dependencies should be similar to the two dimensional problem because the faces parallel to the x-y plane in the extruded mesh cannot have cyclic dependencies. An extruded mesh is also

Table 3.1: Comparison of total S_N source iterations and the total number of NL-S_2 source iterations to converge a scalar flux solution when using a direct solver to invert the NL-S_2 streaming plus collision operator for the scattering ratios shown for a homogeneous problem with $\sigma_t = 5\,\text{cm}^{-1}$ and uniform volumetric source using the mesh shown in Figure 3.1

Scattering Ratio	**Cycle Treatment**	**Outer Iterations**	**Inner Iterations**
0.1	Nothing	3	12
	Net Flow	3	12
0.2	Nothing	3	17
	Net Flow	3	17
0.4	Nothing	4	31
	Net Flow	4	31
0.8	Nothing	5	139
	Net Flow	6	142
0.9	Nothing	6	307
	Net Flow	7	312
0.95	Nothing	7	706
	Net Flow	7	707

used for the three dimensional k-eigenvalue calculation presented in Section 4.2. Here, a simple three dimensional problem is considered which is not extruded. The purpose of this is demonstrate that similar results are seen as the two dimensional results even when faces are not parallel to the x-y plane. The three dimensional mesh is structured but consists of tetrahedrals as shown in in Figure 3.5 and so cycles are present in the NL-S_2 system. Both homogeneous problems and inhomogeneous problems are discussed, obviously for the homogeneous problems, the banded material configuration shown in the figure is not used.

The potential importance of using net-flows on faces in the NL-S_2 system is more apparent for this three-dimensional problem. Each of the eight NL-S_2 has an average of 4,300 cyclic dependencies that must be handled by lagging fluxes. After canceling flows across faces, there are only an average of 90 cyclic dependencies. Despite the difference in total cycles, Table 3.2 shows that when a direct solver is used to invert the NL-S_2 streaming plus collision operators, there is no difference in convergence between using the exact NL-S_2 system or the net flow system for the

homogeneous problems tested. Table 3.3 shows test results for the inhomogeneous configuration described in the table caption. This test result shows a small difference between leaving the cyclic dependencies in the system and canceling them.

The purpose of this section was to demonstrate that using a net flow can drastically reduce the number of cycles and that the inconsistency introduced by doing this does not have any significant negative impact on convergence of source iteration for some representative problems. This results is further enforced in Chapter 4 where net flows are used for eigenvalue calculations in two and three dimensions and good results in terms of convergence are attained. Also of interest is how using a net flow impacts the convergence of source iteration when a sweep solver is used. This idea is explored further in the next section where a sweep solver is used to invert the NL-S_2 streaming plus collision operator instead of the direct linear solver used in this section.

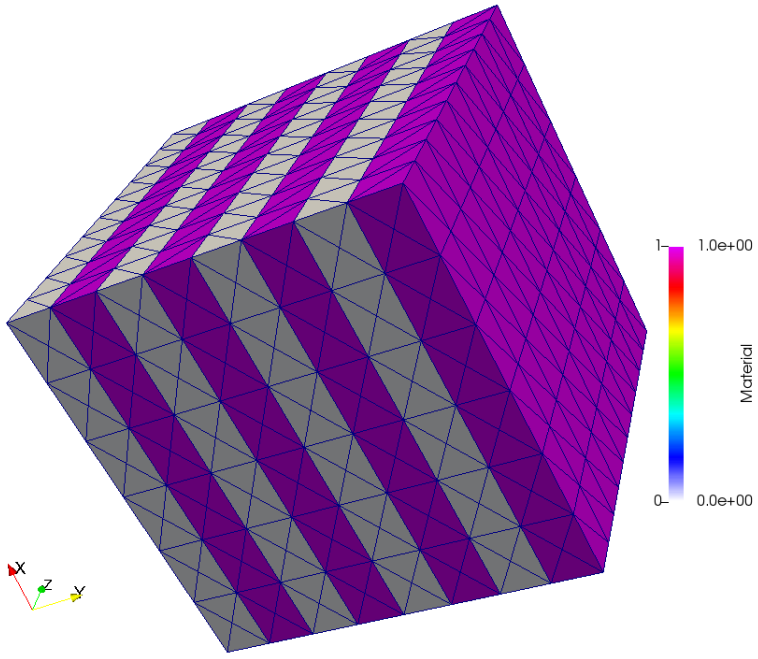


Figure 3.5: The three dimensional extruded mesh tested. The mesh consists of 5760 tetrahedrals and has a banded inhomogeneous material configuration.

Table 3.2: Comparison of total iterations (S_N solves) and the total number of NL-S_2 iterations to converge source iteration for the scattering ratios shown for a homogeneous problem using the mesh shown in Figure 3.5 for a homogeneous problem, that is with both material regions set to the same value.

Scattering Ratio	Total S_N Iterations No Cancellation	Total NL-S_2 Iterations No Cancellation	Total S_N Iterations Cancellation	Total NL-S_2 Iterations Cancellation
0.1	3	12	3	12
0.2	3	17	3	17
0.4	4	32	4	32

Table 3.3: Comparison of total S_N source iterations and the total number of NL-S_2 iterations to converge source iteration for the banded problem with the mesh shown in Figure 3.5 with $\sigma_{t1} = 0.0$, $q_1 = 0.0$ and $\sigma_{t1} = 1.0$, $q_1 = 1.0$ with scattering ratio 0.95

Total S_N Iterations No Cancellation	Total NL-S_2 Iterations No Cancellation	Total S_N Iterations Cancellation	Total NL-S_2 Iterations Cancellation
10	918	12	999

3.1.2 Using a Sweeper to Solve the nls2 Equations

The same problems investigated in the previous section are again used in this section except that a sweeper is used to invert the NL-S_2 streaming plus collision operator instead of a direct solver. For unstructured meshes, one sweep solve is like a guass-siedel iteration since the NL-S_2 system will not generally be block lower triangular. The performance of two different methods for picking a sweep ordering are investigated in this section. The first method, described in more detail in Section 2.3.1 is to build a directed graph representing the dependencies among cells in the NL-S_2 system and then use graph algorithms to remove edges from this graph until it is acyclic, the acyclic graph then giving the sweep ordering. The second method discussed in Section 2.3.1 is to pick a sweep path not based on the average directions that will be used in the NL-S_2 equation, but instead use a sweep path computed for the fixed S_2 direction pointing in the respective octant. To distinguish between these two methods when discussing them in this section, the sweeper using

a sweep path developed by considering the actual average directions in the NL-S_2 system will be referred to simply as the sweeper and the sweeper using a S_2 sweep path regardless of the actual average diction will be referred to as the sweeper using a linear S_2 sweep path.

3.1.2.1 Two-Dimensional Unstructured Mesh

Table 3.4 shows differences in total iteration counts to converge a scalar flux solution when using a sweeper to solve the NL-S_2 equations using source iteration for the problem described in the table caption. Compared to the results shown in the previous section, there is now a significant difference in the total iteration counts between either solving the consistent NL-S_2 system or replacing flows across faces with net flows. Some general trends are that there is less of a difference between the consistent NL-S_2 system and the system using net flows when the total cross section is larger. Results are shown for the mesh in Figure 3.1 and a fine mesh that is refined twice uniformly from the mesh shown. When using net flows, the total iteration counts do not increase between the coarse mesh and the fine mesh, however, when using the consistent NL-S_2 system, the total number of iterations required to converge to a scalar flux solution does increase significantly. Note also that the total number of outer iterations required does not change appreciably between the two methods. This make sense given that the previous section showed little difference between the fully consistent system and the system using net flows in addition to the fact that both methods were testing by converging the maximum difference in successive scalar flux iterates to a fixed value. Table 3.5 shows differences in total iteration counts to converge a scalar flux solution when using a sweeper with an S_2 sweep path to solve the NL-S_2 equations using source iteration for the problem described in the table caption. The results are similar between the two types of sweepers. Again, there is a significant performance difference between solving the consistent NL-S_2 equation and solving the version using net flows.

Tables 3.6 and 3.7 again show differences in iteration counts to converge to a scalar flux solution for the two different sweeper methods and source iteration but this time for the inhomogeneous problem described in the table caption. Similar to the results for homogeneous problems, there is a significant performance increase when using net flows in the NL-S_2 system. Also similar to the

results for the homogeneous problems is that there are only insignificant differences between the two methods for using the sweep solver.

Table 3.8 shows results for the same problem for the results in Table 3.4 except that GMRES is used to solve the NL-S_2 system instead of source iteration. As expected, the number of iterations required to converge a scattering source is lower when using GMRES compared to source iterations. Only results for the sweeper using a S_2 sweep path are shown because as seen in previous results, there are no significant differences between the two sweeper types when using GMRES. Again, there is a significant performance increase when using net flows across faces, but the performance gain is less than in the case of using source iteration to solve the NL-S_2 equations.

3.1.2.2 Three-Dimensional Unstructured Extruded Mesh

The effectiveness of the two sweeper methods as well as the impact of using net flows in the NL-S_2 system was investigated for the three-dimensional extruded mesh shown in Figure 3.2 as well as a version of this mesh that was uniformly refined once. Table 3.9 shows total sweeps required to converge a scalar flux solution when using a sweeper that considers the actual average NL-S_2 average direction and Table 3.10 shows the same result when using a sweeper with an S_2 sweep path. The trends seen for the two dimensional mesh are again apparent in the extruded three dimensional mesh results.

3.1.2.3 Two-Dimensional Skewed Structured Mesh

As discussed previously, an unstructured mesh will generally have many faces connected by cyclic dependencies, but a face connected this way will be so for average directions in two quadrants and will not be so connected for others. For an unstructured mesh, the distribution of cyclic dependencies among the quadrants should be relatively random. As discussed in Section 2.3.1, it is possible for a skewed mesh to have every face connected to its neighbor in a cyclic dependency in two quadrants even though the mesh itself is still structured. Such a mesh is shown in in Figure 3.6a and Figure 3.6. Table 3.11 shows total inner and outer iterations to converge a scalar flux solution for the inhomogeneous material configuration listed in the table caption. As expected based on the

Table 3.4: Comparison of total S_N solves and the total number of NL-S_2 iterations to converge a scalar flux solution when using a sweeper to solve the NL-S_2 equations using source iteration for a homogeneous problem with the scattering ratios shown and a uniform volumetric source. The coarse mesh used is shown in Figure 3.1 and the fine mesh has been refined uniformly twice.

Total Cross Section	**Scattering Ratio**	**Cycle Treatment**	**Outer Iterations Coarse Mesh**	**Inner Iterations Coarse Mesh**	**Outer Iterations Fine Mesh**	**Inner Iterations Fine Mesh**
σ_t=0.1	0.1	Nothing	3	46	3	136
		Net Flow	3	13	3	13
	0.2	Nothing	3	51	3	151
		Net Flow	3	17	3	17
	0.4	Nothing	4	74	3	208
		Net Flow	4	28	4	28
	0.8	Nothing	6	142	4	382
		Net Flow	6	65	6	65
	0.9	Nothing	6	176	6	443
		Net Flow	7	83	7	83
	0.95	Nothing	6	193	6	483
		Net Flow	7	93	7	93
σ_t=5.0	0.1	Nothing	3	14	3	23
		Net Flow	3	12	3	12
	0.2	Nothing	3	20	3	27
		Net Flow	3	17	3	16
	0.4	Nothing	4	34	4	43
		Net Flow	4	31	4	30
	0.8	Nothing	5	147	5	181
		Net Flow	6	142	6	139
	0.9	Nothing	6	326	6	393
		Net Flow	7	312	6	293
	0.95	Nothing	6	697	6	806
		Net Flow	7	660	7	624

previous results, there is again a significant performance difference when using net flows across faces. One difference between these results and those for the unstructured two-dimensional mesh investigated in Section 3.1.2.1 is that in this case, when many faces are nearly aligned with the vector $< \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2} >$ instead of random faces in an unstructured meshes, for some material configurations, the use a sweeper computed for the actual average directions does result in significant

Table 3.5: Comparison of total S_N solves and the total number of NL-S_2 iterations to converge a scalar flux solution when using a sweeper with an S_2 sweep path to solve the NL-S_2 equations using source iteration for a homogeneous problem with the scattering ratios shown and a uniform volumetric source. The coarse mesh used is shown in Figure 3.1 and the fine mesh has been refined uniformly twice.

Total Cross Section	**Scattering Ratio**	**Cycle Treatment**	**Outer Iterations Coarse Mesh**	**Inner Iterations Coarse Mesh**	**Outer Iterations Fine Mesh**	**Inner Iterations Fine Mesh**
σ_t=0.1	0.1	Nothing	3	46	3	136
		Net Flow	3	19	3	36
	0.2	Nothing	3	51	3	151
		Net Flow	3	21	3	39
	0.4	Nothing	4	74	4	209
		Net Flow	4	30	4	49
	0.8	Nothing	6	143	6	386
		Net Flow	6	65	6	88
	0.9	Nothing	7	178	6	448
		Net Flow	7	84	7	105
	0.95	Nothing	7	195	6	488
		Net Flow	7	94	7	115
σ_t=5.0	0.1	Nothing	3	14	3	23
		Net Flow	3	12	3	12
	0.2	Nothing	3	20	3	27
		Net Flow	3	17	3	16
	0.4	Nothing	4	34	4	43
		Net Flow	4	31	4	30
	0.8	Nothing	5	147	5	181
		Net Flow	6	142	6	139
	0.9	Nothing	6	326	6	393
		Net Flow	7	312	6	293
	0.95	Nothing	6	697	6	805
		Net Flow	7	660	7	624

performance differences compared to using the S_2 path sweeper. However, the mesh shown in Figure 3.6 is likely not representative of a mesh that would be encountered in practical use and so the performance difference in this case between using a linear S_2 sweep path versus accounting for the actual average direction is interesting, but probably not important.

Table 3.6: Comparison of total S_N solves and the total number of NL-S_2 iterations to converge a scalar flux solution when using a sweeper to solve the NL-S_2 equations with source iteration for a problem with $\sigma_{t,0} = 0.0$ and $\sigma_{t,1} = 1.0$ with the scattering ratios shown and $q_{v,0} = 0.0$, $q_{v,1} = 1.0$. The coarse mesh used is shown in Figure 3.1 and the fine mesh has been refined uniformly twice.

Scattering Ratio	**Cycle Treatment**	**Outer Iterations Coarse Mesh**	**Inner Iterations Coarse Mesh**	**Outer Iterations Fine Mesh**	**Inner Iterations Fine Mesh**
0.1	Nothing	3	30	3	86
	Net Flow	3	12	3	14
0.2	Nothing	4	41	4	118
	Net Flow	4	19	4	22
0.4	Nothing	5	60	5	177
	Net Flow	5	35	5	40
0.8	Nothing	9	242	10	495
	Net Flow	9	183	10	186
0.9	Nothing	12	546	12	933
	Net Flow	13	443	13	438

Table 3.7: Comparison of total S_N solves and the total number of NL-S_2 iterations to converge a scalar flux solution when using a sweeper with an S_2 sweep path to solve the NL-S_2 equations with source iteration for a problem with $\sigma_{t,0} = 0.0$ and $\sigma_{t,1} = 1.0$ with the scattering ratios shown and $q_{v,0} = 0.0$, $q_{v,1} = 1.0$. The coarse mesh used is shown in Figure 3.1 and the fine mesh has been refined uniformly twice.

Scattering Ratio	**Cycle Treatment**	**Outer Iterations Coarse Mesh**	**Inner Iterations Coarse Mesh**	**Outer Iterations Fine Mesh**	**Inner Iterations Fine Mesh**
0.1	Nothing	3	30	3	89
	Net Flow	3	12	3	19
0.2	Nothing	4	41	4	122
	Net Flow	4	19	4	26
0.4	Nothing	5	60	5	178
	Net Flow	5	35	5	43
0.8	Nothing	9	242	10	496
	Net Flow	9	183	10	188
0.9	Nothing	12	546	12	933
	Net Flow	13	442	13	441

Table 3.8: Comparison of total S_N solves and the total number of NL-S_2 iterations to converge a scalar flux solution when using a sweeper with an S_2 sweep path to solve the NL-S_2 equations using GMRES for a homogeneous problem with the scattering ratios shown and a uniform volumetric source. The coarse mesh used is shown in Figure 3.1 and the fine mesh has been refined uniformly twice.

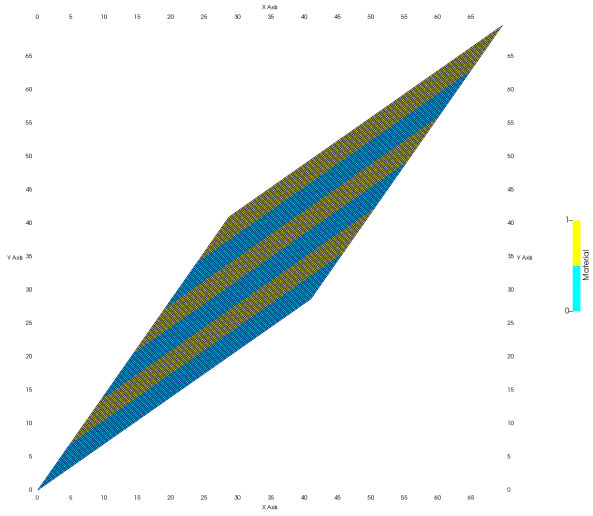
Total Cross Section	**Scattering Ratio**	**Cycle Treatment**	**Outer Iterations Coarse Mesh**	**Inner Iterations Coarse Mesh**	**Outer Iterations Fine Mesh**	**Inner Iterations Fine Mesh**
σ_t=0.1	0.1	Nothing	3	31	3	66
		Net Flow	3	19	3	28
	0.2	Nothing	3	35	4	80
		Net Flow	3	21	5	36
	0.4	Nothing	4	44	4	103
		Net Flow	4	26	4	40
	0.8	Nothing	6	69	6	163
		Net Flow	6	41	6	59
	0.9	Nothing	7	80	7	185
		Net Flow	7	47	7	66
	0.95	Nothing	7	86	7	199
		Net Flow	7	49	8	72
σ_t=5.0	0.1	Nothing	3	14	3	18
		Net Flow	3	12	3	12
	0.2	Nothing	3	16	3	20
		Net Flow	3	14	3	14
	0.4	Nothing	4	22	4	27
		Net Flow	4	19	4	20
	0.8	Nothing	5	46	5	59
		Net Flow	5	39	6	46
	0.9	Nothing	6	71	6	92
		Net Flow	7	64	6	66
	0.95	Nothing	7	116	6	136
		Net Flow	7	100	7	97

Table 3.9: Comparison of total S_N sweeps and the total number of NL-S_2 sweeps to converge a scalar flux solution when using a sweeper to solve the NL-S_2 equations using source iteration for a homogeneous problem with the scattering ratios shown and a uniform volumetric source. The coarse mesh used is shown in Figure 3.2 and the fine mesh has been refined uniformly once.

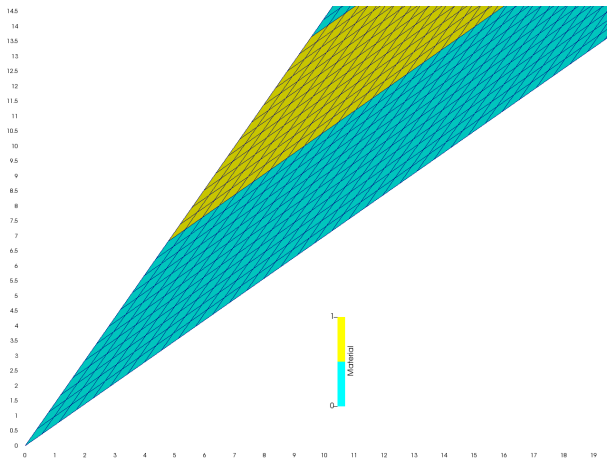
Total Cross Section	**Scattering Ratio**	**Cycle Treatment**	**Outer Iterations Coarse Mesh**	**Inner Iterations Coarse Mesh**	**Outer Iterations Fine Mesh**	**Inner Iterations Fine Mesh**
σ_t=0.1	0.1	Nothing	3	41	3	62
		Net Flow	3	12	3	13
	0.2	Nothing	3	49	3	77
		Net Flow	3	16	3	16
	0.4	Nothing	4	67	4	107
		Net Flow	4	24	4	24
	0.8	Nothing	5	109	5	168
		Net Flow	6	52	6	53
	0.9	Nothing	6	131	6	202
		Net Flow	6	64	6	64
	0.95	Nothing	6	140	6	215
		Net Flow	6	69	7	70
σ_t=1.0	0.1	Nothing	3	15	3	18
		Net Flow	3	12	3	12
	0.2	Nothing	3	20	3	23
		Net Flow	3	17	3	17
	0.4	Nothing	4	35	4	38
		Net Flow	4	31	4	31
	0.8	Nothing	5	155	5	164
		Net Flow	6	147	6	143
	0.9	Nothing	6	349	6	364
		Net Flow	6	327	7	321
	0.95	Nothing	6	729	6	750
		Net Flow	7	693	7	666

Table 3.10: Comparison of total S_N sweeps and the total number of NL-S_2 sweeps to converge a scalar flux solution when using a sweeper with an S_2 sweep path to solve the NL-S_2 equations with source iteration for a homogeneous problem with the scattering ratios shown and a uniform volumetric source. The coarse mesh used is shown in Figure 3.2 and the fine mesh has been refined uniformly once.

Total Cross Section	**Scattering Ratio**	**Cycle Treatment**	**Outer Iterations Coarse Mesh**	**Inner Iterations Coarse Mesh**	**Outer Iterations Fine Mesh**	**Inner Iterations Fine Mesh**
σ_t=0.1	0.1	Nothing	3	40	3	63
		Net Flow	3	18	3	24
	0.2	Nothing	3	49	3	77
		Net Flow	3	21	3	26
	0.4	Nothing	4	67	4	107
		Net Flow	4	27	4	33
	0.8	Nothing	5	111	5	168
		Net Flow	6	55	6	62
	0.9	Nothing	6	132	6	203
		Net Flow	6	66	6	72
	0.95	Nothing	6	141	6	217
		Net Flow	6	71	7	79
σ_t=1.0	0.1	Nothing	3	14	3	18
		Net Flow	3	12	3	12
	0.2	Nothing	3	20	3	23
		Net Flow	3	17	3	17
	0.4	Nothing	4	35	4	38
		Net Flow	4	31	4	31
	0.8	Nothing	5	155	5	164
		Net Flow	6	147	6	143
	0.9	Nothing	6	349	6	364
		Net Flow	6	327	7	321
	0.95	Nothing	6	729	6	750
		Net Flow	7	693	7	666



(a) Skewed structured mesh of 108×108 cells with edge length 50 cm and banded material configuration. The blue-green region (first from bottom) will be denoted as region 1 and the yellow region (second from bottom) will be denoted as region 2



(b) Zoomed image of the lower corner of the mesh shown in Figure 3.6a to show mesh detail.

Figure 3.6: Two-dimensional meshes investigated in this section

Table 3.11: Comparison of total S_N source iterations and the total number of NL-S_2 source iterations to converge a scalar flux solution when using a sweeper to solve the NL-S_2 equations with source iteration for a problem with $\sigma_{t,0} = 0.0$ and $\sigma_{t,1} = 1.0$ with the scattering ratios shown and $q_{v,0} = 0.0$, $q_{v,1} = 1.0$. The coarse mesh used is shown in Figure 3.6a and the fine mesh has been refined uniformly twice.

Scattering Ratio	**Cycle Treatment**	**Outer Iterations Sweeper**	**Inner Iterations Sweeper**	**Outer Iterations S_2 Path Sweeper**	**Inner Iterations S_2 Path Sweeper**
0.1	Nothing	4	152	4	158
	Net Flow	4	17	4	63
0.2	Nothing	5	193	5	194
	Net Flow	5	27	5	70
0.4	Nothing	6	256	6	271
	Net Flow	6	44	6	96
0.8	Nothing	9	634	10	639
	Net Flow	10	188	10	203

3.2 Acceleration on Structured Meshes

The investigations documented in this section use a mesh of cubes or squares and these meshes are potentially interesting to separate because, as discussed in Section 2.3.1, there will not be any cyclic dependencies between cells in the NL-S_2 system. The tests documented with a uniform mesh that has a banded material configuration as shown in Figure 3.8 are useful because they isolate the difficultly of strongly heterogeneous cross sections, specifically opaque regions adjacent to transparent regions, from the difficulty of cyclic dependencies in the linear NL-S_2 system. Note that the material configuration shown in Figure 3.8 contains significant voided region, in fact it is half void.

As discussed in Section 2.6, the computational performance of an algorithm such as the NL-S_2 acceleration investigated in this dissertation can depend significantly on details of the code implementation. Performance results based on cpu time are shown in Chapter 4 and it is reasonable to expect that the code could be made faster with future effort. To present results that are not as sensitive to specific code implementation details, the metric shown in Eq. 3.1 is used to describe acceleration speedup throughout this chapter and the next. In this equation, M is the number of directions in the S_N quadrature set. The number of S_N sweeps in the numerator is the number of sweeps to converge the scattering source when only the S_N or high order equation is used, that is there is no acceleration, and in the denominator, the S_N sweeps is the number required to update the average directions for the NL-S_2 solve. The value of 8 in denominator follows from there being 8 octants and thus 8 streaming plus collision equations to invert (while there are M such equations in the S_N system). Obviously in two-dimensions, symmetry can be exploited and only half of the directions actually need to be solved for. This applied to both the S_N equation and the NL-S_2 equation so Eq. 3.1 is simply written with M and 8 instead of $M/2$ and 4. In summary, this equation is meant to take the effort for the unaccelerated S_N solve and divide that by the effort for the accelerated solve. It is emphasized that this is an ideal speedup value which might be approached as the efficiency of the NL-S_2 code implementation approaches that of the S_N solver with an understanding that whether this ideal speedup can be reached depends on details such as the number

of directions and energy groups for the problem being solved in addition to code implementation details.

$$\text{ideal speed up} = \frac{\text{S}_n \text{ sweeps} \times \text{M}}{\text{NL-S}_2 \text{ sweeps} \times 8 + \text{S}_n \text{ sweep to update avg directions} \times \text{M}} \tag{3.1}$$

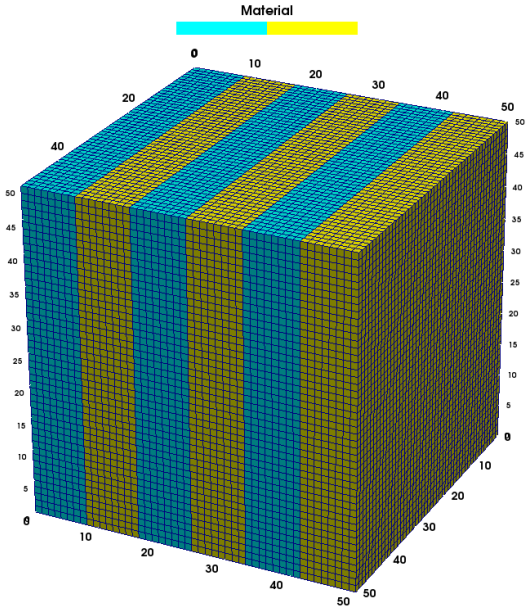


Figure 3.7: Structured three dimensional banded mesh

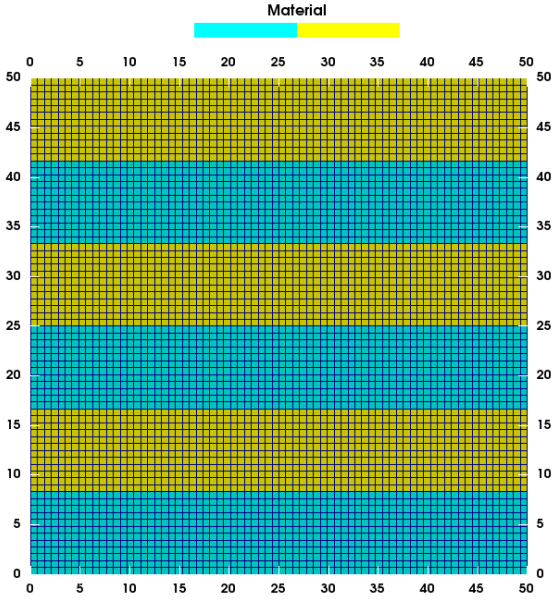


Figure 3.8: Structured mesh of 72×72 cells with edge length 50 cm and banded material configuration. The blue-green region (first from bottom) will be denoted as region 1 and the yellow region (second from bottom) will be denoted as region 2

3.2.1 Two-dimensional Mesh

This section presents NL-S_2 acceleration results for both homogeneous problems and a banded material configuration. With reference to Figure 3.8, one of the material bands is made a void, and the other material band some non-void with a specified σ_t, c (scattering ratio), and fixed volumetric source. Figure 3.9 shows acceleration results for three different quadrature sets and for four different convergence criteria on the inner iteration. The solid lines are results for a homogeneous problem and the dashed lines are for a heterogeneous banded material configuration where $\sigma_{t,1}$ and

c is the material for one band and other band is void. The inner iteration is how tightly the scattering source is converged for the NL-S_2 system before an S_N sweep is performed to update the average directions. The tolerance in this case means that the relative maximum difference of the scalar flux is reduced by the tolerance times the value on the first iteration. For the homogeneous problems with a uniform mesh, the solution should not change significantly as quadrature sets with more directions are used. The primary impact is obviously that the acceleration improves as the S_N sweep becomes more expensive.

A general trend shown in these plots is that looser converge of the inner iteration leads to better acceleration. What convergence tolerance leads to the best acceleration is of course problem dependent in general. The acceleration is not sensitive to the inner convergence criteria over a certain range, there is a small difference in the results for relative tolerance of 0.1 to 1.0e-3 while the tolerance of 1e-6 is significantly less efficient. Fine grain results for inner tolerance convergence are not presented, the purpose of presenting these results is to present evidence that the inner convergence tolerance does not need to be tuned to a precise value to achieve reasonable acceleration.

Acceleration is less effective for the banded material configuration than the homogeneous problem. Additionally, note that for some problems using the S_4 quadrature set (32 directions), an acceleration of zero is shown. This means that the acceleration did not converge. If two S_N sweeps are performed instead of one when computing average directions, then these cases do converge, however, the acceleration is still poor. NL-S_2 of acceleration of an S_N problem with only 32 directions is not of much practical interest, the more important result shown in Figure 3.8 is that NL-S_2 does still accelerate converge for a problem that contains significant voids and strong discontinuities in cross sections.

3.2.2 Three-dimensional Mesh

For homogeneous problems with three-dimensional uniform meshes, the same general trends discussed in the previous section for two dimensional meshes are again apparent. Specially, as the the number of directions in the S_N quadrature set is increased, the ideal acceleration of NL-S_2 acceleration is also increased. As such, only results for one S_N quadrature set containing 288

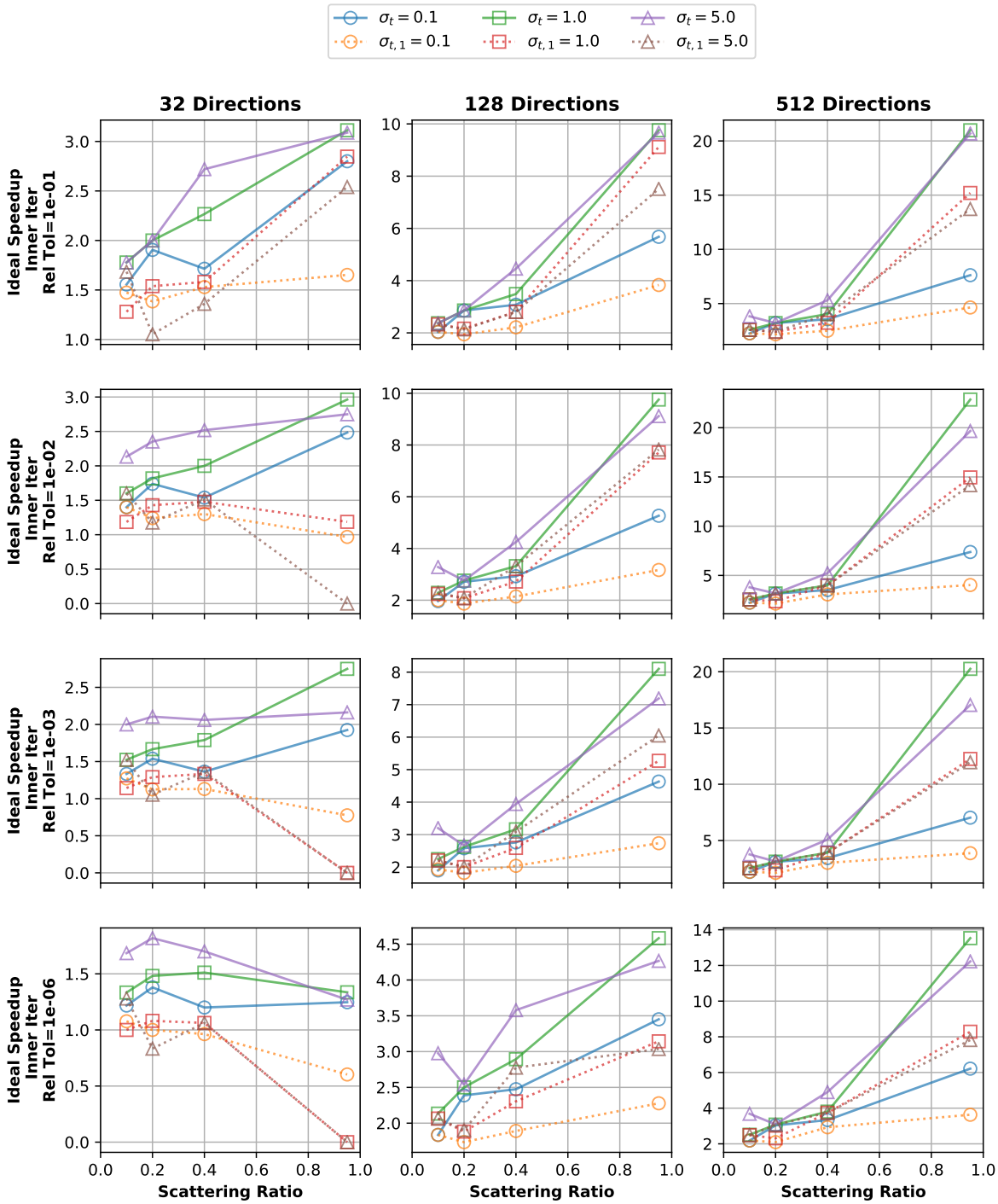


Figure 3.9: Ideal acceleration using the mesh in Figure 3.8 where homogeneous cross section results are shown with solid lines and a banded material configuration is shown with the dotted lines where one band has the cross section described and the other is a void. The meaning of Inner Rel Tol is described in Section 3.2.1

directions is shown in Figure 3.10.

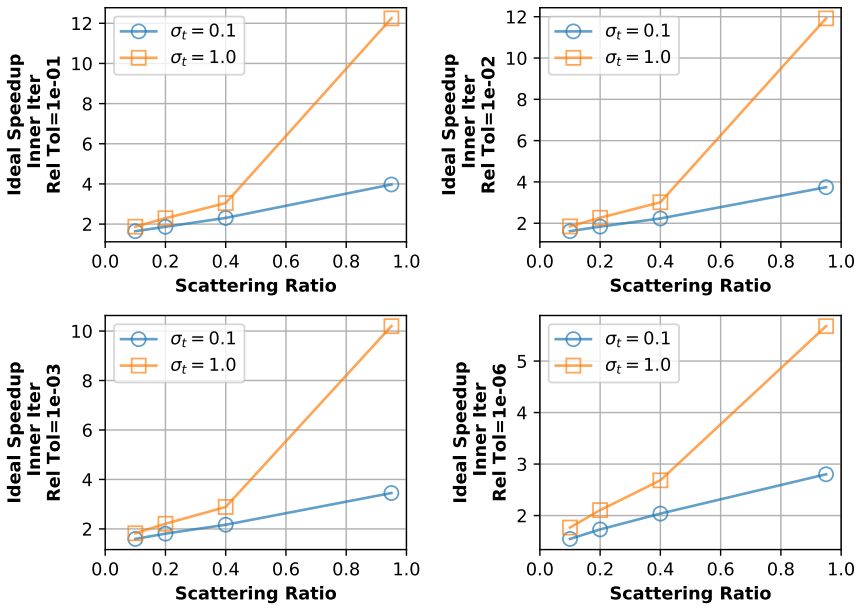


Figure 3.10: Ideal acceleration using the mesh in Figure 3.7 with homogeneous cross sections. The meaning of Inner Rel Tol is described in Section 3.2.1

3.3 Acceleration on Unstructured Meshes

Performance results are presented for NL-S_2 acceleration with an unstructured mesh in this section. Based on the results presented in Section 3.3, only the sweeper using the linear S_2 sweep path is investigated in this section. Additionally, all results are for calculations where the NL-S_2 system has been modified to use a net flow accross each face as described in Section 2.3.2.

3.3.1 Two-dimensional Mesh

Results for both a homogeneous problem and a problem with an opaque region and a void region are shown in Figure 3.11. With reference to Figure 3.1, region 1, that is the region outside the circle is opaque having the varying cross sections described in the plots and the region inside the circle, region 2 is a void. Compared to the results for a structured mesh, that is comparing with Figure 3.9, the ideal acceleration results are poorer. The sweep no longer directly inverts the

NL-S_2 streaming plus collision operator, but some data must be lagged and so a decrease in the effectiveness of the acceleration scheme is expected, although the decrease is small enough that the overall NL-S_2 acceleration is still effective. For the heterogeneous results shown in Figure 3.11 with the dotted lines, more cases have an acceleration of zero meaning that the acceleration scheme did not converge to a scalar flux. This convergence problem for these strongly heterogeneous cases can be avoided by performing more than one S_N sweep in the outer iteration, that is when the average directions are computed.

Figure 3.12 shows the same result as Figure 3.9 except that two S_N sweeps are used in the outer iteration where the average directions are computed instead of one. This adds some additional expense to the overall NL-S_2 acceleration scheme, but using two sweeps results in most test cases converging. If more sweeps are used, all cases can be made to converge, but the acceleration for these cases is obviously poor. The purpose of these results was to demonstrate that NL-S_2 acceleration can still be effective for problems with voids without needing any special changes to the algorithm. The test cases presented here are not exhaustive, but provide some evidence that NL-S_2 should be useful for accelerating more complex problems such as the k-eigenvalue tests presented in the next chapter and that using more than one sweep in the outer iteration may be useful for problems with heterogeneous materials.

3.3.2 Three-dimensional Mesh

For homogeneous problems with the three-dimensional extruded unstructured mesh, the same general trends discussed in the previous section for two dimensional meshes are again apparent. Specially, as the the number of directions in the S_N quadrature set is increased, the idea acceleration of NL-S_2 acceleration is also increased. As such, only results for one S_N quadrature set containing 288 directions is shown in Figure 3.13.

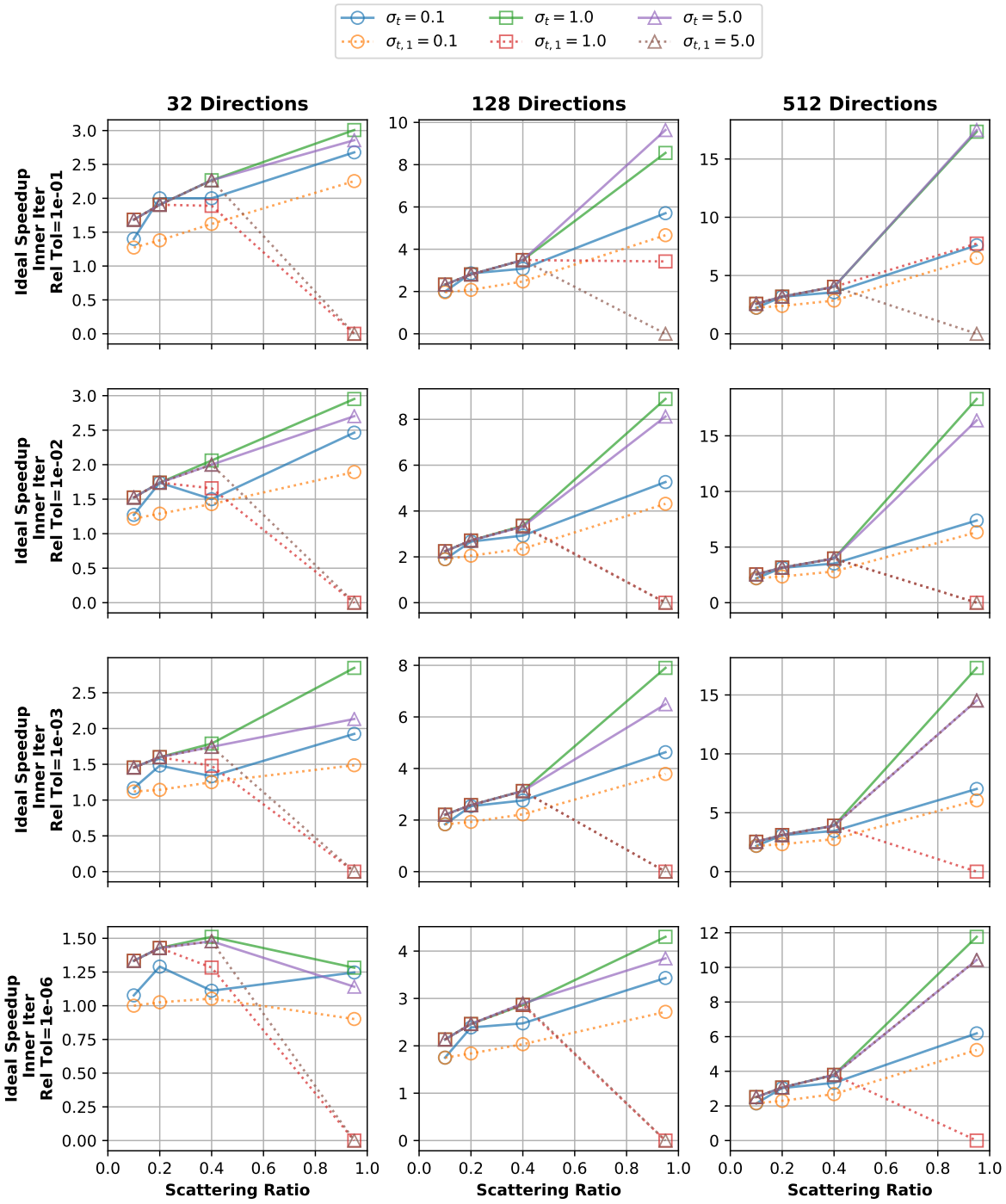


Figure 3.11: Ideal acceleration using the mesh in Figure 3.12 where homogeneous cross sections are shown with solid lines and a heterogeneous material configuration is shown with the dotted lines where inside the circile is a void and the region outside the circle has the cross section listed in the figure legend. The meaning of Inner Rel Tol is described in Section 3.2.1

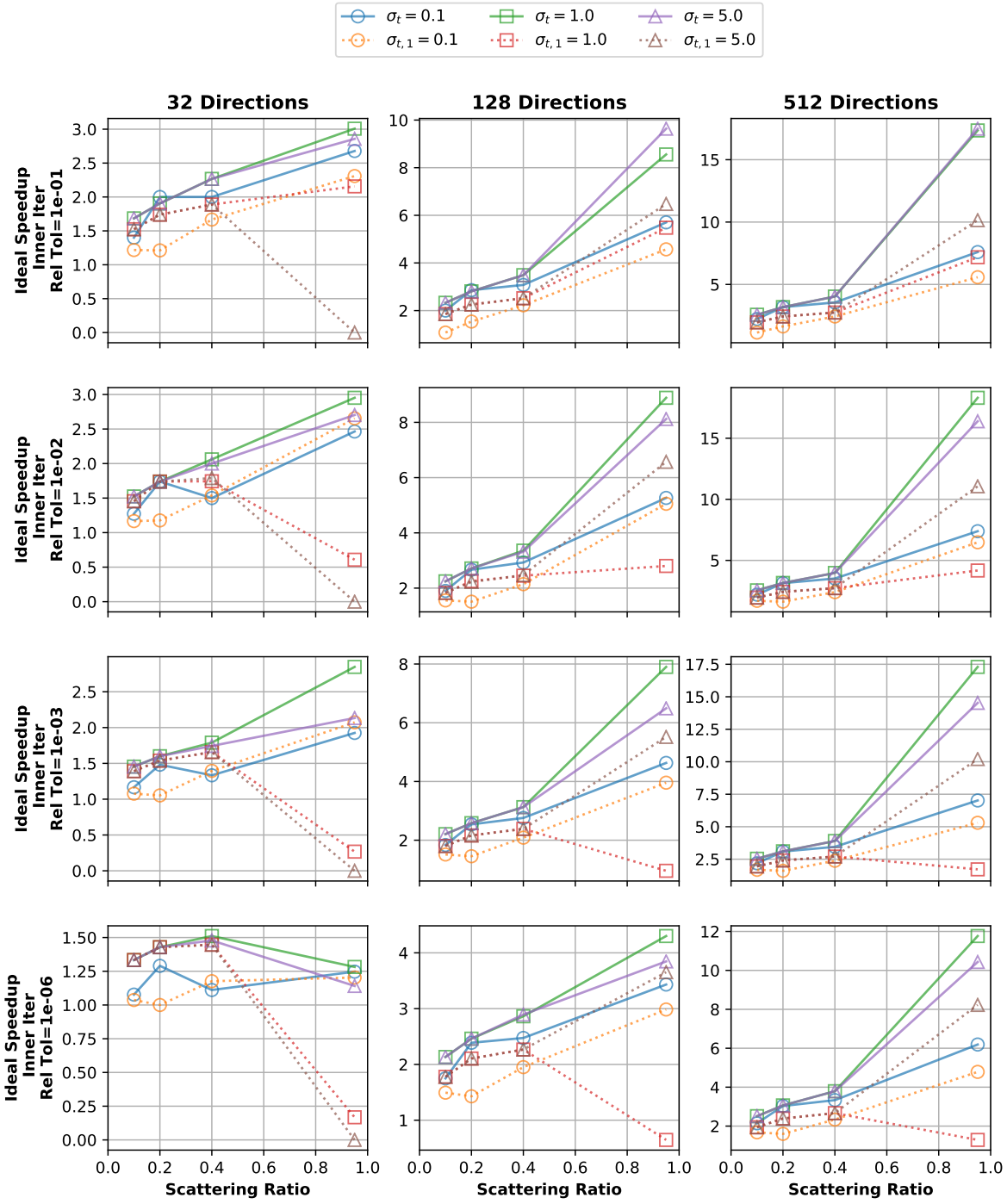


Figure 3.12: Ideal acceleration using the mesh in Figure 3.12 where homogeneous cross sections are shown with solid lines and a heterogeneous material configuration is shown with the dotted lines where inside the circile is a void and the region outside the circle has the cross section listed in the figure legend. Two S_N sweeps instead of one is used when computing the average direction for the heterogeneous results. The meaning of Inner Rel Tol is described in Section 3.2.1

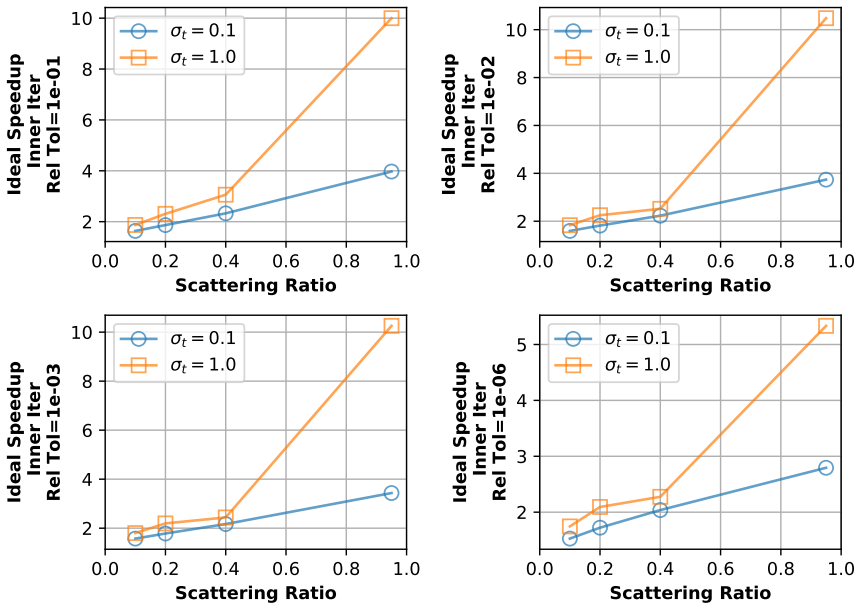


Figure 3.13: Ideal acceleration on an unstructured extruded mesh with homogeneous cross sections. The meaning of Inner Rel Tol is described in Section 3.2.1

4. NL-S_2 ACCELERATION OF K-EIGENVALUE CALCULATIONS

The previous chapter showed that NL-S_2 acceleration can be effective for fixed source problems with unstructured extruded three dimensional meshes, especially when the technique to reduce coupling between mesh cells in the NL-S_2 system, which is described in Section 2.3.2, is used. In this chapter, the additional complexity of an eigenvalue calculation is added. This chapter shows that the results and conclusions from the previous section are valid for the acceleration of k-eigenvalue calculations as well. K-eigenvalue calculations are reviewed in Section 1.4. The problems investigated in this section use considerably larger and more complicated unstructured meshes than in the previous chapter. The well known 2D C5G7 benchmark problem is investigated in this chapter as well as three other k-eigenvalue problems which are based on the C5G7 benchmark, but have been modified to test specific items like acceleration in the presence of anisotropic scattering and also voids. All results presented in this section use a guass-chebeychev product quadrature for the S_N equations. For quick reference, the S_6 quadrature set has 72 directions, S_8 has 128, S_{10} has 200, and S_{12} has 288 directions.

4.1 Two-Dimensional C5G7 Benchmark

The C5G7 benchmark is a popular problem for testing the effectiveness of a method for heterogeneous lattice calculations. C5 refers to the fifth test configuration proposed in [39] and later a 7 energy group problem based on this configuration was proposed in [40]. The benchmark problem has only isotropic scattering and the multigroup cross sections are provided. Although the problem is heterogeneous, diffusion based acceleration techniques have been shown to be effective for this problem. The benchmark is based on light water reactor assemblies and thus the scalar flux is generally diffusive and there are no voids to potentially cause difficulties implementing a diffusion based acceleration method.

Although it is expected that diffusion based methods can be implemented so that they are faster than NL-S_2, there are several reasons the C5G7 benchmark is still a useful test to present

NL-S_2 results for. Firstly, the ability to use a sweep solver for the low order system may still be useful for problems similar to the C5G7 benchmark in a code where an efficient diffusion based solver has not been implemented. Secondly, this is a well known multigroup problem with cross sections already developed. Finally, this heterogeneous four assembly problem is meshed with an unstructured meshing algorithm (using the gmsh program) and so this is a good test of NL-S_2 on an unstructured mesh. The previous chapter included investigations of fixed source acceleration on unstructured meshes, but the meshes included fewer elements than those investigated in this chapter. The C5G7 mesh used in this section is unstructured and contains 454,491 quadrilaterals. The results in this section indicate that a sweep solver is still effective for a considerably more complex mesh than was investigated in the previous chapter. The material configuration for the benchmark is shown in Figure 4.1. The mesh itself is too refined to show the detail in the figure.

Results for the S_N k-eigenvalue value solution are presented in Table 4.1. The eigenvalue was computed by converging the relative change of successive iterations to within 1e-6. Power iteration was used with GMRES to solve the transport equation with a lagged fission source, that is the inner iteration as shown in Eq. 1.10. The scattering source was not fully converged between power iterations, but the normalized residual was decreased by two orders of magnitude between each update to the lagged k-eigenvalue. The total number of Krylov iterations indicates the total number of times a solution is approximated in a Krylov space and so gives an indication of, for example, how many times vectors are orthogonalized. As discussed in Section 1.2, the implementation of GMRES involves sweeps to compute the action of an operator. Each iteration, a sweep is performed and in addition to this, each time an iteration is started to converge a scattering source, a sweep is performed to computed a right hand side as shown in Eq. 1.5. The reference [41] MCNP k-eigenvalue for this benchmark is 1.18655 which is only 30 pcm different from the S12 result shown in Table 4.1. This indicates that mesh used in this Section is refined enough to produce an accurate answer.

The NL-S_2 acceleration of the k-eigenvalue calculation was implemented in a relatively simple way. There are three levels of iteration, an outer iteration where one or more S_N sweeps are

performed to compute average directions and an inner iteration consisting of two levels, the outer of these levels being where the k-eigenvalue is computed and the inner iteration being where the scattering source is converged for a fixed k-eigenvalue. Note that when performing the S_N sweep, the eigenvalue is lagged at the previous value calculated in the inner NL-S_2 iterations or at the initial guess during the start of the calculation. Simple power iteration is again used to compute a k-eigenvalue. The methodology used for convergence criteria is as follows. After an outer iteration where the S_N sweep is performed and average directions computed, inner iterations are performed until the difference between successive k-eigenvalue iterations is reduced by one order of magnitude compared to the first difference. On the inner most iteration, where the NL-S_2 scattering source is converged for a lagged k-eigenvalue, the GMRES solve is performed until the relative residual is reduced by two orders of magnitude after which a new k-eigenvalue iterate is computed. For example, after the S_N sweep has been performed, the NL-S_2 iterations begin with the previous k-eigenvalue, call this iterate k^0. With this k-eigenvalue, the scattering source is converged to the level described previously and then a new k-eigenvalue iterate is computed, call this k^1. A relative difference is computed $k^0_{diff} = \frac{abs(k^0 - k^1)}{k^1}$. This iterative scheme is repeated until $k^n_{diff} < 0.1 * k^0_{diff}$ after which time the outer most iteration begins again by performing one or more S_N sweeps to compute updated average directions. The iterations are terminated when the relative difference between successive eigenvalue iterates is less than 1e-7. This iterative scheme is simple and likely not optimal. But if a better criteria is implemented for determining levels of convergence for each iteration level, the acceleration results presented in this chapter should only be improved.

Results for the performance of NL-S_2 acceleration are shown in Table 4.2 and Table 4.3. The difference between these two results is that for those shown in Table 4.2, two S_N sweeps are used on the outer iteration while for the results in Table 4.3, only a single sweep is used. The acceleration results when using two sweeps in the outer iteration are more stable while the acceleration when using only one sweep shows degradation when more directions are used in the S_N quadrature set. The reported ideal speedup was computed using Eq. 3.1 which is based only on the number of

NL-S_2 sweeps compared to the S_N sweeps. Some reasons why the actual speedup from the NL-S_2 code is less than the ideal speedup are discussed in Section 2.6.

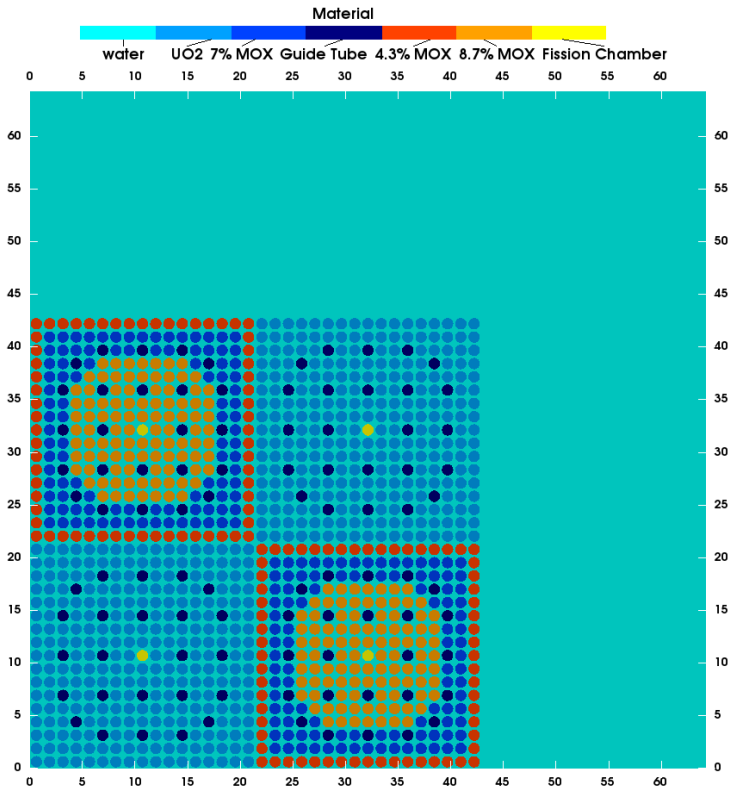


Figure 4.1: Image showing the C5G7 material configuration. Note that the left edge (x=0 axis) and bottom edge (y=0 axis) are reflecting boundaries.

Table 4.1: Eigenvalue results for the unaccelerated S_N solver

Quadrature	**k-eigenvalue**	**Total S_N Sweeps**	**Krylov Iterations**
S6	1.185820	481	424
S8	1.18605	471	415
S10	1.18706	471	415
S12	1.18685	470	414

Table 4.2: Eigenvalue and performance for the NL-S_2 system when using two S_N sweeps in the outer iteration

Quadrature	NLS2 k-eigenvalue	S_N Sweeps	NLS2 Sweeps	Krylov Iterations	Ideal Speedup	Actual Speedup
S6	1.185842	14	886	785	2.16	1.67
S8	1.186073	14	1028	913	3.67	2.23
S10	1.187077	14	1041	925	5.96	2.98
S12	1.186880	14	1064	945	8.31	3.86

Table 4.3: Eigenvalue and performance for the NL-S_2 system when using one S_N sweep in the outer iteration

Quadrature	NLS2 k-eigenvalue	S_N Sweeps	NLS2 Sweeps	Krylov Iterations	Ideal Speedup	Actual Speedup
S6	1.185810	6	554	487	3.53	2.58
S8	1.185990	9	698	613	5.44	3.12
S10	1.187080	21	1122	973	5.17	2.51
S12	1.186870	35	1572	1345	4.80	2.15

4.2 Three Dimensional Single Assembly Test Problem

A three dimensional test problem is presented in this section which has similarities to the three-dimensional C5G7 benchmark, but uses a single assembly and is shorter overall than the benchmark problem. This smaller problem is used instead of the full three dimensional C5G7 benchmark so that the problem can be run on a small number of processors, specifically not more than several hundred. Figure 4.4 shows the material configuration of the test problem investigated in this section. The single assembly problem uses the same MOX assembly from the C5G7 benchmark. Additionally, this problem is also topped with water and has three reflecting boundaries similar to the three dimensional C5G7 benchmark. The geometry is 100 cm tall with the top 10 cm of this consisting of water. The two dimensional mesh structure shown Figure 4.2 was extruded into 147 layers and the three-dimensional extruded mesh used consists of 5,380,000 hexahedrals along with 17,640 prisms. It is noted also that the two dimensional mesh structure shown Figure 4.2 was

simply extruded for the full 100 cm height of the test problem, that is into the water region were all mesh cells were simply set to water. The 147 levels are shown in Figure 4.3 to help visual the level of mesh refinement. Note the top of the problem, the water region, has slightly taller cells which was done so more refinement was available where the solution is changing more rapidly and fission is occurring.

The calculation was performed in parallel since the large problem size makes running on a single processor impractical. Figure 4.2 shows an x-y cut of the three dimensional mesh tested in this section. The apparent construction lines in this image which form a 2x5 grid are the partitioning lines. As discussed in Section 2.2.1, there are in general several possible ways to setup a parallel computation when using a parallel sweeper. To avoid the complication of having to lag angular flux solutions at processor boundaries, a structured partitioning is used along with the unstructured mesh as shown in the image. The lack of balance between the processor domains is a result of simplifying the process of creating the mesh, the partitioning cut lines do not pass through fuel rods, only between them. Each processor was given 7 of the 147 layers stacked on one region from the 2x5 grid shown in Figure 4.2 requiring the problem to be run on 210 processors.

As discussed in Section 2.2.1, the scalability of a parallel sweep solve for the NL-S_2 system will be worse because the S_N system will generally have many more directions than the NL-S_2 system. However, parallel implementations of sweeper to solve the NL-S_2 system are not meant to be a focus of this dissertation and the parallel implementation of the NL-S_2 is likely not ideal. As such, results are presented for only two different parallel implementations of the NL-S_2 solver. First, a traditional S_N style sweeper where the processor domains wait to get the most recent solution data was implemented and tested, that is something equivalent to a forward substitution of the block lower triangular matrix where the blocks are the processor domains. Note this sweep produces the exact same iterative convergence rate for the scattering source and fission source as if the problem were solved in serial. Secondly, the parallel sweep that is like a processor domain-wise block jacobi iteration was also implemented and tested. This type of parallel sweeper was reviewed briefly in Section 2.2.1. To quickly summarize the important point, flux information is

simply lagged at processor boundaries and so each processor can begin calculations immediately without waiting on data from its neighbor, the trade off for the increased parallel efficiency being that the convergence rates for the scattering source will decrease as more processors are used.

An S_N k-eigenvalue solution was calculated using the same procedure described in the previous section. The S_N solution as well at information about total effort to converge to a solution is shown in Table 4.4. Cross sectional cuts through the three dimensional problem showing the scalar flux solution for the each of the 7 energy groups are shown in Figures 4.5 through 4.11.

The NL-S_2 solution results for the sweeper lagging no data at processor boundaries is shown in Table 4.5. The ideal speedup result for this three-dimensional problem are considerably better than the results shown in Table 4.2 for the two-dimensional C5G7 problem. This is partly do to the slower convergence of the S_N solve only in three-dimensions. Considerably more power iterations are required to converge a k-eigenvalue for the three dimensional problem than the two-dimensional problem and the impact of this can be seen in the large number of S_N sweeps taken to converge to a solution in Table 4.4. The relatively low number of NL-S_2 sweeps taken, similar to the number taken in two-dimensions, indicates that the sweeper is still a highly effective linear solver for the NL-S_2 system in three-dimensions and the overall acceleration capability does not degrade. However, the sweeper, at least as currently implemented, is not an effective way of solving the NL-S_2 equation in parallel. Although the ideal speedup is over 15, the actual speedup is less than 1 meaning that the NL-S_2 calculation actually took longer than the S_N solve. However, the high ideal speedup and good iterative performance of the sweeper for the NL-S_2 equations motivates future work to find more scalable ways of solving the NL-S_2 equations. Table 4.6 shows NL-S_2 acceleration results using the processor domain-wise block jacobi style sweeper. As expected, the number of NL-S_2 sweeps increases significantly. Even for this problem run on only 210 processors, four times more sweeps are required with this style sweeper. The ideal speedup is considerably lower and the although the actual speed is improved.

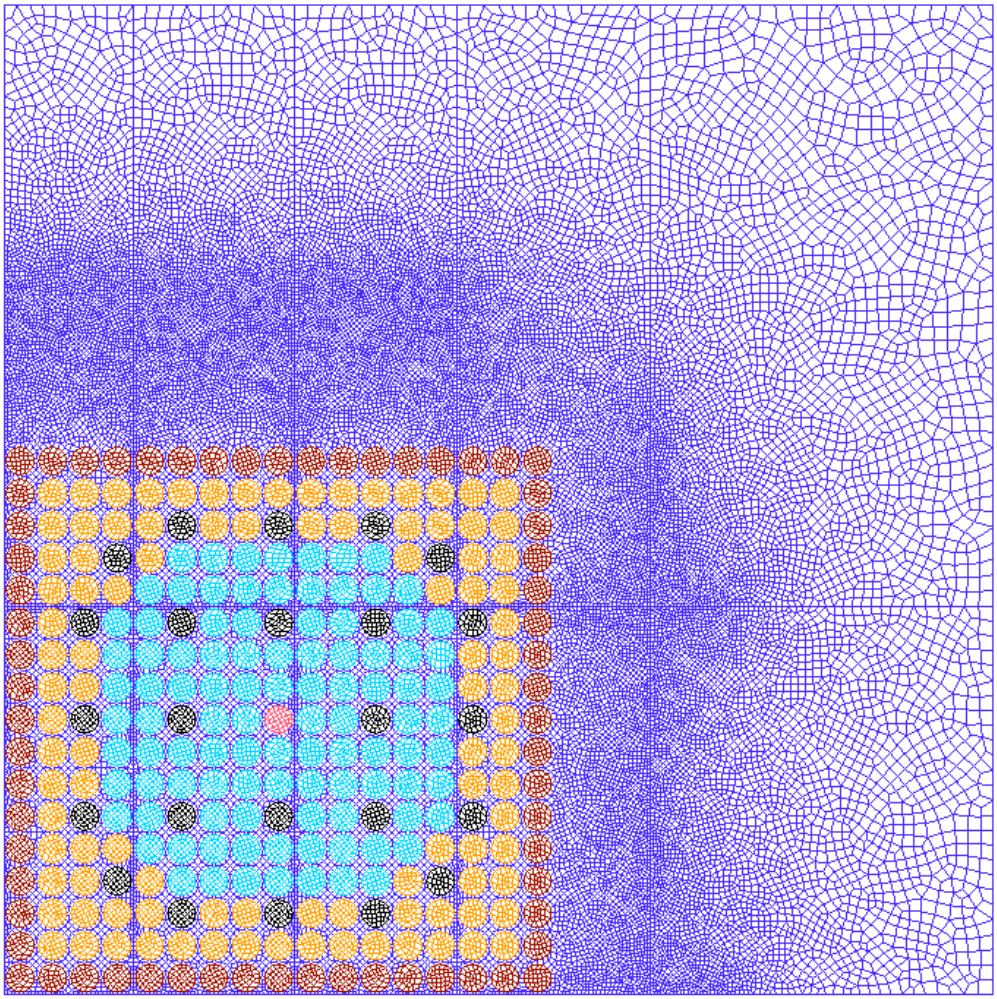


Figure 4.2: x-y plane of the three-dimensional test problem showing the mesh detail which is extruded and the structured mesh partitioning used to make the distributed mesh for parallel computation. The different colors show different material regions.

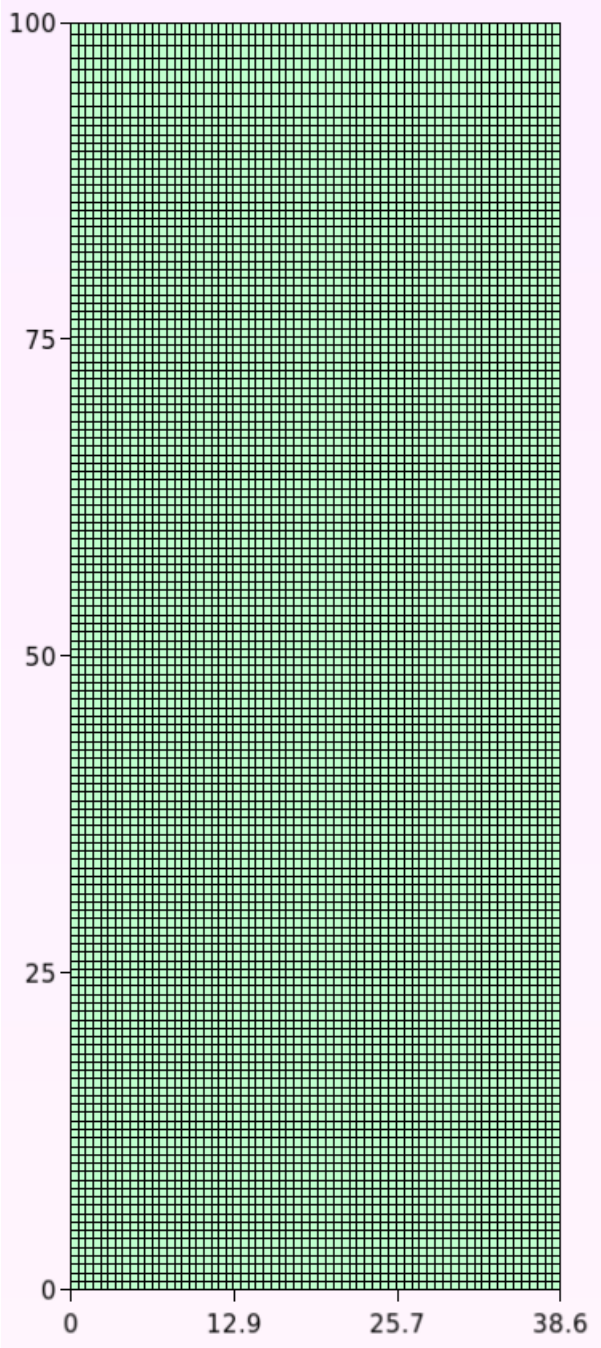


Figure 4.3: x-z plane of the three-dimensional test problem showing the extruded level of refinement.

Table 4.4: Eigenvalue results for the unaccelerated S_N solver

Quadrature	**k-eigenvalue**	**Total S_N Sweeps**	**Krylov Iterations**
S12	0.953244	766	685

Table 4.5: Eigenvalue and performance for the NL-S_2 system when using the correct sweep ordering among processors, that is the same style of sweeper used for the S_N system

S_N Sweeps Outer Iteration	**NLS2 k-eigenvalue**	**S_N Sweeps**	**NLS2 Sweeps**	**Krylov Iterations**	**Ideal Speedup**	**Actual Speedup**
2	0.953373	16	1048	949	16.98	0.72
3	0.953354	18	935	847	17.42	0.81

Table 4.6: Eigenvalue and performance for the NL-S_2system for the S12 S_N quadrature shown for both two and three S_N sweeps per outer iteration when using a domain wise block-jacobi style parallel implementation

S_N Sweeps Outer Iteration	**NLS2 k-eigenvalue**	**S_N Sweeps**	**NLS2 Sweeps**	**Krylov Iterations**	**Ideal Speedup**	**Actual Speedup**
2	0.953397	14	3955	3844	6.347	2.03
3	0.953399	18	3913	3803	5.854	1.95

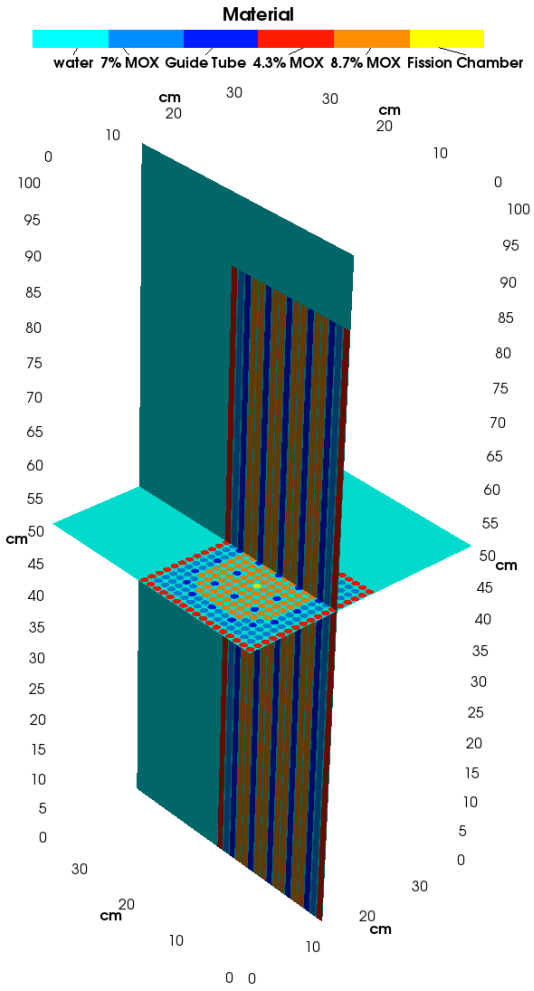


Figure 4.4: Material configuration for the three-dimensional problem investigated in this section shown as a x-y plane cut and an y-z plane cut. The single assembly has the same materials and layout as the MOX assembly in the C5G7 benchmark. Three of the boundaries are set to reflecting conditions, the other vacuum

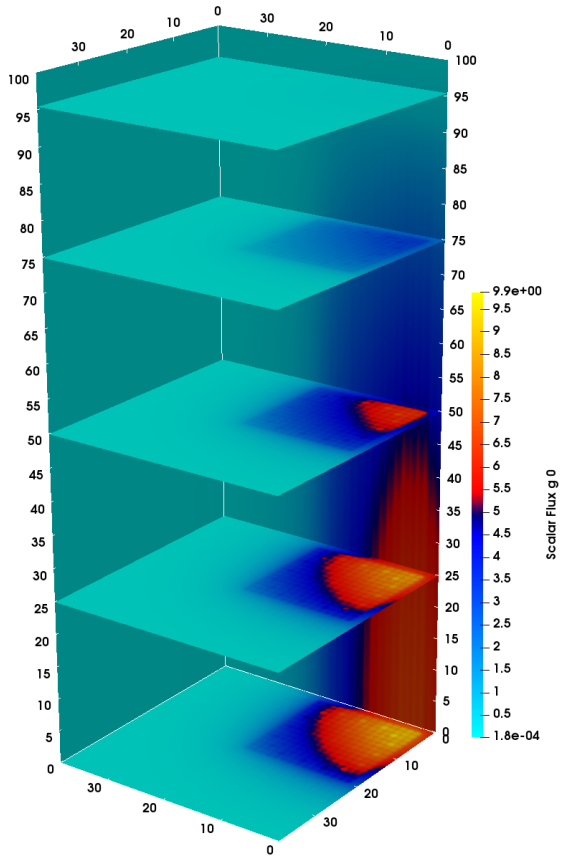


Figure 4.5: Scalar flux solution for group 0 shown as x-y cuts at several elevations

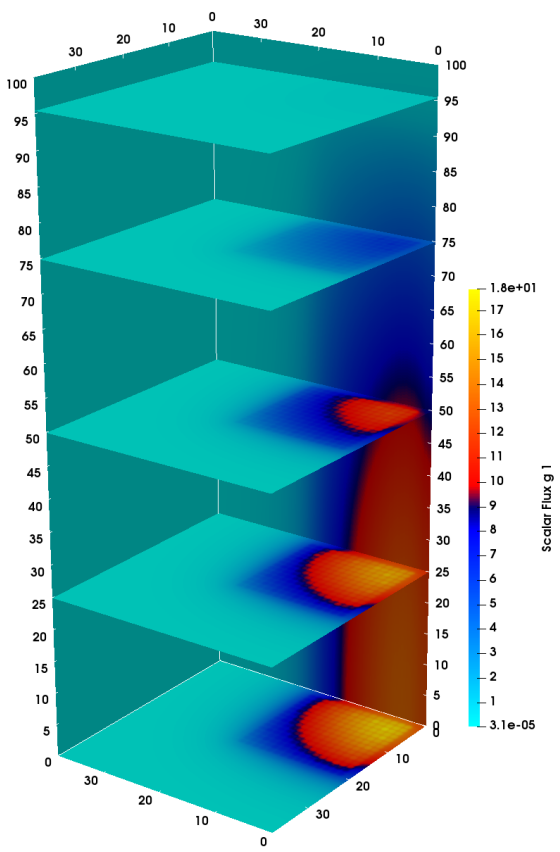


Figure 4.6: Scalar flux solution for group 1 shown as x-y cuts at several elevations

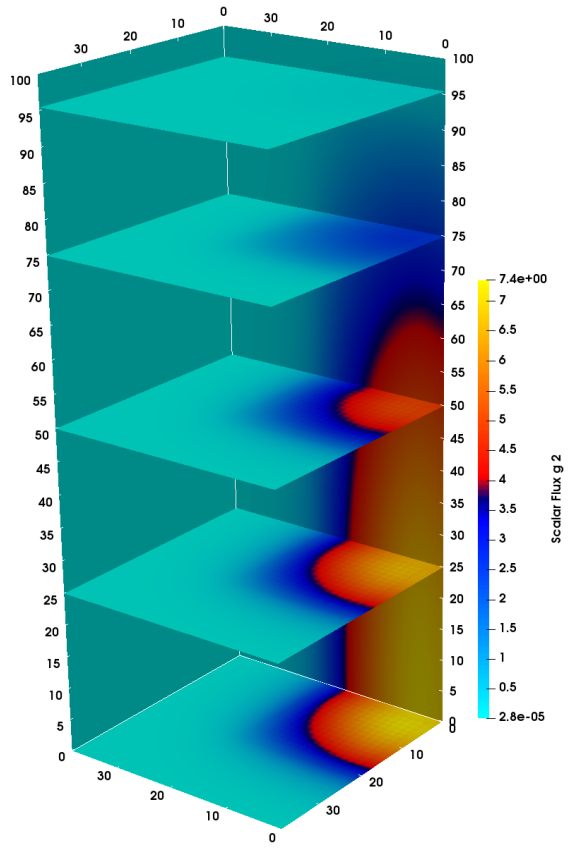


Figure 4.7: Scalar flux solution for group 2 shown as x-y cuts at several elevations

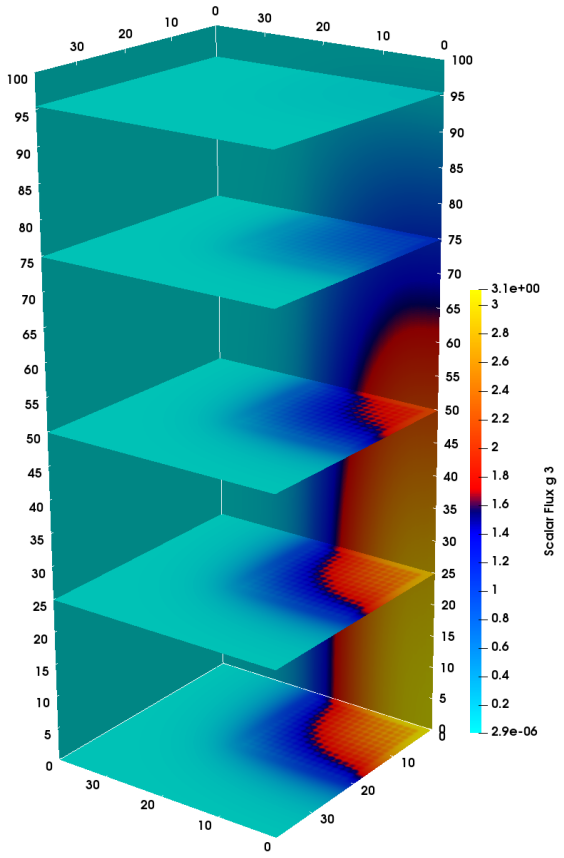


Figure 4.8: Scalar flux solution for group 3 shown as x-y cuts at several elevations

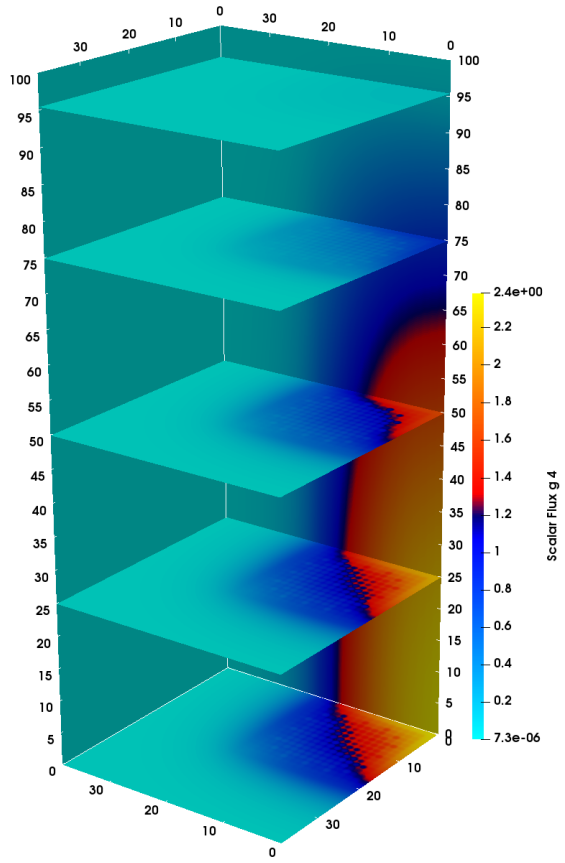


Figure 4.9: Scalar flux solution for group 4 shown as x-y cuts at several elevations

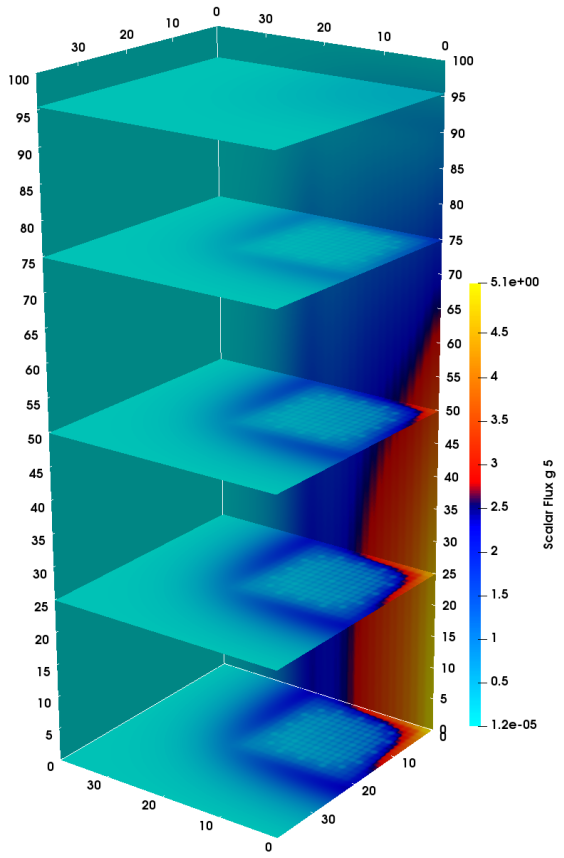


Figure 4.10: Scalar flux solution for group 5 shown as x-y cuts at several elevations

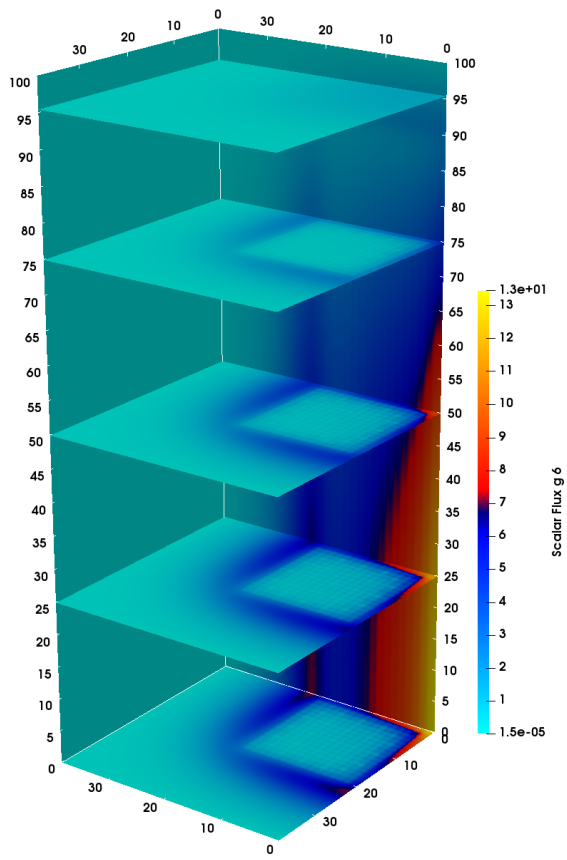


Figure 4.11: Scalar flux solution for group 6 shown as x-y cuts at several elevations

4.3 C5G7 with Anisotropic Scattering Modification

Acceleration of anisotropic scattering with NL-S_2 is discussed in Section 1.5 where a simple method is described that involves accelerating only the scalar flux. The effectiveness of such a technique will of course decrease as higher order scattering terms become more important. In this section, NL-S_2 is investigated for a k-eigenvalue problem with anisotropic scattering. The problem investigated in this section is meant to be representative of the amount of anisotropic scattering which would be encountered in reactor applications.

The same C5G7 problem described in Section 4.1 is again used here except that the cross section for water has been modified to included anisotropic scattering. The multigroup energy boundaries were taken from Table 12 in [42] and NJOY was used to develop the scattering cross

sections. P3 scattering was used with weighting spectrum option 5, that is the epri-cell lwr spectrum. The mesh used for the test in this section was coarser than that used in the C5G7 benchmark section this time consisting of 145,799 quadrilaterals. The same problem was using the deterministic transport code Griffin to verify the correctness of the k-eigenvalue calculation in chi-tech with anisotropic scattering (the correct 2D C5G7 benchmark result already verified the code related to k-eigenvalue calculations for problems with isotropic scattering). The same mesh was used, but the linear discretization used was bilinear instead of the PWLD discretization used to generate the chi-tech results. Simple power iteration was used with the parallel sweeper in Griffin [43]. The computed k-eigenvalue computed from Griffin was the same for the same S_8 quadrature used to generate the corresponding chi-tech result.

The same iterative procedures described in Section 4.1 for calculating a k-eigenvalue with power iteration were again used in here for this problem with anisotropic scattering except that the anisotropic contribution to the scattering source was held constant during the NL-S_2 power iterations and the higher flux moments beyond the scalar flux were updated only on the outer most iteration where the S_N sweep is performed to update average directions. The S_N results are shown in Table 4.7 and the corresponding NL-S_2 results are shown in Table 4.8. The point of the results in this section is simply do demonstrate that using a relatively simple technique, NL-S_2 can still be used to accelerate problems with a representative level of anisotropic scattering.

Table 4.7: Eigenvalue results for the unaccelerated S_N solver

Quadrature	**k-eigenvalue**	**Total S_N Sweeps**	**Krylov Iterations**
S8	1.240960	842	799
S10	1.241569	820	778
S12	1.241531	1115	1056

Table 4.8: Eigenvalue and performance for the NL-S_2 system for the problem with anisotropic scattering and using two S_N sweeps on the outer most iteration

S_N Quadrature	NLS2 k-eigenvalue	S_N Sweeps	NLS2 Sweeps	Krylov Iterations	Ideal Speedup	Actual Speedup
S8	1.240962	26	3505	3352	3.63	1.17
S10	1.241572	26	3354	3208	5.57	1.71
S12	1.241530	26	3358	3212	10.49	3.30

4.4 C5G7 Based Problem with Void

In this section, a pincell type problem with a large void in the center is investigated. The pincell is based on the C5G7 configuration. A 6x6 grid of UO2 pins from the C5G7 is taken and the center pin is replaced with a square void as shown in Figure 4.12. A similar test problem was constructed in [44]. The impact of reflecting boundaries on the sweeper was not discussed before this point, but is relevant for the pincell problem. For the pin cell problem depicted in Figure 4.12 all of the boundaries are reflecting, the presence of reflecting boundaries does impact the convergence of the solver because all boundaries are reflecting. To perform the transport sweep, angular flux information must be lagged at some boundary. The determination of what boundary information to lag is a graph problem, the same idea discussed in Section 2.2.2 where edges must be removed for the graph to make is acyclic is relevant. Since boundary information is lagged, the streaming plus collision operators are no longer exactly inverted by the sweeper. The specific mesh used to generate the results in this section is not that shown in Figure 4.12, but a version of this mesh that was refined uniformly three times. The mesh used contained 165,632 quadrilaterals.

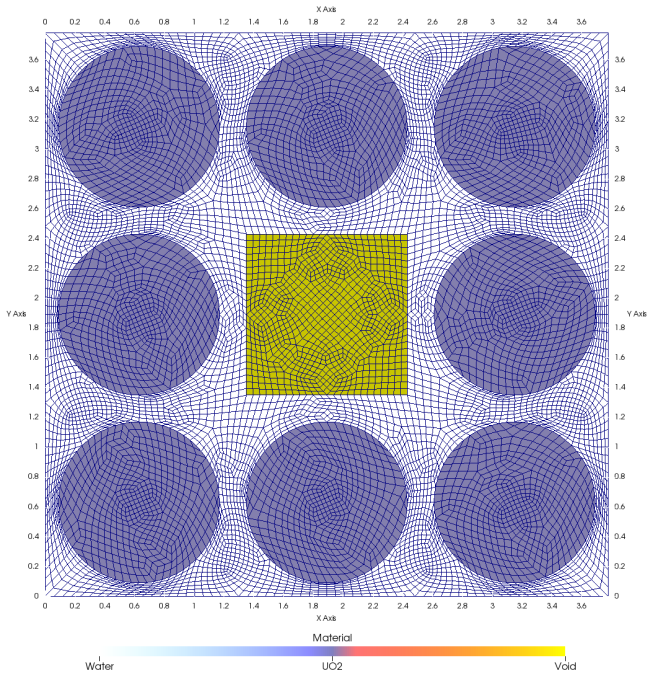


Figure 4.12: A modified C5G7 like pin cell problem with a square void in the middle. The water and UO2 cross sections are those from the isotropic C5G7 benchmark. All boundaries and reflecting. Mesh shown is a coarse version of the mesh used so that mesh detail in visible in image.

This same problem was run with Griffin and as mentioned in the section presenting C5G7 anisotropic scattering results, bilinear discretization was used with Griffin instead of the PWLD discretization. The same k-eigenvalue was calculated with Griffin as chi-tech. k-eigenvalue calculation results for the S_N solver are shown in Table 4.9. Compared to the results shown for two-dimensional C5G7 or the three-dimensional C5G7 like problem, far fewer power iterations are required to converge a k-eigenvalue for this problem and this is reflected in the small number of S_N sweeps shown in Table 4.9.

The NL-S_2 acceleration results are shown in Table 4.10 and Table 4.11 for two S_N sweeps and three S_N sweeps used on the outer iteration respectively. For this problem, using only one sweep on the outer iteration results in poor convergence, specially the acceleration did not converge in a reasonable number of iterations. However, the results presented in the two tables indicates that when more than one sweep is used on the outer iteration, the NL-S_2 acceleration is not signifi-

cantly affected by the void. The ideal speedups shown in these tables are low, however, this is simply the result of the problem being easy to solve with the S_N solver. That is few iterations are required with simply using power iteration with the S_N sweeper only. If the problem required more power iterations to converge, the NL-S_2 acceleration effectiveness should increase. The scalar flux solution from the S_N solve with S_{12} quadrature is shown in Figures 4.13 through 4.19.

Table 4.9: Eigenvalue results for the unaccelerated S_N solver

Quadrature	**k-eigenvalue**	**Total S_N Sweeps**	**Krylov Iterations**
S6	1.33235	72	67
S8	1.33222	70	65
S10	1.33321	73	68
S12	1.33282	72	67

Table 4.10: Eigenvalue for the NL-S_2 system with 2 sweeps per NL-S_2 iteration

S_N Quadrature	**NLS2 k-eigenvalue**	**S_N Sweeps**	**NLS2 Sweeps**	**Krylov Iterations**	**Ideal Speedup**	**Actual Speedup**
S6	1.33235	18	409	361	1.28	0.57
S8	1.3222	18	406	359	1.95	0.82
S10	1.33321	16	352	312	2.55	1.19
S12	1.33282	18	407	360	2.89	1.40

Table 4.11: Eigenvalue for the NL-S_2 system with 3 sweeps per NL-S_2 iteration

S_N Quadrature	**NLS2 k-eigenvalue**	**S_N Sweeps**	**NLS2 Sweeps**	**Krylov Iterations**	**Ideal Speedup**	**Actual Speedup**
S6	1.33235	27	351	315	1.26	0.60
S8	1.3222	27	354	318	1.65	0.85
S10	1.33321	27	352	316	2.32	1.25
S12	1.33282	27	354	318	2.54	1.42

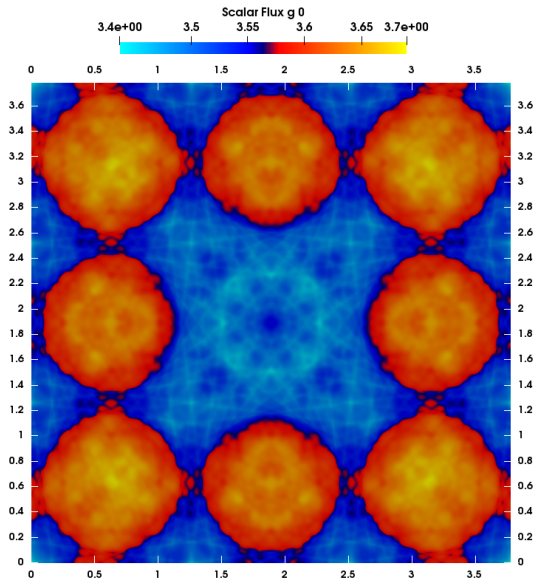


Figure 4.13: Scalar flux solution for group 0

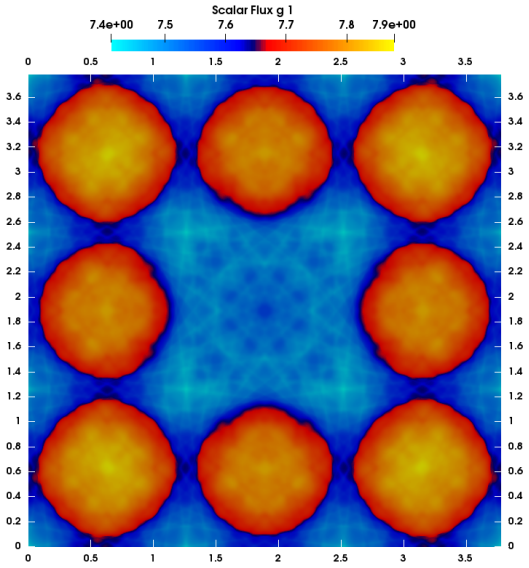


Figure 4.14: Scalar flux solution for group 1

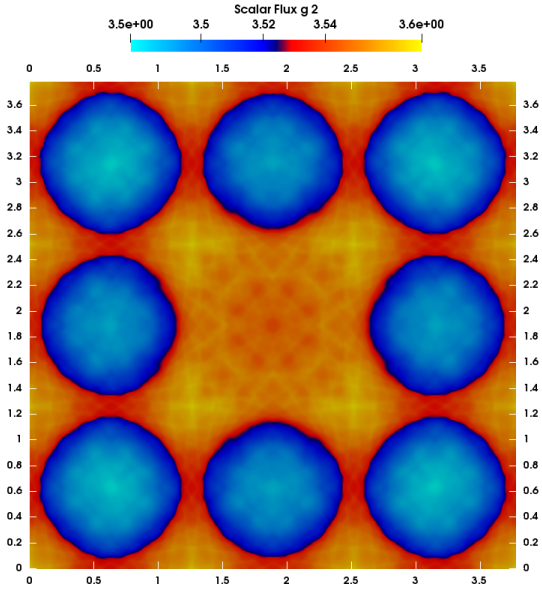


Figure 4.15: Scalar flux solution for group 2

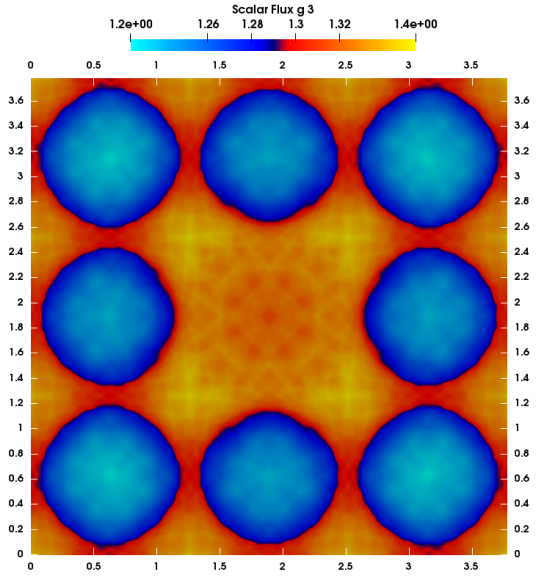


Figure 4.16: Scalar flux solution for group 3

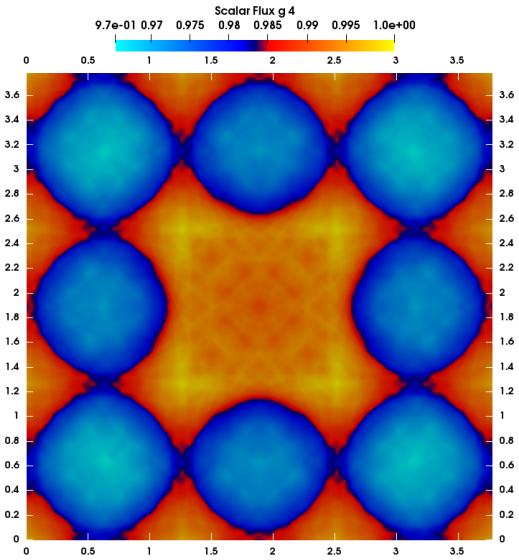


Figure 4.17: Scalar flux solution for group 4

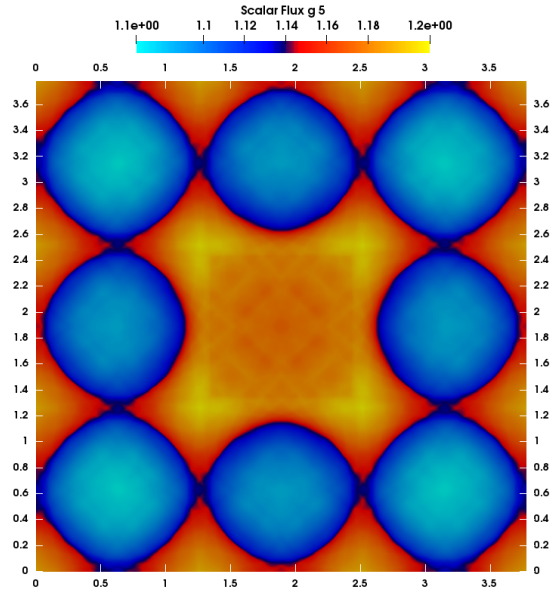


Figure 4.18: Scalar flux solution for group 5

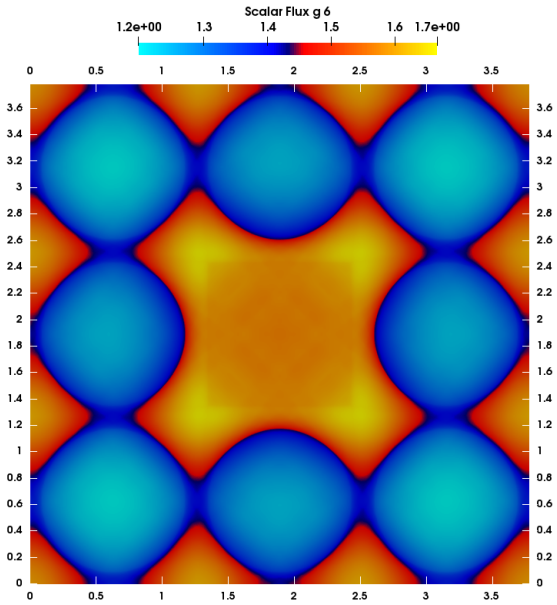


Figure 4.19: Scalar flux solution for group 6

5. CONCLUSIONS AND FUTURE WORK

The NL-S_2 equations investigated in this dissertation have a relatively simple formulation since they are consistently discretized and derived by integrating the discretized S_N equations over partial ranges of the unit sphere. A big potential benefit to NL-S_2 acceleration is that an S_N style sweeper might be effective for inverting the NL-S_2 "streaming plus collision" operator. However, the NL-S_2 system, while looking similar in some ways to an S_N system with S_2 quadrature has some significant differences. For any mesh other than one consisting entirely of rectangles or rectangular cuboids, the NL-S_2 system will have many cyclic dependencies coupling cells. In this dissertation, several methods for using an S_N style sweeper were investigated for the NL-S_2 system. It was found that modifications could be made to the NL-S_2 linear system that drastically reduce the amount of off-diagonal matrix coefficients. The modified NL-S_2 system is equivalent to the original at converge of the scalar flux solution. An S_N style sweeper was found to be highly effective for this modified NL-S_2 streaming plus collision operator for all problems tested including problems with large voids and relatively complicated unstructured two-dimensional meshes and three-dimensional extrusions of these meshes.

Although an S_N style sweeper was found to be effective for inverting the NL-S_2 streaming plus collision operator, the implementation used in this dissertation was not efficient in parallel. Sweepers for the S_N system can be highly efficient in parallel partly because there are many directions to work on when solving the S_N equation. However, there are only eight directions when using the sweeper for the NL-S_2 equations. For the three-dimensional k-eigenvalue calculation investigated in this dissertation, the ideal speedup for NL-S_2 acceleration was very good. But the actual speedup was less than 1 because the problem was solved using several hundred processors. The parallel efficiency of methods for solving the NL-S_2 equations on many processors is an area for future research. NL-S_2 was found to still accelerate both fixed source problems and k-eigenvalue problems with voids. As with other nonlinear methods, stability issues were demonstrated for problems with strongly heterogeneous cross sections, specifically a thick opaque region connected

to a void, however, these stability problems are easily handled by performing two or three S_N sweeps on the outermost iteration instead of one. Using two or three S_N sweeps on the outermost iteration can lead to a faster overall solve as well, for example, using two S_N sweeps on the outermost iteration was found to lead to a faster overall solve than using just one S_N sweep for the 2D C5G7 benchmark problem investigated.

Two iterative methods were investigated in this dissertation for converging the scattering source in the NL-S_2 system, source iteration and GMRES. GMRES was found to be effective, but only a sweep preconditioner (L^{-1}) for GMRES was investigated in this dissertation for solving the NL-S_2 equations. This method can be slow to converge for problems with significant scattering. Better preconditioning is an area for future research and it is noted that methods which are generally too expensive in terms memory requirements to be practical might be viable for preconditioning the NL-S_2 equation because there are fewer unknowns in this system. More sophisticated preconditioning should lead to more efficient convergence of NL-S_2 scalar flux iterates and so the ideal acceleration results presented in this dissertation are not a limit, but can be improved with further research.

REFERENCES

[1] G. Bell and S. Glasstone, *Nuclear Reactor Theory*. 1970.

[2] B. Chang, T. Manteuffel, S. McCormick, J. Ruge, and B. Sheehan, "Spatial multigrid for isotropic neutron transport," *SIAM J. Scientific Computing*, vol. 29, pp. 1900–1917, 2007.

[3] Y. Saad and M. H. Schultz, "Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems," *Siam Journal on Scientific and Statistical Computing*, vol. 7, pp. 856–869, 1986.

[4] C. T. Kelley and Z. Q. Xue, "Gmres and integral operators," *SIAM Journal on Scientific Computing*, vol. 17, no. 1, pp. 217–10, 1996.

[5] B. Guthrie, J. P. Holloway, and B. W. Patton, "Gmres as a multi-step transport sweep accelerator," *Transport Theory and Statistical Physics*, vol. 28, no. 1, pp. 83–102, 1999.

[6] J. E. Morel and J. M. McGhee, "A self-adjoint angular flux equation," *Nuclear Science and Engineering*, vol. 132, no. 3, pp. 312–325, 1999.

[7] F. Kong, Y. Wang, D. Gaston, C. Permann, A. Slaughter, A. Lindsay, M. DeHart, and R. Martineau, "A highly parallel multilevel newton-krylov-schwarz method with subspace-based coarsening and partition-based balancing for the multigroup neutron transport equation on three-dimensional unstructured meshes," *SIAM J. Scientific Computing*, vol. 42, no. 5, pp. C193–C220, 2020.

[8] M. L. Adams and E. W. Larsen, "Fast iterative methods for discrete-ordinates particle transport calculations," *Progress in Nuclear Energy*, vol. 40, no. 1, pp. 3 – 159, 2002.

[9] J. Morel, "Nuen 625 course lecture notes," Spring 2018.

[10] J. S. Warsa, T. A. Wareing, and J. E. Morel, "Fully consistent diffusion synthetic acceleration of linear discontinuous sn transport discretizations on unstructured tetrahedral meshes," *Nuclear Science and Engineering*, vol. 141, no. 3, pp. 236–251, 2002.

[11] Y. Wang and J. C. Ragusa, "Diffusion synthetic acceleration for high-order discontinuous finite element sn transport schemes and application to locally refined unstructured meshes," *Nuclear Science and Engineering*, vol. 166, no. 2, pp. 145–166, 2010.

[12] B. Turcksin and J. C. Ragusa, "Discontinuous diffusion synthetic acceleration for sn transport on 2d arbitrary polygonal meshes," *Journal of Computational Physics*, vol. 274, pp. 356–369, 2014.

[13] J. Lou and J. More, "Linear diffusion-synthetic acceleration with voids," in *International Conference on Mathematics and Computational Methods Applied to Nuclear Science and Engineering (M&C 2017)*, Korean Nuclear Society, 2017.

[14] B. S. Southworth, M. Holec, and T. S. Haut, "Diffusion synthetic acceleration for heterogeneous domains, compatible with voids," *Nuclear Science and Engineering*, vol. 195, no. 2, pp. 119–136, 2021.

[15] J. S. Warsa, T. A. Wareing, and J. 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*, vol. 147, no. 3, pp. 218–248, 2004.

[16] G. L. Ramone, M. L. Adams, and P. F. Nowak, "A transport synthetic acceleration method for transport iterations," *Nuclear Science and Engineering*, vol. 125, no. 3, pp. 257–283, 1997.

[17] D. Y. Anistratov and V. Y. Gol'din, "Nonlinear methods for solving particle transport problems," *Transport Theory and Statistical Physics*, vol. 22, no. 2-3, pp. 125–163, 1993.

[18] T. A. Germogenova and T. A. Sushkevich, "Solution of the transport equation by the mean flux method," *Problems of Shielding Physics*, vol. 3, pp. 34–46, 1969.

[19] T. A. Germogenova, "Convergence of some approximate methods of solving the transport equation.," *Soviet Mathematics - Doklady*, vol. 9, pp. 855–858, 1968.

[20] S. Schunert, Y. Wang, F. Gleicher, J. Ortensi, B. Baker, V. Laboure, C. Wang, M. DeHart, and R. Martineau, "A flexible nonlinear diffusion acceleration method for the sn transport

equations discretized with discontinuous finite elements," *Journal of Computational Physics*, vol. 338, pp. 107–136, 2017.

[21] S. Schunert, H. Hammer, J. Lou, Y. Wang, J. Ortensi, F. Gleicher, B. Baker, M. DeHart, and R. Martineau, "Using directional diffusion coefficients for nonlinear diffusion acceleration of the first order sn equations in near-void regions," in *2016 ANS Winter Meeting and Nuclear Technology Expo*, ANS, 2016.

[22] M. Adams, "New nonlinear methods for linear transport calculations," in *Mathematical methods and supercomputing in nuclear applications Proceedings Vol 1*, pp. 683–694, 1993.

[23] D. Y. Anistratov and E. W. Larsen, "Nonlinear and linear alpha-weighted methods for particle transport problems," *Journal of Computational Physics*, vol. 173, no. 2, pp. 664–684, 2001.

[24] L. Roberts and D. Y. Anistratov, "Nonlinear weighted flux methods for particle transport problems," *Transport Theory and Statistical Physics*, vol. 36, no. 7, pp. 589–608, 2007.

[25] L. Roberts and D. Y. Anistratov, "Nonlinear weighted flux methods for particle transport problems in two-dimensional cartesian geometry," *Nuclear Science and Engineering*, vol. 165, no. 2, pp. 133–148, 2010.

[26] W. H. Reed and T. R. Hill, "Triangular mesh methods for the neutron transport equation (CONF-730414–2)," 1973.

[27] P. Lesaint and P. A. Raviart, "On a finite element method for solving the neutron transport equation," *Publications mathématiques et informatique de Rennes*, no. S4, 1974.

[28] C. Johnson and J. PitkÃd'ranta, "An analysis of the discontinuous galerkin method for a scalar hyperbolic equation," *Mathematics of Computation*, vol. 46, no. 173, pp. 1–26, 1986.

[29] F. BREZZI, L. D. MARINI, and E. SÃIJLI, "Discontinuous galerkin methods for first-order hyperbolic problems," *Mathematical Models and Methods in Applied Sciences*, vol. 14, no. 12, pp. 1893–1903, 2004.

[30] T. R. Z. J. Zienkiewicz, O.C., *Finite Element Method - Its Basis and Fundamentals (6th Edition)*. Elsevier, 2005.

[31] J. I. Vermaak, J. C. Ragusa, M. L. Adams, and J. E. Morel, "Massively parallel transport sweeps on meshes with cyclic dependencies," *Journal of Computational Physics*, vol. 425, p. 109892, 2021.

[32] T. Cormen, C. Leiserson, R. Rivest, and S. Clifford, *Introduction to Algorithms*, pp. 594–597. Cambridge, Massachusetts: The MIT Press, third ed., 2009.

[33] R. Zerr, *Solution Of The Within-Group Multidimensional Discrete Ordinates Transport Equations On Massively Parallel Architectures*. PhD thesis, Pennsylvania State University, 2011.

[34] T. S. Bailey and R. D. Falgout, "Analysis of massively parallel discrete-ordinates transport sweep algorithms with collisions (LLNL-CONF-407968)," Oct. 2008.

[35] R. S. Baker and K. R. Koch, "An sn algorithm for the massively parallel cm-200 computer," *Nucl. Sci. Eng.*, vol. 128, 1998.

[36] M. P. Adams, M. L. Adams, W. D. Hawkins, T. Smith, L. Rauchwerger, N. M. Amato, T. S. Bailey, and R. D. Falgout, "Provably optimal parallel transport sweeps on regular grids (LLNL-CONF-407968)," 2013.

[37] M. Rosa, J. S. Warsa, and J. H. Chang, "Fourier analysis of inexact parallel block-jacobi splitting with transport synthetic acceleration," *Nuclear Science and Engineering*, vol. 164, no. 3, pp. 248–263, 2010.

[38] T. S. Bailey, *The piecewise linear discontinuous finite element method applied to the RZ and XYZ transport equations*. PhD thesis, Texas A&M University, 2008.

[39] C. Cavarec, J. Perron, D. Verwaerde, and J. West, "Benchmark calculations of power distributions within assemblies. HT-12/94006.," 1994.

[40] E. E. Lewis, M. A. Smith, N. Tsoulfanidis, G. Palmiotti, T. A. Taiwo, and R. N. Blomquist, "Benchmark specification for deterministic 2-d/3-d mox fuel assembly transport calculations without spatial homogenisation (C5G7 MOX). NEA/NSC/DOC(2001)4.," 2001.

[41] *Benchmark on Deterministic Transport Calculations Without Spatial Homogenization: A 2-D/3-D MOX Fuel Assembly Benchmark*. OECD, 2003.

[42] M. D. DeHart, Z. Mausolff, Z. Weems, D. Popp, K. Smith, F. Shriver, S. Goluoglu, Z. Prince, and J. Ragusa, "Preliminary results for the oecd/nea time dependent benchmark using rattlesnake, rattlesnake-iqs and tdkeno (INL/EXT-16-39723)," 2016.

[43] Y. Wang, Z. M. Prince, J. T. Hanophy, C. Lee, Y. S. Jung, H. Park, L. H. Harbour, and J. Ortensi, "Performance improvements for the griffin transport solvers (INL/EXT-21-64272-Rev000)," 2021.

[44] V. M. Laboure, Y. Wang, and M. D. DeHart, "Least-squares pn formulation of the transport equation using self-adjoint-angular-flux consistent boundary conditions (INL/CON-15-36402)," 2016.