ABSTRACT

POWELL, BRIAN PAUL. An Advanced Algorithm for Construction of Integral Transport Matrix Method Operators Using Accumulation of Single Cell Coupling Factors. (Under the direction of Yousry Y. Azmy.)

The Integral Transport Matrix Method (ITMM) has been shown to be an effective method for solving the neutron transport equation in large domains on massively parallel architectures. In the limit of very large number of processors, the speed of the algorithm, and its suitability for unstructured meshes, i.e. other than an ordered Cartesian grid, is limited by the construction of four matrix operators required for obtaining the solution in each sub-domain. The existing algorithm used for construction of these matrix operators, termed the differential mesh sweep, is computationally expensive and was developed for a structured grid. This work proposes the use of a new algorithm for construction of these operators based on the construction of a single, fundamental matrix representing the transport of a particle along every possible path throughout the sub-domain mesh. Each of the operators is constructed by multiplying an element of this fundamental matrix by two factors dependent only upon the operator being constructed and on properties of the emitting and incident cells. The ITMM matrix operator construction time for the new algorithm is demonstrated to be shorter than the existing algorithm in all tested cases with both isotropic and anisotropic scattering considered. While also being a more efficient algorithm on a structured Cartesian grid, the new algorithm is promising in its geometric robustness and potential for being applied to an unstructured mesh, with the ultimate goal of application to an unstructured tetrahedral mesh on a massively parallel architecture.

المنارة للاستشارات

www.manaraa.com

© Copyright 2013 by Brian Paul Powell

All Rights Reserved

www.manaraa.com

An Advanced Algorithm for Construction of Integral Transport Matrix Method Operators Using Accumulation of Single Cell Coupling Factors

by
Brian Paul Powell

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Nuclear Engineering

Raleigh, North Carolina

2013

APPROVED BY:

Dmitriy Y. Anistratov

Robert E. White

Yousry Y. Azmy
Chair of Advisory Committee

www.manaraa.com

DEDICATION

To my wife, Maria, and daughter, Isabella

المنـــارة للاستشارات

www.manaraa.com

BIOGRAPHY

The author graduated from the United States Military Academy at West Point in May, 2006 with a B.S. in Nuclear Engineering. He then served as an active duty U.S. Army Officer from 2006 to 2011 with deployments to both Iraq and Afghanistan. In 2011, Brian returned to academia to pursue graduate studies in Nuclear Engineering, specifically in the field of Computational Neutron Transport Theory under the direction of Professor Yousry Y. Azmy. In 2012, Brian was awarded the Department of Energy Computational Science Graduate Fellowship.

المنارة للاستشارات
www.manaraa.com

ACKNOWLEDGEMENTS

The Department of Energy Computational Science Graduate Fellowship (DOE CSGF) and the Department of Energy Nuclear Energy University Program (DOE NEUP) both supported this work, in part. I would also like to thank my advisor, Professor Yousry Y. Azmy, for his endless patience and support.

www.manaraa.com

TABLE OF CONTENTS

المنارة للاستشارات
www.manaraa.com

LIST OF TABLES

المنارة للاستشارات
www.manaraa.com

LIST OF FIGURES

المنارة للاستشارات
www.manaraa.com

المنارة للاستشارات

www.manaraa.com

Chapter 1

Introduction

The neutron transport equation governs neutron movement through a medium over time at various energies and directions. The neutron transport equation is [1]

$$\frac{1}{v}\frac{\partial}{\partial t}\psi(\vec{r},\hat{\Omega},E,t) + \hat{\Omega}\cdot\vec{\bigtriangledown}\psi(\vec{r},\hat{\Omega},E,t) + \sigma(\vec{r},E)\psi(\vec{r},\hat{\Omega},E,t) = \sigma_s(\vec{r},E)\phi(\vec{r},E,t) + q(\vec{r},\hat{\Omega},E,t) \ , \tag{1.1}$$

where v is the neutron speed, $\psi(\vec{r},\hat{\Omega},E,t)$ is the neutron angular flux at position $\vec{r}$, direction of travel $\hat{\Omega}$, energy E, and time t, $\sigma(\vec{r},E)$ is the total neutron cross section at position $\vec{r}$ and energy E, $\sigma_s(\vec{r},E)$ is the neutron scattering cross section at position $\vec{r}$ and energy E, $\phi(\vec{r},E,t)$ is the neutron scalar flux at position $\vec{r}$, energy E, and time t, and $q(\vec{r},\hat{\Omega},E,t)$ is the fixed neutron source at position $\vec{r}$, direction of travel $\hat{\Omega}$, energy E, and time t.

For the purposes of this work, the neutron transport equation is considered to be time-independent. The removal of the time variable results in the form of the transport equation considered here,

$$\hat{\Omega}\cdot\vec{\bigtriangledown}\psi(\vec{r},\hat{\Omega},E) + \sigma(\vec{r},E)\psi(\vec{r},\hat{\Omega},E) = \sigma_s(\vec{r},E)\phi(\vec{r},E) + q(\vec{r},\hat{\Omega},E) \ . \tag{1.2}$$

This form of the transport equation must be discretized in space, angle, and energy to allow for a numerical solution. The Integral Transport Matrix Method (ITMM) for solving the neutron transport equation, which is the method examined in detail in this work, makes use of each of these discretizations.

In the field of computational neutron transport theory, advanced methods for solving the neutron transport equation are needed for solving larger problems on a greater number of processors. Solution algorithms suitable for massively parallel architectures must continue to achieve speedup as the number of processors is increased. One such method that has shown promise in this area is the ITMM.

This work focuses on the basic premise of the ITMM, which is the construction of four matrix

www.manaraa.com

operators used iteratively in two equations to converge to the solution of the neutron transport equation. This work is motivated by the desire to implement the ITMM on geometries for complicated engineering using unstructured grids. The rise of unstructured tetrahedral mesh transport methods and codes as a means to a more accurate representation of geometric configurations typical of radiation transport problems requires adaptations of the ITMM for the application of this method to such a mesh.

A complete algorithm for the construction of the four matrix operators on a Cartesian grid has already been developed. This algorithm, although effective for constructing the matrix operators on a Cartesian grid, is computationally expensive and requires orderly mesh structure to perform well. As such, the need for a method of constructing the ITMM matrix operators that is both more computationally efficient and geometrically robust is evident.

The method presented in this work, although confined to three-dimensional Cartesian meshes at this stage, is promising in that it has demonstrated significantly faster matrix operator construction times and has the potential to be applied with relative ease to an unstructured tetrahedral mesh. This algorithm derives its efficiency from the fact that all transport of particles through a mesh, whether originating from a distributed source or an incoming angular flux, follows the same paths from the emergent face of a cell to the incident face of a cell. This principle allows for the creation of a fundamental matrix for face-to-face transport, off of which each of the matrix operators required for the ITMM can be constructed. It will therefore be termed the Fundamental Matrix Method (FMM).

In the following chapter, relevant literature will be reviewed. This will be followed by a derivation of the ITMM. The FMM will then be derived, along with a complete example of the construction of the four ITMM operators for a 3 x 3 cell system in two dimensions. A performance model for the FMM algorithm will then be described, followed by a comparison of matrix operator construction time results for several different scenarios for both the existing algorithm and the FMM. This will be followed by a comparison of the FMM timing results to the algorithm performance model. Finally, conclusions will be presented along with thoughts on future work to be completed through the use of the FMM algorithm.

المنارة للاستشارات
www.manaraa.com

Chapter 2

Review of Literature

The Integral Transport Matrix Method (ITMM) of solving the neutron transport equation on multiprocessing platforms is a spatial domain decomposition method and is therefore ideally applied to a large domain in a massively parallel computational environment. Given that this work focuses on the construction of the matrix operators needed for the ITMM and not computation of the solution, either in serial or parallel, the review of literature will focus on the development of the use of matrix operators in solving the neutron transport equation. The ITMM makes use of angular, spatial, and energy discretization, so a brief review of these methods will be examined as well.

2.1 Discretization of the Transport Equation

Energy discretization methods, generally referred to as multigroup methods, allow for solving the transport equation for neutrons in a specific energy range. This is accomplished by integrating equation 1.2 over all energies, resulting in [1]

$$\hat{\Omega} \cdot \vec{\bigtriangledown}\psi_g(\vec{r},\hat{\Omega}) + \sigma_g(\vec{r})\psi_g(\vec{r},\hat{\Omega}) = \sigma_{s,g}(\vec{r})\phi_g(\vec{r}) + q_g(\vec{r},\hat{\Omega}) \ , \tag{2.1}$$

where the subscript g indicates a specific energy group. Lewis and Miller discuss multigroup methods in detail [1]. Here, it is sufficient to note that the methods derived in this work solve the transport equation for a single energy group, hence the subscript g will be suppressed.

Angular discretization, as accomplished by the method of discrete ordinates, is a key component of this work. Discretizing equation 2.1 by angle yields [1]

$$(\mu_n \frac{\partial}{\partial x} + \eta_n \frac{\partial}{\partial y} + \xi_n \frac{\partial}{\partial z})\psi_n(\vec{r}) + \sigma(\vec{r})\psi_n(\vec{r}) = \sigma_s(\vec{r})\phi(\vec{r}) + q(\vec{r}) \ , \tag{2.2}$$

www.manaraa.com

where μ is the angular cosine in the x-direction, η is the angular cosine in the y-direction, ξ is the angular cosine in the z-direction, and the subscript n indicates a discrete angle. The discrete ordinates method has been used in production-level radiation transport codes since the 1960s, with much of the early development done by Carlson and Lathrop [2]. Significant advances in discrete ordinates methodology since that time have been described by Larsen and Morel [3]. Lewis and Miller [1] provide a thorough description of a discrete ordinates solution algorithm in one, two, and three dimensions using the diamond difference spatial differencing method. By solving the transport equation for the angular flux at several individual angles in quadrature discrete ordinates allows for an accurate approximation to the solution of the transport equation. However, discrete ordinates algorithms generally require a repetitive and computationally expensive mesh sweep in order to converge to the solution. In the ITMM formulation, however, only a single mesh sweep was required in single sub-domain, with convergence achieved by iterating on the incoming angular flux to each sub-domain [4].

Spatial discretization, also known as spatial differencing, allows for solving the transport equation in a finite domain by separating that domain into multiple nodes, or cells, in each of which the flux solution can be found and related to neighboring cells. Considering equation 2.2, spatial central differencing of the partial derivatives comprising the streaming operator yields [1]

$$\begin{aligned}\frac{\mu_n}{\triangle x_i}(\psi_{n,i+1/2,j,k} - \psi_{n,i-1/2,j,k}) + \frac{\eta_n}{\triangle y_j}(\psi_{n,i,j+1/2,k} - \psi_{n,i,j-1/2,k}) + \\ \frac{\xi_n}{\triangle z_k}(\psi_{n,i,j,k+1/2} - \psi_{n,i,j,k-1/2}) + \sigma_{i,j,k}\psi_{n,i,j,k} = \sigma_{s,i,j,k}\phi_{i,j,k} + q_{i,j,k}\ .\end{aligned} \tag{2.3}$$

Where i, j, k are the indices of the given cell in the $x-, y-, z-$ direction, respectively. The subscript $i-1/2, j, k$ indicates the left x-face of cell i, j, k and the subscript $i+1/2, j, k$ indicates the right x-face of cell i, j, k, with analogous meanings in the y and z directions.

Larsen and Morel [3] describe in detail three methods of spatial discretization used in solving the transport equation: The characteristic method, the linear discontinuous method, and nodal methods. Any one of these methods form what is called the auxiliary equation, which relates the incoming and outgoing angular flux in a cell to the angular flux at the cell center. In the ITMM formulation used in this work, the diamond difference equation is the auxiliary equation of choice. The diamond difference method is the approximation of the angular flux at the cell center by

$$\psi_{n,i,j,k} = \frac{1}{2}(\psi_{n,i+1/2,j,k} + \psi_{n,i-1/2,j,k}) \tag{2.4}$$

and analogously for the y and z directions. Lewis and Miller [1] describe the diamond difference method in detail, and its application to this work is described in the next chapter.

المنارة للاستشارات
www.manaraa.com

2.2 Parallel Domain Decomposition of the Transport Equation

The benefits of the ITMM are realized only on multiprocessing platforms. The method is not competitive in serial mode but is highly competitive in parallel [4] and therefore the application of parallel domain decomposition must be discussed. Domain decomposition can accomplished through three distinct means: Energy domain decomposition, angular domain decomposition, and (as in the ITMM formulation) spatial domain decomposition. A hybrid decomposition combining any of these three methods is also possible. Developments in parallel domain decomposition are also much more recent than discretization methods due to the availability of parallel computing resources that became widespread starting in the 1980s.

Energy domain decomposition involves solving the transport equation in a sub-domain for different energy groups, each on a separate processor. The obvious advantage of domain decomposition by energy is the ability to rapidly solve transport problems involving a large span of energies. As such, energy domain decomposition is particularly applicable to nuclear reactor core simulation, where neutrons are constantly changing energy through collisions and neutrons at varying energies have important properties (e.g. thermal neutrons causing fission). Much of the early work in energy domain decomposition was completed by Weinke and Hiromoto [5,6,7]. Despite its applications, energy domain decomposition is the least used of domain decompositions due to the possibility that the energy groups could be solved out of order, increasing the iterative cost of convergence with increasing number of participating processors[8,9].

Angular domain decomposition is based on the previously described discrete ordinates angular discretization method, the main difference being that the transport equation can be solved for each ordinate on a separate processor, assuming vacuum boundary conditions on all external faces of the problem domain. The resulting angular fluxes are then summed in quadrature on the base processor or in parallel to achieve a solution for the scalar flux, or, if desired, angular moments of the flux. While advantageous in that, in Cartesian geometry, the ordinates do not need to be solved in any particular order, the disadvantage of angular domain decomposition is that the decomposition is limited to the number of ordinates, which generally will not exceed several hundred, to a few thousand at the very most. Therefore, this limitation does not allow for massively parallel implementations of angular domain decomposition schemes. Another disadvantage of angular domain decomposition is that the full spatial domain must be replicated on all processors (e.g. large flux arrays), which requires significant memory storage. Fischer and Azmy [8] detailed a performance model for both angular and spatial domain decomposition methods in order to determine which method was better applied to a given problem. They concluded that, for a small computational cluster, it is more efficient to discretize by angle and that the opposite is true as the number of processors is increased. Azmy [9] also described the development of angular domain decomposition methods in detail.

www.manaraa.com

Spatial domain decomposition is the domain decomposition method used by the ITMM. By separating a domain into multiple smaller sub-domains, each solved on a separate processor, the transport equation can be solved faster. A disadvantage of spatial domain decomposition is the necessity of a sub-domain to have the solution to the outgoing angular flux in neighboring domains. Any disadvantage that this causes, however, is outweighed by the ability to decompose the domain into a large number of smaller sub-domains to match the number of available processors, assuming (as typical for large applications worthy of massively parallel solution) that the spatial discretization yields significantly more computational cells than available processors for execution.

Early work in spatial domain decomposition was completed by Yavuz and Larsen [10], who first attempted a spatial domain decomposition algorithm on one-dimensional and two-dimensional domains. Yavuz and Larsen [11] also developed the Alternating Direction Transport Sweep (ADTS), which allowed for adjacent sub-domains to receive updated values for the incoming angular flux. Although their method was able to achieve parallel speedup, it was not designed for massively parallel architectures and is therefore not competitive with the ITMM. A detailed account of the development of early spatial domain decomposition methods is provided by Azmy [9].

It is important to emphasize one spatial domain decomposition method: The KBA (Koch, Baker, Alcouffe) algorithm [12,13], also known as the wavefront algorithm, is a method of spatial domain decomposition that is used by the current state-of-the-art neutron transport codes [4]. The KBA algorithm involves the conduct of a sequential mesh sweep on a diagonal wavefront through the given domain, where the transport equation is solved on a single processor for each sub-domain and the outgoing angular fluxes are used as incoming angular fluxes in the adjacent sub-domain. It has been shown [13] that the KBA algorithm is effective on massively parallel architectures. The shortcoming of the KBA algorithm, however, is that, due to the sequential nature of the parallel mesh sweep, some processors must remain dormant. The ITMM solution algorithm works simultaneously across the entire domain, minimizing unused processors and therefore is competitive with KBA as a massively parallel spatial domain decomposition method [4]. An important difference to note between the KBA and ITMM algorithms is that KBA is synchronous and ITMM is asynchronous, resulting in the disadvantage of KBA, processor idleness, being essentially traded for increasing iterations in the ITMM.

2.3 Response Matrix Methods

The ITMM, at its foundation, has close similarity to response matrix methods in that it makes use of four distinct matrix operators (described in the next chapter) which are multiplied by

www.manaraa.com

a known quantity (e.g., incoming angular flux at the boundaries) and provide a response (e.g., outgoing flux at the boundaries). Each ITMM matrix operator is then essentially a response matrix.

Response matrix methods (RMM) have been developed for use in solving the transport equation, but with widely varied techniques [1]. A thorough review and derivation of the RMM in addition to applications in nuclear engineering for both transport theory and the diffusion approximation were provided by Lindahl and Weiss in 1981 [14]. Lindahl and Weiss asserted that the RMM provides the solution of a transport problem on a large domain by consolidating solutions of smaller sub-domains. They also state that the advantageous characteristic of the RMM is the ability to consider each sub-domain as a separate problem. This general introduction to the RMM is remarkably similar to the basis of the ITMM, which breaks a larger domain into smaller sub-domains on a multiprocessing platform, allowing for the consideration of each sub-domain as a separate problem, where the global solution is obtained by iterating on the angular flux at the boundaries of the sub-domains. Lindahl and Weiss consider each sub-domain, or node, as they identify it, to be characterized by a response kernel which is, in essence, a probability of particle interaction inside the node. The response kernel is then more concretely identified as a response matrix which, when multiplied by an incoming angular flux, produces an outgoing angular flux. The definition of the response matrix is also expanded to include the response of a sub-domain over several nodes rather than just a single node. Lindahl and Weiss go on to define four distinct response matrices which relate in the incoming current, outgoing current, scalar flux, and fixed source in the sub-domain. These matrices are used in two equations iterating on the incoming current to provide a global solution to the transport equation.

The RMM of Lindahl and Weiss bears significant similarity to the ITMM. Both methods, generally, make use of multiple cell, or node, sub-domains and the four response matrices using the iterative solution method described in the previous paragraph. The differences lie in the specifics of the method. The RMM uses particle currents, whereas the ITMM is able to make use of angular flux via the discrete ordinates method and spatial differencing. The response matrices are also constructed in a very different manner. Lindahl and Weiss use block matrices defining the response of each node individually to construct the larger response matrix. This construction method leads to computational storage inefficiency by requiring the storage of more zeros than meaningful data. Contrasting this approach, the ITMM requires the construction of differential matrix operators and, through the use of discrete ordinates, allows for operator elements to be grouped efficiently and avoids unnecessary storage.

Although response matrix methods exist for several techniques including Monte Carlo, collision probabilities, and finite elements [1], it is the use of discrete ordinates that is the basis for the ITMM. A discrete ordinates formulation of the response matrix method was developed in 1992

المنارة للاستشارات
www.manaraa.com

by Hanebutte and Lewis [15], who proposed using a response matrix algorithm to iteratively solve the transport equation. The same two-equation iteration technique used by Lindahl and Weiss [14] is used to iterate on the incoming angular flux. The main difference between the two techniques is that Hanebutte and Lewis iterate on the incoming angular flux as opposed to the incoming current used by Lindahl and Weiss. The discrete ordinates method allows for this difference. The method of Hanebutte and Lewis was also limited to the response of a single-cell sub-domain and was therefore computationally expensive in an iterative sense. The ITMM then, is an evolution of the work of both Hanebutte and Lewis and Lindahl and Weiss.

2.4 The Integral Transport Matrix Method

The response matrix algorithm of Hanebutte and Lewis [15] was not further developed until 1997 by Azmy [16], who first proposed a method for using full-domain operators, rather than single cell operators, to allow for a solution to the transport equation without the need for repetitive mesh sweeps that continue to dominate deterministic method solution algorithms. Azmy also described the algorithm required to construct the matrix operator that relates the scalar flux spatial moments in all cells to the fixed source spatial moments. This matrix operator is identified as $\boldsymbol{A}$ in the equation

$$\phi^v = \boldsymbol{A}(\sigma^s \phi^p + S) \ , \tag{2.5}$$

where ϕ^v is the scalar flux in the current iteration, ϕ^p is the scalar flux in the previous iteration, σ^s is the scattering cross section operator (basically a diagonal matrix whose elements are the cell by cell isotropic scattering cross section), and S is the fixed neutron source. Azmy identified that $\boldsymbol{A}$ is essentially an iterative map of the scalar flux. The algorithm to construct $\boldsymbol{A}$ was based on a mesh sweep using differential relations between the angular flux and scalar flux spatial moments to create the matrix identified as the iteration Jacobian. An assumption critical to the development of the ITMM made in this work is that the iterations defined in equation 2.5 are convergent yielding a converged scalar flux solution, ϕ^∞, giving

$$\phi^\infty = (I - \sigma^s \boldsymbol{A})^{-1} \boldsymbol{A} S \ . \tag{2.6}$$

Through this assumption, previous and current iterates of the cell scalar flux converge to the same value, eliminating a variable and allowing for a solution of the converged cell scalar flux without iteration based on the matrix operator $\boldsymbol{A}$ and the fixed neutron source S. Azmy's work is the first thorough description of an algorithm required to create a matrix operator in the ITMM, $\boldsymbol{A}$. However, it is limited to the case of vacuum boundary conditions and also does not consider outgoing angular flux from the sub-domain.

www.manaraa.com

Azmy continued to evolve the method in 1999 [17] by describing a matrix operator $\boldsymbol{B}$ such that

$$\boldsymbol{B} = (\boldsymbol{I} - \sigma^s \boldsymbol{A}) \ . \tag{2.7}$$

Azmy provides a thorough description of the algorithm required to create $\boldsymbol{B}$, including the specific values of diagonal and off-diagonal elements of the matrix and also examines preconditioning methods to allow for faster convergence. Although he further developed the algorithm for matrix operator $\boldsymbol{B}$ in this work, he continued to use only vacuum boundary conditions in his calculations. Rosa, Azmy, and Morel expanded this work in 2009 [18], examining spectral properties of $\boldsymbol{A}$ and $\boldsymbol{B}$ and further developing the algorithm for the construction of these operators on configurations with vacuum boundary conditions. Rosa *et al.* identified $\boldsymbol{B}$ as the integral transport matrix.

Zerr provided major developments in the ITMM in 2011 [4] by expanding the method to include non-vacuum incoming and outgoing angular flux at the boundaries of the sub-domain. In doing so, he created four matrix operators $\boldsymbol{J}_{\phi}$, $\boldsymbol{J}_{\psi}$, $\boldsymbol{K}_{\phi}$, and $\boldsymbol{K}_{\psi}$, with $\boldsymbol{J}_{\phi}$ related to Azmy's previously defined operator as

$$\boldsymbol{J}_{\phi} = \boldsymbol{A}\boldsymbol{C} \ , \tag{2.8}$$

where $\boldsymbol{C}$ is the scattering-ratio matrix (a diagonal matrix whose elements are the cell by cell scattering ratios). The use of each of these operators will be described in the following chapter. Most importantly to this work, Zerr describes the construction algorithm, the Differential Mesh Sweep (DMS), used to create each of the four matrix operators of the ITMM that are necessary to account for non-trivial incoming angular flux. Zerr also developed a code for parallel implementation of this algorithm in constructing the operators and iteratively solving the transport equation across sub-domains. The DMS matrix operator construction algorithm is the motivation of this work in that it is desirable, based on the parallel performance demonstrated by Zerr, for the ITMM to be applied to other geometries. A more geometrically robust and less computationally expensive algorithm for the construction of the four matrix operators would be ideal in this application.

www.manaraa.com

Chapter 3

The Integral Transport Matrix Method

The starting point of the ITMM is the neutron balance equation for a single cell in three dimensions, which, considering only isotropic scattering, is [1]

$$\begin{aligned}\varepsilon^x_{n,i,j,k}\psi_{n,i_{out},j,k} + \varepsilon^y_{n,i,j,k}\psi_{n,i,j_{out},k} + \varepsilon^z_{n,i,j,k}\psi_{n,i,j,k_{out}} + \psi_{n,i,j,k} \\ = c_{i,j,k}\phi_{i,j,k} + \sigma^{-1}_{t,i,j,k}q_{i,j,k} + \varepsilon^x_{n,i,j,k}\psi_{n,i_{in},j,k} + \varepsilon^y_{n,i,j,k}\psi_{n,i,j_{in},k} + \varepsilon^z_{n,i,j,k}\psi_{n,i,j,k_{in}}\end{aligned} \tag{3.1}$$

where,

$$\varepsilon^x_{n,i,j,k} = \frac{|\mu_n|}{\sigma_{t,i,j,k}\Delta x_i}, \text{AFyz.} \tag{3.2}$$

AFyz stands for "analogously for y and z", and, by the discrete ordinates approximation with quadrature weights w_n, the scalar flux (ϕ) is related to the angular fluxes (ψ) by the quadrature sum

$$\phi_{i,j,k} = \sum_{n=1}^{D} w_n\psi_{n,i,j,k}, \tag{3.3}$$

where D is the total number of angles. In the above equations, μ_n is the angular cosine with respect to the x-axis of the direction of particle travel along the n^{th} discrete ordinate. $\sigma_{t,i,j,k}$ is the macroscopic total interaction cross section of the material in cell i,j,k. Δx_i is the width of the cell in the x direction. $c_{i,j,k}$ is the scattering ratio, $\frac{\sigma_{s,i,j,k}}{\sigma_{t,i,j,k}}$, of the material in cell i,j,k , where $\sigma_{s,i,j,k}$ is the macroscopic scattering cross section in the same cell. $q_{i,j,k}$ is the distributed source in cell i,j,k. $\psi_{n,i_{out},j,k}$ is the angular flux along the n^{th} discrete ordinate leaving cell i,j,k out of the $x = constant$ face, with analogous definitions for angular flux leaving at the $y = constant$ and $z = constant$ faces. $\psi_{n,i_{in},j,k}$ is the angular flux traveling along the n^{th} discrete ordinate entering cell i,j,k in the $x = constant$ face, with analogous definitions for angular flux entering

المنارة للاستشارات
www.manaraa.com

at the $y = constant$ and $z = constant$ faces. Finally, $\psi_{n,i,j,k}$ is the cell-centered angular flux in cell i, j, k of neutrons traveling along the n^{th} discrete ordinate.

The diamond difference relation in all three dimensions is used to establish a closed matrix system of equations for the unknown fluxes on the left hand side of equation 3.1 [1],

$$\psi_{n,i,j,k} = \frac{1}{2}(\psi_{n,i_{in},j,k} + \psi_{n,i_{out},j,k}), \text{AFyz}, \tag{3.4}$$

resulting in

$$\begin{bmatrix} 1 & \varepsilon^z_{n,i,j,k} & \varepsilon^y_{n,i,j,k} & \varepsilon^x_{n,i,j,k} \\ 1 & -0.5 & 0 & 0 \\ 1 & 0 & -0.5 & 0 \\ 1 & 0 & 0 & -0.5 \end{bmatrix} \begin{bmatrix} \psi_{n,i,j,k} \\ \psi_{n,i,j,k_{out}} \\ \psi_{n,i,j_{out},k} \\ \psi_{n,i_{out},j,k} \end{bmatrix} = \begin{bmatrix} 1 & \varepsilon^z_{n,i,j,k} & \varepsilon^y_{n,i,j,k} & \varepsilon^x_{n,i,j,k} \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} c_{i,j,k}\phi^p_{i,j,k} + \sigma^{-1}_{t,i,j,k} q_{i,j,k} \\ \psi_{n,i,j,k_{in}} \\ \psi_{n,i,j_{in},k} \\ \psi_{n,i_{in},j,k} \end{bmatrix}. \tag{3.5}$$

Left multiplying both sides of equation 3.5 by the inverted coefficient matrix from the left hand side yields

$$\begin{bmatrix} \psi_{n,i,j,k} \\ \psi_{n,i,j,k_{out}} \\ \psi_{n,i,j_{out},k} \\ \psi_{n,i_{out},j,k} \end{bmatrix} = \Gamma_n \begin{bmatrix} c_{i,j,k}\phi^p_{i,j,k} + \sigma^{-1}_{t,i,j,k} q_{i,j,k} \\ \psi_{n,i,j,k_{in}} \\ \psi_{n,i,j_{in},k} \\ \psi_{n,i_{in},j,k} \end{bmatrix} \tag{3.6}$$

where

$$\Gamma_n = \begin{bmatrix} j_\phi & k_{\phi,z} & k_{\phi,y} & k_{\phi,x} \\ j_{\psi,z} & k_{\psi,z\to z} & k_{\psi,y\to z} & k_{\psi,x\to z} \\ j_{\psi,y} & k_{\psi,z\to y} & k_{\psi,y\to y} & k_{\psi,x\to y} \\ j_{\psi,x} & k_{\psi,z\to x} & k_{\psi,y\to x} & k_{\psi,x\to x} \end{bmatrix} \tag{3.7}$$

The Γ matrix is then a set of coupling factors (named here according to their function, which will be clarified later) between the distributed (fixed and scattering) source, incoming angular fluxes, and outgoing angular fluxes. Alternative spatial discretization methods correspond to different auxiliary relations, equation 3.4, that yield the same relation, equation 3.5, with different Γ_n elements.

www.manaraa.com

All of the previous equations provide the relationships between the incoming and outgoing angular flux and the scalar flux for only a single cell. When considering a multi-cell sub-domain, the relationship between the cells is simply an extension of

$$\psi_{n,i+1_{in},j,k} = \psi_{n,i_{out},j,k}, \text{AFyz} \tag{3.8}$$

Using this relation to extend equation 3.6 to a global system with vacuum boundary conditions, the single cell values of ϕ and q need to be extended to the vectors $\boldsymbol{\phi}$ and $\boldsymbol{q}$ containing the values of the scalar flux and fixed source, respectively, in each cell. In a source iteration scheme with $\boldsymbol{\phi^p}$ as the previous iterate of the scalar flux and $\boldsymbol{\phi^v}$ as the current iterate of the scalar flux using equations 3.6 and 3.3, the system reduces to [4]

$$\boldsymbol{\phi^v} = \boldsymbol{A}(\boldsymbol{C}\boldsymbol{\phi^p} + \boldsymbol{\Sigma_t^{-1}}\boldsymbol{q}) \ , \tag{3.9}$$

where $\boldsymbol{C}$ is the scattering ratio diagonal matrix and $\boldsymbol{\Sigma_t^{-1}}$ is the inverse total cross section diagonal matrix. $\boldsymbol{A}$ is then a coefficient matrix constructed from elements of Γ which relates the previous scalar flux iterate to the new scalar flux iterate. Azmy [16] defines A as an iterative map of the scalar flux. Partially differentiating equation 3.9 with respect to $\boldsymbol{\phi^p}$ yields the iteration Jacobian Matrix,

$$\frac{\boldsymbol{\partial\phi^v}}{\boldsymbol{\partial\phi^p}} = \boldsymbol{AC} \tag{3.10}$$

$\boldsymbol{AC}$ was denoted by Zerr [4] as $\boldsymbol{J_\phi}$.

To factor $\boldsymbol{J_\phi}$ instead of $\boldsymbol{A}$ from equation 3.9, the second term needs to be altered to

$$\boldsymbol{\phi^v} = \boldsymbol{A}(\boldsymbol{C}\boldsymbol{\phi^p} + \boldsymbol{\Sigma_t^{-1}}\boldsymbol{\Sigma_s}\boldsymbol{\Sigma_s^{-1}}\boldsymbol{q}) \tag{3.11}$$

Substituting

$$\boldsymbol{C} = \boldsymbol{\Sigma_t^{-1}}\boldsymbol{\Sigma_s} \tag{3.12}$$

yields

$$\boldsymbol{\phi^v} = \boldsymbol{AC}(\boldsymbol{\phi^p} + \boldsymbol{\Sigma_s^{-1}}\boldsymbol{q}) \tag{3.13}$$

Then, substituting $\boldsymbol{J_\phi}$ for $\boldsymbol{AC}$,

$$\boldsymbol{\phi^v} = \boldsymbol{J_\phi}(\boldsymbol{\phi^p} + \boldsymbol{\Sigma_s^{-1}}\boldsymbol{q}) \tag{3.14}$$

The converged scalar flux solution will occur when

www.manaraa.com

$$\boldsymbol{\phi^v} = \boldsymbol{\phi^p} = \boldsymbol{\phi^\infty} \tag{3.15}$$

Substituting this relation into equation 3.14 yields

$$\boldsymbol{\phi^\infty} = (\boldsymbol{I} - \boldsymbol{J_\phi})^{-1}\boldsymbol{J_\phi}\boldsymbol{\Sigma_s^{-1}q} \tag{3.16}$$

Removing the constraints of vacuum boundary conditions, as is necessary to apply the ITMM to a generic sub-domain, requires that the effect of the incoming angular flux on the scalar flux within each cell of the sub-domain be accounted for. To that end, another term is added to equation 3.14, yielding [4]

$$\boldsymbol{\phi^v} = \boldsymbol{J_\phi}(\boldsymbol{\phi^p} + \boldsymbol{\Sigma_s^{-1}q}) + \boldsymbol{K_\phi}\boldsymbol{\psi_{in}} \tag{3.17}$$

where $\boldsymbol{\psi_{in}}$ is a vector containing all incoming angular fluxes to the sub-domain and $\boldsymbol{K_\phi}$ is a matrix operator constructed from the elements of Γ which defines the effect of the incoming angular flux on the scalar flux in each cell of the sub-domain. The dimension of $\boldsymbol{\psi_{in}}$ is then the number of faces comprising the exterior of the sub-domain multiplied by the number of incoming ordinates to each surface. $\boldsymbol{K_\phi}$ has the same number of columns as the dimension of $\boldsymbol{\psi_{in}}$ and the number of rows is equal to the number of cells in the sub-domain. The construction of $\boldsymbol{K_\phi}$ will be described in the next section.

By the same logic that achieved equation 3.16, the converged scalar flux solution with incoming angular flux at the boundaries is then [4]

$$\boldsymbol{\phi^\infty} = (\boldsymbol{I} - \boldsymbol{J_\phi})^{-1}\boldsymbol{J_\phi}\boldsymbol{\Sigma_s^{-1}q} + (\boldsymbol{I} - \boldsymbol{J_\phi})^{-1}\boldsymbol{K_\phi}\boldsymbol{\psi_{in}^\infty}\ , \tag{3.18}$$

where $\boldsymbol{\psi_{in}^\infty}$ is a vector containing all converged incoming angular fluxes to the sub-domain. Both the effect of a fixed source ($\boldsymbol{J_\phi}$) and of an incoming angular flux ($\boldsymbol{K_\phi}$) on the scalar flux in each cell in the sub-domain have now been accounted for. The next consideration, therefore, is the calculation of the outgoing angular flux from a sub-domain consisting of two components: Upon convergence, the outgoing angular flux satisfies [4],

$$\boldsymbol{\psi_{out}^\infty} = \boldsymbol{J_\psi}\boldsymbol{\phi^\infty} + \boldsymbol{K_\psi}\boldsymbol{\psi_{in}^\infty} \tag{3.19}$$

where $\boldsymbol{\psi_{in}}^\infty$ and $\boldsymbol{\phi^\infty}$ have been previously defined, $\boldsymbol{J_\psi}$ is a matrix operator constructed from the elements of Γ with dimensions that are the transpose of the previously defined matrix operator, $\boldsymbol{K_\phi}$, and $\boldsymbol{K_\psi}$ is a square matrix operator (assuming the typical reflective symmetry of discrete ordinates) constructed from the elements of Γ with a dimension equal to the dimension

www.manaraa.com

of the vector $\boldsymbol{\psi_{in}}$.

As the construction of each matrix operator can become intractable, it is useful to summarize their definitions, shown in Table 3.1, below.

Table 3.1: Matrix Operator Definitions

Matrix Operator	Definition
$\boldsymbol{J_\phi}$	The effect of the cell averaged distributed source in each cell on the cell averaged uncollided scalar flux in all cells
$\boldsymbol{J_\psi}$	The effect of the cell averaged distributed source in each cell on the outgoing angular flux on all external faces
$\boldsymbol{K_\phi}$	The effect of the incoming angular flux on each external face on the cell averaged uncollided scalar flux in all cells
$\boldsymbol{K_\psi}$	The effect of the incoming angular flux on each external face on the outgoing angular flux on all external faces

3.1 Consideration of Anisotropic Scattering

Consideration of anisotropic scattering in equations 3.18 and 3.19 does not alter the equations. It does, however, significantly alter the contents of the vector $\boldsymbol{\phi}^{\infty}$, and therefore the contents of the matrix operators $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{J_\phi}$. A lengthy derivation by Zerr [4], avoided here for brevity, provides that the scattering source is given by:

$$q_s = \sum_{l=0}^{L}\sum_{m=0}^{l} \sigma_{sl}(2-\delta_{m0})[Y_{lm}^{e}(\hat{\Omega})\varphi_l^m + Y_{lm}^{o}(\hat{\Omega})\vartheta_l^m] \tag{3.20}$$

In the case of isotropic scattering (i.e. $L = 0$), it can be seen that the scattering source reduces to

$$q_s = \sigma_{s,0}\phi_0^o \tag{3.21}$$

as expected. The formation of the anisotropic scattering source, however, requires significant alterations to the matrix equation 3.6.

Considering equation 3.20, equation 3.6 becomes [4]

المنارة للاستشارات
www.manaraa.com

$$\begin{bmatrix} \sigma_{tijk} & |\xi|/\Delta z_k & |\eta|/\Delta y_j & |\mu|/\Delta x_i \\ 1 & -0.5 & 0 & 0 \\ 1 & 0 & -0.5 & 0 \\ 1 & 0 & 0 & -0.5 \end{bmatrix} \begin{bmatrix} \psi_{n,i,j,k} \\ \psi_{n,i,j,k_{out}} \\ \psi_{n,i,j_{out},k} \\ \psi_{n,i_{out},j,k} \end{bmatrix} =$$

$$\begin{bmatrix} Y^e_{00} & Y^e_{10} & 2Y^e_{11} & 2Y^o_{11} & \cdots & 2Y^o_{LL} & |\xi|/\Delta z_k & |\eta|/\Delta y_j & |\mu|/\Delta x_i \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0.5 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0.5 \end{bmatrix} \begin{bmatrix} \sigma_{s0}\varphi^0_0 + q^e_{00} \\ \sigma_{s1}\varphi^0_1 + q^e_{10} \\ \sigma_{s1}\varphi^1_1 + q^e_{11} \\ \sigma_{s1}\vartheta^1_1 + q^o_{11} \\ \vdots \\ \sigma_{sL}\vartheta^L_L + q^o_{LL} \\ \psi_{n,i,j,k_{in}} \\ \psi_{n,i,j_{in},k} \\ \psi_{n,i_{in},j,k} \end{bmatrix} \quad (3.22)$$

Inverting the left hand side coefficient matrix and left multiplying both both sides by it yields a new formulation for the matrix, Γ, thus generalizing the Γ matrix to Γ_{anis}

$$\begin{bmatrix} \psi_{n,i,j,k} \\ \psi_{n,i,j,k_{out}} \\ \psi_{n,i,j_{out},k} \\ \psi_{n,i_{out},j,k} \end{bmatrix} = \Gamma_{anis} \begin{bmatrix} \sigma_{s0}\varphi^0_0 + q^e_{00} \\ \sigma_{s1}\varphi^0_1 + q^e_{10} \\ \sigma_{s1}\varphi^1_1 + q^e_{11} \\ \sigma_{s1}\vartheta^1_1 + q^o_{11} \\ \vdots \\ \sigma_{sL}\vartheta^L_L + q^o_{LL} \\ \psi_{n,i,j,k_{in}} \\ \psi_{n,i,j_{in},k} \\ \psi_{n,i_{in},j,k} \end{bmatrix} \quad (3.23)$$

The Γ_{anis} matrix has dimensions 4 x $((L+1)^2+3)$ and is a set of coupling factors between each angular moment of the scattering plus fixed source, incoming fluxes to the cell, and outgoing face angular fluxes. The effect of including anisotropic scattering on the dimensions of each matrix operator and, ultimately, the effect on construction algorithm performance will be examined in chapter 4.

The DMS algorithm for the construction of the four matrix operators, $\boldsymbol{J_\phi}$, $\boldsymbol{J_\psi}$, $\boldsymbol{K_\phi}$, and , $\boldsymbol{K_\psi}$ was developed by Azmy [16,17] and Zerr [4]. The algorithm developed in this work is intended to improve upon the DMS by accomplishing two goals: Be more geometrically robust and less

المنارة للاستشارات
www.manaraa.com

computationally expensive. In consideration of these goals, this algorithm was developed on the foundation of the matrix equation 3.6 and 3.23 for isotropic and anisotropic scattering, respectively.

3.2 The Differential Mesh Sweep (DMS) Algorithm

As mentioned earlier in this chapter, the DMS algorithm is the existing method of constructing the ITMM matrix operators. For the purpose of comparison to the new algorithm developed here, a summary of the DMS algorithm for the isotropic scattering case is described in this section. A much more detailed explanation of the algorithm is provided by Zerr [4]. Zerr's description of the algorithm will be used, including naming conventions. As such, it is important to note Zerr's naming convention of the $\mathbf{\Gamma}$ matrix from equation 3.6 compared to the naming convention used in this work from equation 3.7,

$$\mathbf{\Gamma}_{\boldsymbol{n}} = \begin{bmatrix} j_\phi & k_{\phi,z} & k_{\phi,y} & k_{\phi,x} \\ j_{\psi,z} & k_{\psi,z\to z} & k_{\psi,y\to z} & k_{\psi,x\to z} \\ j_{\psi,y} & k_{\psi,z\to y} & k_{\psi,y\to y} & k_{\psi,x\to y} \\ j_{\psi,x} & k_{\psi,z\to x} & k_{\psi,y\to x} & k_{\psi,x\to x} \end{bmatrix} = \begin{bmatrix} \gamma^{aa} & \gamma^{axy} & \gamma^{axz} & \gamma^{ayz} \\ \gamma^{xyz} & \gamma^{xyxy} & \gamma^{xyxz} & \gamma^{xyyz} \\ \gamma^{xza} & \gamma^{xzxy} & \gamma^{xzxz} & \gamma^{xzyz} \\ \gamma^{yza} & \gamma^{yzxy} & \gamma^{yzxz} & \gamma^{yzyz} \end{bmatrix}, \tag{3.24}$$

where the superscripts are indicative of the relation between the element, the vector to which it is contributing, and the vector which it is multiplying. For the remainder of this section, this naming convention for the elements of Γ will be used, while the previous naming convention will be resumed in the next chapter to the describe the new algorithm.

The beginning of the DMS algorithm is the partial differentiation of equation 3.6 in all cells (i,j,k) with respect to the previous iterate of the scalar flux in each cell (i',j',k') , resulting in [4]

$$\frac{\partial\psi_{n,i,j,k}}{\partial\phi^p_{i',j',k'}} = \gamma^{aa}_{nijk} c_{ijk} \frac{\partial\phi^p_{i,j,k}}{\partial\phi^p_{i',j',k'}} + \gamma^{axy}_{nijk} \frac{\partial\psi_{n,i,j,k_{in}}}{\partial\phi^p_{i',j',k'}} + \gamma^{axz}_{nijk} \frac{\partial\psi_{n,i,j_{in},k}}{\partial\phi^p_{i',j',k'}} + \gamma^{ayz}_{nijk} \frac{\partial\psi_{n,i_{in},j,k}}{\partial\phi^p_{i',j',k'}} \tag{3.25}$$

and

$$\frac{\partial\psi_{n,i_{out},j,k}}{\partial\phi^p_{i',j',k'}} = \gamma^{yza}_{nijk} c_{ijk} \frac{\partial\phi^p_{i,j,k}}{\partial\phi^p_{i',j',k'}} + \gamma^{yzxy}_{nijk} \frac{\partial\psi_{n,i,j,k_{in}}}{\partial\phi^p_{i',j',k'}} + \gamma^{yzxz}_{nijk} \frac{\partial\psi_{n,i,j_{in},k}}{\partial\phi^p_{i',j',k'}} + \gamma^{yzyz}_{nijk} \frac{\partial\psi_{n,i_{in},j,k}}{\partial\phi^p_{i',j',k'}} \tag{3.26}$$

, AFyz. The elements calculated by equation 3.26 and AFyz are accumulated by the DMS algorithm into three matrices as the mesh sweep through the sub-domain is occurring: $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$, respectively, such that the elements of these matrices for the adjacent cells to the

www.manaraa.com

starting cell of the sweep are [4]

$$X_{(i+1,j,k)(i',j',k')} = \frac{\partial \psi_{n,i_{out},j,k}}{\partial \phi^p_{i',j',k'}} \ , \tag{3.27}$$

$$Y_{(i,j+1,k)(i',j',k')} = \frac{\partial \psi_{n,i,j_{out},k}}{\partial \phi^p_{i',j',k'}} \ , \tag{3.28}$$

and

$$Z_{(i,j,k+1)(i',j',k')} = \frac{\partial \psi_{n,i,j,k_{out}}}{\partial \phi^p_{i',j',k'}} \ . \tag{3.29}$$

As the DMS continues, the elements of $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ are updated as [4]

$$X_{(i+1,j,k)(i',j',k')} = \gamma^{yzxy}_{nijk} Z_{(i,j,k)(i',j',k')} + \gamma^{yzxz}_{nijk} Y_{(i,j,k)(i',j',k')} + \gamma^{yzyz}_{nijk} X_{(i,j,k)(i',j',k')} \ , \tag{3.30}$$

$$Y_{(i+1,j,k)(i',j',k')} = \gamma^{xzxy}_{nijk} Z_{(i,j,k)(i',j',k')} + \gamma^{xzxz}_{nijk} Y_{(i,j,k)(i',j',k')} + \gamma^{xzyz}_{nijk} X_{(i,j,k)(i',j',k')} \ , \tag{3.31}$$

and

$$Z_{(i+1,j,k)(i',j',k')} = \gamma^{xyxy}_{nijk} Z_{(i,j,k)(i',j',k')} + \gamma^{xyxz}_{nijk} Y_{(i,j,k)(i',j',k')} + \gamma^{xyyz}_{nijk} X_{(i,j,k)(i',j',k')} \ . \tag{3.32}$$

Upon completion of the sweep, the terms of equation 3.25 are updated as

$$\frac{\partial \psi_{n,i,j,k}}{\partial \phi^p_{i',j',k'}} = \gamma^{ayz}_{nijk} X_{(i,j,k)(i',j',k')} + \gamma^{axz}_{nijk} Y_{(i,j,k)(i',j',k')} + \gamma^{axy}_{nijk} Z_{(i,j,k)(i',j',k')} \ . \tag{3.33}$$

The elements of $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ are then used to update the ITMM matrix operator $\boldsymbol{J_\phi}$ using the quadrature sum from equation 3.3

$$J_{\phi(i,j,k)(i',j',k')} = \sum_{n=1}^{D} w_n \frac{\partial \psi_{n,i,j,k}}{\partial \phi^p_{i',j',k'}} \tag{3.34}$$

The elements of the ITMM matrix operator $\boldsymbol{J_\psi}$ are also calculated using the $\boldsymbol{X}$, $\boldsymbol{Y}$, and $\boldsymbol{Z}$ as, for angular fluxes leaving the sub-domain in the x-direction,

$$J_{\psi,n,(i,j,k)(i',j',k')} = X_{(i+1,j,k)(i',j',k')}, \text{ AFyz.} \tag{3.35}$$

As stated in chapter 2, Zerr also developed the algorithm for construction of the ITMM matrix operators necessary to operate on the incoming angular flux to the sub-domain. To this end, he created nine additional matrices: $\boldsymbol{XBCX}$, $\boldsymbol{XBCY}$, $\boldsymbol{XBCZ}$, $\boldsymbol{YBCX}$, $\boldsymbol{YBCY}$, $\boldsymbol{YBCZ}$, $\boldsymbol{ZBCX}$, $\boldsymbol{ZBCY}$, and $\boldsymbol{ZBCZ}$. These matrices hold values representing the effect of an entering angular flux to the sub-domain in the constant face at the beginning of the matrix name (i.e.

www.manaraa.com

x, y, or z) on the angular flux leaving all cells in the sub-domain out of the constant face at the end of the matrix name. Considering an angular flux entering the sub-domain in the x-direction in the primary octant. The values of the appropriate matrices for the adjacent cells to the incoming angular flux are [4]

$$XBCX_{(2,j,k)(1,j,k)} = \gamma^{yzyz}_{(1,j,k)} \; , \tag{3.36}$$

$$XBCY_{(1,j+1,k)(1,j,k)} = \gamma^{xzyz}_{(1,j,k)} \; , \tag{3.37}$$

and

$$XBCZ_{(1,j,k+1)(1,j,k)} = \gamma^{xyyz}_{(1,j,k)} \; . \tag{3.38}$$

Updated recursively as the sweep continues, these values become

$$\begin{aligned} XBCX_{(i+1,j,k)(1,j',k')} = \gamma^{yzyz}_{(i,j,k)} XBCX_{(i,j,k)(1,j',k')} + \\ \gamma^{yzxz}_{(i,j,k)} XBCY_{(i,j,k)(1,j',k')} + \gamma^{yzzy}_{(i,j,k)} XBCZ_{(i,j,k)(1,j',k')} \; , \end{aligned} \tag{3.39}$$

$$\begin{aligned} XBCY_{(i,j+1,k)(1,j',k')} = \gamma^{xzyz}_{(i,j,k)} XBCX_{(i,j,k)(1,j',k')} + \\ \gamma^{xzxz}_{(i,j,k)} XBCY_{(i,j,k)(1,j',k')} + \gamma^{xzxy}_{(i,j,k)} XBCZ_{(i,j,k)(1,j',k')} \; , \end{aligned} \tag{3.40}$$

and

$$\begin{aligned} XBCZ_{(i,j,k+1)(1,j',k')} = \gamma^{xyyz}_{(i,j,k)} XBCX_{(i,j,k)(1,j',k')} + \\ \gamma^{xyxz}_{(i,j,k)} XBCY_{(i,j,k)(1,j',k')} + \gamma^{xyxy}_{(i,j,k)} XBCZ_{(i,j,k)(1,j',k')} \; . \end{aligned} \tag{3.41}$$

Analogous relations exist for angular flux entering the sub-domain in the y and z directions. The elements of these nine matrices are then used to create the remaining two ITMM matrix operators for an angular flux entering sub-domain in the x-direction in the manner [4]

$$\begin{aligned} K_{\phi,n,(i,j,k)(1,j',k')} = w_n \gamma^{ayz}_{(i,j,k)} XBCX_{(i,j,k)(1,j',k')} + \\ w_n \gamma^{axz}_{(i,j,k)} XBCY_{(i,j,k)(1,j',k')} + w_n \gamma^{axy}_{(i,j,k)} XBCZ_{(i,j,k)(1,j',k')} \end{aligned} \tag{3.42}$$

and

$$K_{\psi,n,(i,j,k)(1,j',k')} = XBCX_{(i,j,k)(1,j',k')} \tag{3.43}$$

with analogous relations for angular fluxes entering the sub-domain in the y and z directions.

With a summary of the DMS algorithm now provided, the next chapter will describe the new

المنارة للاستشارات
www.manaraa.com

algorithm and highlight several differences between the two.

www.manaraa.com

Chapter 4

The Fundamental Matrix Method Algorithm

As mentioned previously, the motivation to create a new ITMM matrix operator construction algorithm was driven by the desire for a less computationally expensive and more geometrically robust algorithm. With these objectives in mind, consideration of a single cell system and the associated coupling factors shown in the Γ matrix was an ideal starting point due to its simplicity.

Beginning with equation 3.6, partial derivatives are taken with respect to each incoming angular flux component and the cell averaged scalar flux. First, partially differentiating equation 3.6 with respect to the cell averaged scalar flux:

$$\begin{bmatrix} \frac{\partial \psi_{n,i,j,k}}{\partial \phi^{p}_{i,j,k}} \\ \frac{\partial \psi_{n,i,j,k_{out}}}{\partial \phi^{p}_{i,j,k}} \\ \frac{\partial \psi_{n,i,j_{out},k}}{\partial \phi^{p}_{i,j,k}} \\ \frac{\partial \psi_{n,i_{out},j,k}}{\partial \phi^{p}_{i,j,k}} \end{bmatrix} = \begin{bmatrix} j_{\phi_{i,j,k}} c_{i,j,k} \\ j_{\psi,z_{i,j,k}} c_{i,j,k} \\ j_{\psi,y_{i,j,k}} c_{i,j,k} \\ j_{\psi,x_{i,j,k}} c_{i,j,k} \end{bmatrix} \tag{4.1}$$

Partially differentiating equation 3.6 with respect to the incoming angular flux in the x direction yields

$$\begin{bmatrix} \frac{\partial \psi_{n,i,j,k}}{\partial \psi_{n,i_{in},j,k}} \\ \frac{\partial \psi_{n,i,j,k_{out}}}{\partial \psi_{n,i_{in},j,k}} \\ \frac{\partial \psi_{n,i,j_{out},k}}{\partial \psi_{n,i_{in},j,k}} \\ \frac{\partial \psi_{n,i_{out},j,k}}{\partial \psi_{n,i_{in},j,k}} \end{bmatrix} = \begin{bmatrix} k_{\phi,x_{i,j,k}} \\ k_{\psi,x\rightarrow z_{i,j,k}} \\ k_{\psi,x\rightarrow y_{i,j,k}} \\ k_{\psi,x\rightarrow x_{i,j,k}} \end{bmatrix}, \text{AFyz.} \tag{4.2}$$

www.manaraa.com

To create a general framework for the response of the angular and scalar flux in one cell of a domain due to the production and in-leakage at another cell, only the bottom three equations of matrix equation 4.2 and the analogous equations in the y and z directions are used. As such, only the mapping of the angular flux from incoming to outgoing faces is being considered. These equations, however, also only consider a single cell system. To consider an entire domain, these single cell operators need to be accumulated along every possible path of particles from the starting to destination cells. This accumulation of values representing the response of the angular flux on an incoming face of one cell of a domain to the angular flux on the outgoing face of another cell is termed the fundamental transport matrix and denoted by $\boldsymbol{F}$. The elements of $\boldsymbol{F}$ are represented by

$$F_{n(i,j,k)(i',j',k'),di,df} = \sum_{p=1}^{P} \prod_{m=1}^{M} k_{\psi,d1_{m,p} \to d2_{m,p},n,m,p} \tag{4.3}$$

where p is a possible path from cell i', j', k' to cell i, j, k and di and df are the incident and emergent constant faces (x, y, or z) of the element of $\boldsymbol{F}$. The total number of possible paths is P, a value which depends on the geometry of the domain, and the total number of cells along each respective path is M. The subscripts of k_ψ, $d1_{m,p}$ and $d2_{m,p}$ are either x, y, or z depending on which constant face the particle is incident on and emergent at, respectively, for cell m of path p. For the first cell of path p, $d1_{1,p} = di$, and for the last cell of path p, $d2_{M,p} = df$. $k_{\psi,d1_{m,p} \to d2_{m,p},n,m,p}$ is then the value of k_ψ from face $d1$ to face $d2$ in cell m along path p for the n^{th} discrete ordinate.

Although the FMM algorithm has been completed in three dimensions, figures 4.1 through 4.4, below, show a two dimensional depiction, for visual clarity, of the particle transport represented by $F_{n(i,j)(i',j'),di,df}$.

www.manaraa.com

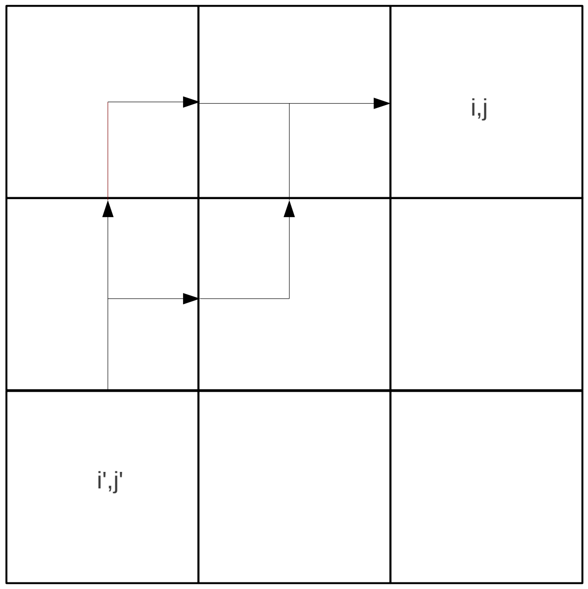


Figure 4.1: Two Dimensional Representation of $F_{n(i,j)(i',j'),y,x}$ for an Ordinate in the Primary Octant

www.manaraa.com

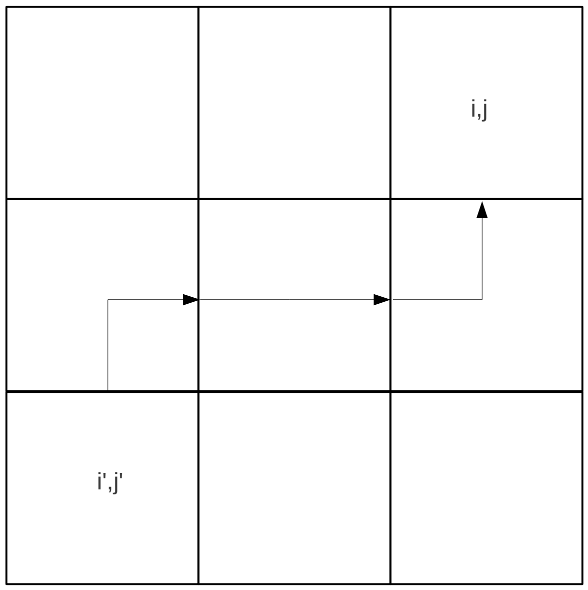


Figure 4.2: Two Dimensional Representation of $F_{n(i,j)(i',j'),y,y}$ for an Ordinate in the Primary Octant

المنارة للاستشارات
www.manaraa.com

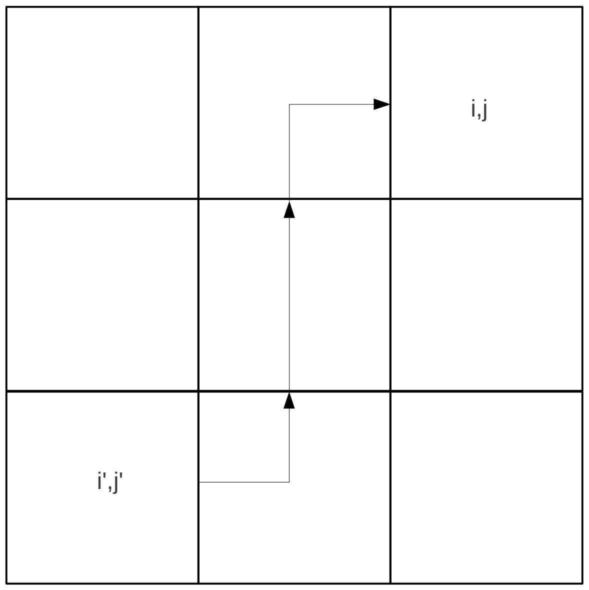


Figure 4.3: Two Dimensional Representation of $F_{n(i,j)(i',j'),x,x}$ for an Ordinate in the Primary Octant

www.manaraa.com

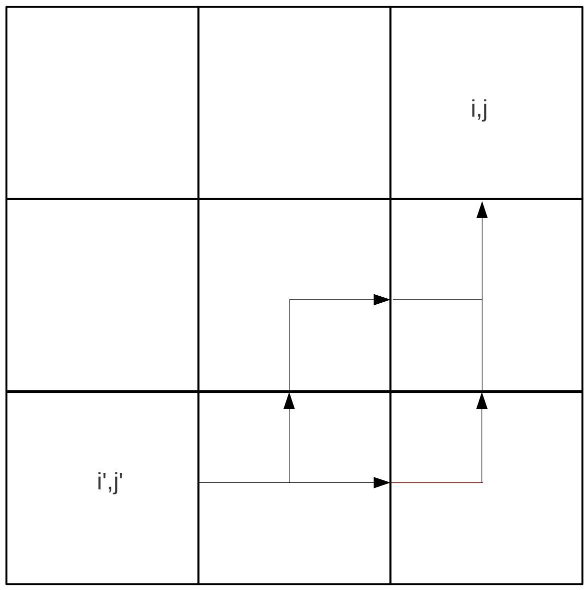


Figure 4.4: Two Dimensional Representation of $F_{n(i,j)(i',j'),x,y}$ for an Ordinate in the Primary Octant

It can be seen from a three dimensional extension of figures 4.1 through 4.4 how the element $F_{n(i,j,k)(i',j',k'),di,df}$ can then be used to construct each of the four ITMM matrix operators in a computationally efficient manner by avoiding repeat calculations since it represents the relation of the outgoing angular flux at any face in a domain to the incoming flux at any other face.

For an angular flux emergent from the face of one cell and incident on the face of another cell in a three dimensional system, there are nine distinct elements of $\boldsymbol{F}$ resulting from three possible emergent faces and three possible incident faces. To accumulate values into each ITMM operator, the only remaining calculation (after the calculation of the appropriate element of $\boldsymbol{F}$) is to multiply $F_{n(i,j,k)(i',j',k'),di,df}$ by the respective values it is required to represent for the emergent and incident cells. Recalling that the emergent cell is i', j', k' and the incident cell is i, j, k, the respective operators require multiplication of $F_{n(i,j,k)(i',j',k'),di,df}$ by the single cell operators in the manner shown in Table 4.1.

www.manaraa.com

Table 4.1: Emergent and Incident Multipliers of $F_{n(i,j,k)(i',j',k'),di,df}$ for Construction of ITMM Operators

Operator	Emergent Cell Multiplier	Incident Cell Multiplier
$\boldsymbol{J_\phi}$	$j_{\psi,di,i',j',k'}c_{i',j',k'}$	$k_{\phi,df,i,j,k}$
$\boldsymbol{J_\psi}$	$j_{\psi,di,i',j',k'}c_{i',j',k'}$	$k_{\psi,d1\rightarrow df,i,j,k}$
$\boldsymbol{K_\phi}$	$k_{\psi,d1\rightarrow di,i',j',k'}$	$k_{\phi,df,i,j,k}$
$\boldsymbol{K_\psi}$	$k_{\psi,d1\rightarrow di,i',j',k'}$	$k_{\psi,d1\rightarrow df,i,j,k}$

From equations 3.6 and 3.7, it can be seen how each single cell operator shown in Table 4.1 has a different value based on the direction it is required to represent, hence the necessity of nine distinct elements of $\boldsymbol{F}$ representing particle transport from the faces of the emergent cell to the to the faces of the incident cell in a three dimensional system.

The calculation of each element of $\boldsymbol{F}$ may appear to be computationally expensive due to the sheer number of paths that a particle can travel from an emergent face to an incident face, especially as the number of intervening cells grows. The FMM algorithm, however, avoids repeat calculations in the elements of $\boldsymbol{F}$ (just as it avoids repeat calculations for matrix operator elements by using $\boldsymbol{F}$) by recognizing that each path is simply an extension of a previously traversed path whose $\boldsymbol{F}$ elements have already been computed. A path to a face of an adjacent cell from a face of a cell for which the element of $\boldsymbol{F}$ has already been calculated only needs the respective elements of $\boldsymbol{F}$ to be multiplied by the appropriate value of k_ψ for each adjacent cell. In this manner, single cell operators, or, more specifically, single cell values of k_ψ, are compounded along a path to create an element of $\boldsymbol{F}$.

Each ITMM operator can then be constructed from elements of $\boldsymbol{F}$ as follows.

The matrix operator $\boldsymbol{J_\phi}$, of dimensions (I x J x K) x (I x J x K)

$$\boldsymbol{J_\phi} = \begin{bmatrix} j_{\phi,1,1,1}c_{1,1,1} & \cdots & j_{\psi,I,J,K}c_{I,J,K}F_{n(1,1,1)(I,J,K)}k_{\phi,1,1,1} \\ \vdots & \ddots & \vdots \\ j_{\psi,1,1,1}c_{1,1,1}F_{n(I,J,K)(1,1,1)}k_{\phi,I,J,K} & \cdots & j_{\phi,I,J,K}c_{I,J,K} \end{bmatrix} \tag{4.4}$$

www.manaraa.com

where

$$
\begin{aligned}
&j_{\psi,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k')}k_{\phi,(i,j,k)} = j_{\psi,x,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),x,x}k_{\phi,x,(i,j,k)}+\\
&j_{\psi,x,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),x,y}k_{\phi,y,(i,j,k)} + j_{\psi,x,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),x,z}k_{\phi,z,(i,j,k)}+\\
&j_{\psi,y,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),y,x}k_{\phi,x,(i,j,k)} + j_{\psi,y,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),y,y}k_{\phi,y,(i,j,k)}+\\
&j_{\psi,y,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),y,z}k_{\phi,z,(i,j,k)} + j_{\psi,z,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),z,x}k_{\phi,x,(i,j,k)}+\\
&j_{\psi,z,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),z,y}k_{\phi,y,(i,j,k)} + j_{\psi,z,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),z,z}k_{\phi,z,(i,j,k)}.
\end{aligned}
\tag{4.5}
$$

It is important to note that diagonal elements of $\boldsymbol{J_\phi}$ do not use an element of $\boldsymbol{F}$ in their construction. This is because the diagonal elements represent the effect of the cell averaged distributed source in a cell on the cell averaged scalar flux in the same cell. Since there is no transport to account for between cells, no element of $\boldsymbol{F}$ is used and the single cell operator j_ϕ is used instead of j_ψ on the diagonal elements.

The matrix operator $\boldsymbol{J_\psi}$, of dimensions (D) x 2(IJ + JK + IK) x (I x J x K)

$$
\boldsymbol{J_\psi} = \begin{bmatrix}
j_{\psi,n,1,1,1}c_{1,1,1}F_{n(1,1,1)(1,1,1)}k_{\psi,n,1,1,1} & \cdots & j_{\psi,n,I,J,K}c_{I,J,K}F_{n(1,1,1)(I,J,K)}k_{\psi,n,1,1,1}\\
\vdots & \cdots & \vdots\\
j_{\psi,n,1,1,1}c_{1,1,1}F_{n(I,J,K)(1,1,1)}k_{\psi,n,I,J,K} & \cdots & j_{\psi,n,I,J,K}c_{I,J,K}F_{n(I,J,K)(I,J,K)}k_{\psi,n,I,J,K}\\
\vdots & \cdots & \vdots\\
j_{\psi,N,1,1,1}c_{1,1,1}F_{N(1,1,1)(1,1,1)}k_{\psi,N,1,1,1} & \cdots & j_{\psi,N,I,J,K}c_{I,J,K}F_{N(1,1,1)(I,J,K)}k_{\psi,N,1,1,1}\\
\vdots & \cdots & \vdots\\
j_{\psi,N,1,1,1}c_{1,1,1}F_{N(I,J,K)(1,1,1)}k_{\psi,N,I,J,K} & \cdots & j_{\psi,N,I,J,K}c_{I,J,K}F_{N(I,J,K)(I,J,K)}k_{\psi,N,I,J,K}
\end{bmatrix}
\tag{4.6}
$$

where, if considering the exiting x-face of the final cell and analogously for the exiting y-face and z-face of the final cell,

$$
\begin{aligned}
&j_{\psi,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k')}k_{\psi,(i,j,k)} = j_{\psi,x,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),x,x}k_{\psi,x\to x,(i,j,k)}+\\
&j_{\psi,x,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),x,y}k_{\psi,y\to x,(i,j,k)}+j_{\psi,x,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),x,z}k_{\psi,z\to x,(i,j,k)}+\\
&j_{\psi,y,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),y,x}k_{\psi,x\to x,(i,j,k)}+j_{\psi,y,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),y,y}k_{\psi,y\to x,(i,j,k)}+\\
&j_{\psi,y,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),y,z}k_{\psi,z\to x,(i,j,k)}+j_{\psi,z,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),z,x}k_{\psi,x\to x,(i,j,k)}+\\
&j_{\psi,z,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),z,y}k_{\psi,y\to x,(i,j,k)}+j_{\psi,z,(i',j',k')}c_{i',j',k'}F_{n(i,j,k)(i',j',k'),z,z}k_{\psi,z\to x,(i,j,k)}.
\end{aligned}
\tag{4.7}
$$

The matrix operator $\boldsymbol{K_\phi}$, of dimensions transpose of $\boldsymbol{J_\psi}$, (I x J x K) x 2(IJ + JK + IK) x

المنارة للاستشارات
www.manaraa.com

(D)

$$\boldsymbol{K_\phi} = \begin{bmatrix} k_{\psi,n,1,1,1}F_{n(1,1,1)(1,1,1)}k_{\phi,n,1,1,1} & \cdots & k_{\psi,n,1,1,1}F_{n(1,1,1)(I,J,K)}k_{\phi,n,1,1,1} \\ \vdots & \cdots & \vdots \\ k_{\psi,n,1,1,1}F_{n(I,J,K)(1,1,1)}k_{\phi,n,I,J,K} & \cdots & k_{\psi,n,I,J,K}F_{n(I,J,K)(I,J,K)}k_{\phi,n,I,J,K} \\ \vdots & \cdots & \vdots \\ k_{\psi,N,1,1,1}F_{N(1,1,1)(1,1,1)}k_{\phi,N,1,1,1} & \cdots & k_{\psi,N,I,J,K}F_{N(1,1,1)(I,J,K)}k_{\phi,N,1,1,1} \\ \vdots & \cdots & \vdots \\ k_{\psi,N,1,1,1}F_{N(I,J,K)(1,1,1)}k_{\phi,N,I,J,K} & \cdots & k_{\psi,N,I,J,K}F_{N(I,J,K)(I,J,K)}k_{\phi,N,I,J,K} \end{bmatrix}^T \tag{4.8}$$

where, if considering the entering x-face of the initial cell and analogously for the entering y-face and z-face of the initial cell,

$$\begin{aligned} k_{\psi,(i',j',k')}F_{n(i,j,k)(i',j',k')}k_{\phi,(i,j,k)} = {} & k_{\psi,x\to x,(i',j',k')}F_{n(i,j,k)(i',j',k'),x,x}k_{\phi,x,(i,j,k)} + \\ & k_{\psi,x\to x,(i',j',k')}F_{n(i,j,k)(i',j',k'),x,y}k_{\phi,y,(i,j,k)} + k_{\psi,x\to x,(i',j',k')}F_{n(i,j,k)(i',j',k'),x,z}k_{\phi,z,(i,j,k)} + \\ & k_{\psi,x\to y,(i',j',k')}F_{n(i,j,k)(i',j',k'),y,x}k_{\phi,x,(i,j,k)} + k_{\psi,x\to y,(i',j',k')}F_{n(i,j,k)(i',j',k'),y,y}k_{\phi,y,(i,j,k)} + \\ & k_{\psi,x\to y,(i',j',k')}F_{n(i,j,k)(i',j',k'),y,z}k_{\phi,z,(i,j,k)} + k_{\psi,x\to z,(i',j',k')}F_{n(i,j,k)(i',j',k'),z,x}k_{\phi,x,(i,j,k)} + \\ & k_{\psi,x\to z,(i',j',k')}F_{n(i,j,k)(i',j',k'),z,y}k_{\phi,y,(i,j,k)} + k_{\psi,x\to z,(i',j',k')}F_{n(i,j,k)(i',j',k'),z,z}k_{\phi,z,(i,j,k)}. \end{aligned} \tag{4.9}$$

The matrix operator $\boldsymbol{K_\psi}$, of dimensions 2(IJ + JK + IK) x (D) x 2(IJ + JK + IK) x (D)

$$\boldsymbol{K_\psi} = \begin{bmatrix} k_{\psi,n,1,1,1}F_{n(1,1,1)(1,1,1)}k_{\psi,n,1,1,1} & \cdots & k_{\psi,n,1,1,1}F_{n(1,1,1)(I,J,K)}k_{\psi,n,1,1,1} \\ \vdots & \cdots & \vdots \\ k_{\psi,n,1,1,1}F_{n(I,J,K)(1,1,1)}k_{\psi,n,I,J,K} & \cdots & k_{\psi,n,I,J,K}F_{n(I,J,K)(I,J,K)}k_{\psi,n,I,J,K} \\ \vdots & \cdots & \vdots \\ k_{\psi,N,1,1,1}F_{N(1,1,1)(1,1,1)}k_{\psi,N,1,1,1} & \cdots & k_{\psi,N,I,J,K}F_{N(1,1,1)(I,J,K)}k_{\psi,N,1,1,1} \\ \vdots & \cdots & \vdots \\ k_{\psi,N,1,1,1}F_{N(I,J,K)(1,1,1)}k_{\psi,N,I,J,K} & \cdots & k_{\psi,N,I,J,K}F_{N(I,J,K)(I,J,K)}k_{\psi,N,I,J,K} \end{bmatrix}^T \tag{4.10}$$

where, if considering the entering x-face of the initial cell and and the exiting x-face of the final cell and analogously for the entering y-face and z-face of the initial cell and exiting y-face and

www.manaraa.com

z-face of the final cell,

$$\begin{multline}
k_{\psi,(i',j',k')}F_{n(i,j,k)(i',j',k')}k_{\psi,(i,j,k)} = k_{\psi,x\to x,(i',j',k')}F_{n(i,j,k)(i',j',k'),x,x}k_{\psi,x\to x,(i,j,k)}+\\
k_{\psi,x\to x,(i',j',k')}F_{n(i,j,k)(i',j',k'),x,y}k_{\psi,y\to x,(i,j,k)} + k_{\psi,x\to x,(i',j',k')}F_{n(i,j,k)(i',j',k'),x,z}k_{\psi,z\to x,(i,j,k)}+\\
k_{\psi,x\to y,(i',j',k')}F_{n(i,j,k)(i',j',k'),y,x}k_{\psi,x\to x,(i,j,k)} + k_{\psi,x\to y,(i',j',k')}F_{n(i,j,k)(i',j',k'),y,y}k_{\psi,y\to x,(i,j,k)}+\\
k_{\psi,x\to y,(i',j',k')}F_{n(i,j,k)(i',j',k'),y,z}k_{\psi,z\to x,(i,j,k)} + k_{\psi,x\to z,(i',j',k')}F_{n(i,j,k)(i',j',k'),z,x}k_{\psi,x\to x,(i,j,k)}+\\
k_{\psi,x\to z,(i',j',k')}F_{n(i,j,k)(i',j',k'),z,y}k_{\psi,y\to x,(i,j,k)} + k_{\psi,x\to z,(i',j',k')}F_{n(i,j,k)(i',j',k'),z,z}k_{\psi,z\to x,(i,j,k)}.
\end{multline} \tag{4.11}$$

In essence, we have consolidated the expensive, recursive calculations required to compute the four ITMM operators into the computation of $F_{n(i,j,k)(i',j',k')}$, followed by a straightforward and computationally cheap pre-fix and post-fix operations indicated in equations 4.4, 4.6, 4.8, and 4.10.

In comparison to the FMM, the DMS, as described in the previous chapter, requires twelve intermediate matrices in order to populate the four ITMM matrix operators. These are the three matrices with values representing the outgoing angular flux from every cell in the x, y, and z directions and the nine matrices representing the effect of incoming angular flux in each direction on the outgoing angular flux in each direction. These values are then used in the manner described in the previous chapter to create the four ITMM matrix operators. The FMM uses only a single intermediate matrix, $\boldsymbol{F}$, from which all the elements of the four ITMM matrix operators are created in a relatively simple manner.

The main reason that this difference is important is the time required for memory access. As each element of $\boldsymbol{F}$ is created, it is promptly operated on to create the corresponding elements of the four ITMM matrix operators, allowing for the element of $\boldsymbol{F}$ to remain in the memory cache for the necessary operations and then be discarded without another access. The DMS, however, creates the necessary twelve intermediate matrices and places them into memory, requiring that they be accessed at a later time to create the four ITMM matrix operators.

Another advantage of the FMM compared to the DMS is the nature of the mesh sweep. The DMS uses a single mesh sweep (per angle) to calculate the the required values to populate the aforementioned twelve matrices. In comparison, the FMM conducts a sweep from each cell of the mesh to all other cells. While seemingly a disadvantage for the FMM due to the quantity of sweeps required, the calculations are more simple in nature because only elements of $\boldsymbol{F}$ are being created in the sweep. The FMM, in this manner, is also more geometrically robust in that only knowledge of the spatial relation to the adjacent cells is required to create the elements of $\boldsymbol{F}$.

المنارة للاستشارات
www.manaraa.com

4.1 Example Construction of Elements of F for a 3 x 3 Cell Domain

In order to demonstrate how the four matrix operators are constructed from single-cell operators, consider a 3 x 3 cell domain as shown in figure 4.5. For clarity. this illustration employs a two dimensional domain; however, implementation of the new algorithm and the numerical tests reported in the following Chapter utilize a three-dimensional Cartesian grid and more numerous cells than this 3 x 3 example.

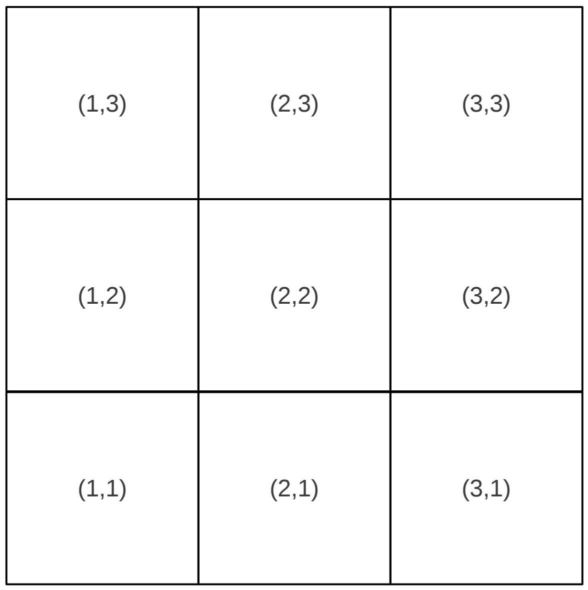


Figure 4.5: 3 x 3 Cell Domain with Numbered Cells

This example will demonstrate the construction of the ITMM matrix operators in the given domain with isotropic scattering for a single ordinate, $\hat{\Omega}_n$, in the primary octant. Each cell has its own matrix system of equations with the $\mathbf{\Gamma}$ matrix constructed from the ordinate values and the total cross section of the material in each cell. The system of equations for Cell (1,1) is,

www.manaraa.com

with analogous relations for the other cells,

$$\begin{bmatrix} \psi_{(1,1),n} \\ \psi_{(1,1),n,y_{out}} \\ \psi_{(1,1),n,x_{out}} \end{bmatrix} = \begin{bmatrix} j_{\phi,(1,1)} & k_{\phi,(1,1),y} & k_{\phi,(1,1),x} \\ j_{\psi,(1,1),y} & k_{\psi,(1,1),y\to y} & k_{\psi,(1,1),x\to y} \\ j_{\psi,(1,1),x} & k_{\psi,(1,1),y\to x} & k_{\psi,(1,1),x\to x} \end{bmatrix} \begin{bmatrix} c_{(1,1)}\phi^{p}_{(1,1)} + \sigma^{-1}_{t,(1,1)} q_{(1,1)} \\ \psi_{(1,1),n,y_{in}} \\ \psi_{(1,1),n,x_{in}} \end{bmatrix} \tag{4.12}$$

Between adjacent cells, the element of ***F*** representing neutron streaming from one cell to the next is unity because of the physically inspired condition imposing continuity of the angular flux across cell interfaces. The other (nonzero) elements of ***F*** are quantified by tracing the possible paths through the domain. It is important to distinguish the emergent face and the incident face because the emergent and incident multipliers will vary based on this distinction.

The following figures show a step-by-step process for calculating the elements of ***F*** and each of the four ITMM operators in the example 3 x 3 domain for an ordinate in the primary octant and emerging from cell (1,1). The tables following each figure give the value of every element of ***F*** and the four ITMM matrix operators calculated in each step of the FMM algorithm.

المنارة للاستشارات
www.manaraa.com

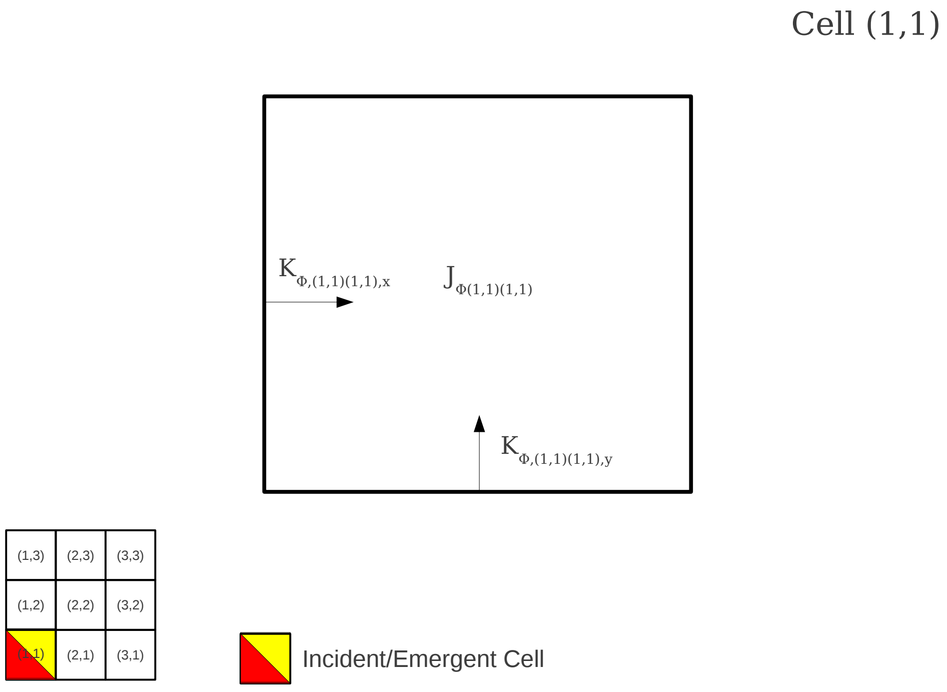


Figure 4.6: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (1,1) for an Ordinate in the Primary Octant

www.manaraa.com

Table 4.2: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J_\phi}$, $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{K_\psi}$ Resulting from an Angular Flux Incident on a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
N/A	N/A

Elements of $\boldsymbol{J_\phi}$ Calculated	
Element	Value
$J_{\phi,(1,1)(1,1)}$	$j_{\phi,(1,1)}c_{(1,1)}$

Elements of $\boldsymbol{K_\phi}$ Calculated	
Element	Value
$K_{\phi,(1,1)(1,1),x}$	$k_{\phi,(1,1),x}$
$K_{\phi,(1,1)(1,1),y}$	$k_{\phi,(1,1),y}$

Elements of $\boldsymbol{J_\psi}$ Calculated	
Element	Value
N/A	N/A

Elements of $\boldsymbol{K_\psi}$ Calculated	
Element	Value
N/A	N/A

المنارة للاستشارات

www.manaraa.com

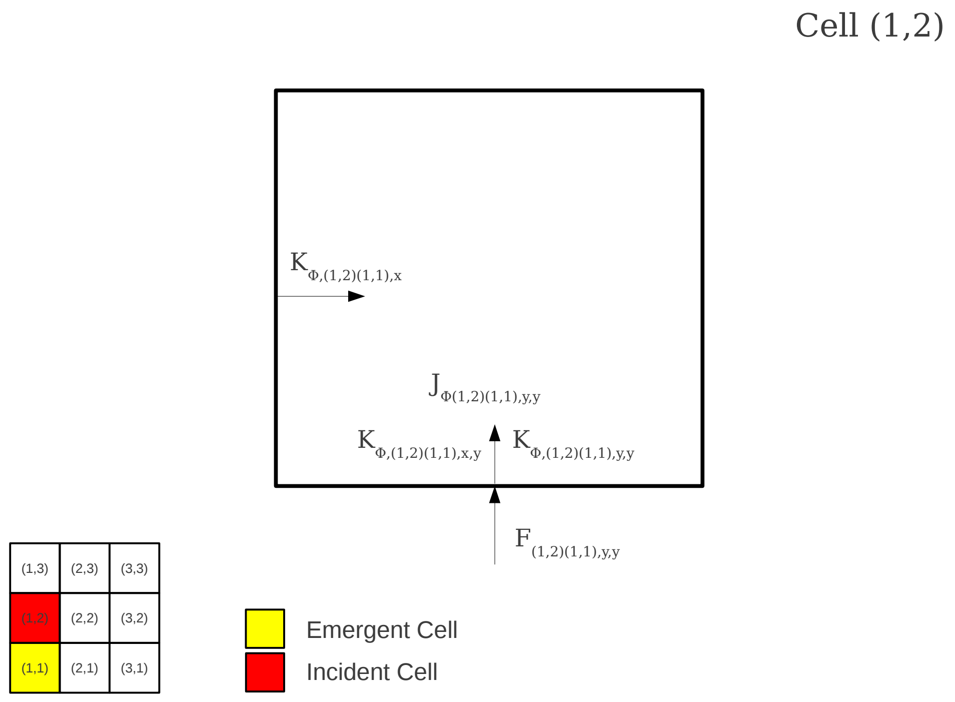


Figure 4.7: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (1,2) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

www.manaraa.com

Table 4.3: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J_\phi}$, $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{K_\psi}$ Resulting from an Angular Flux Incident on a Face of Cell (1,2) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
$F_{(1,2)(1,1),y,y}$	1
Elements of $\boldsymbol{J_\phi}$ Calculated	
Element	Value
$J_{\phi,(1,2)(1,1),y,y}$	$j_{\psi,(1,1)}c_{(1,1)}F_{(1,2)(1,1),y,y}k_{\phi,(1,2),y}$
Elements of $\boldsymbol{K_\phi}$ Calculated	
Element	Value
$K_{\phi,(1,2)(1,1),x,y}$	$k_{\psi,(1,1),x\to y}F_{(1,2)(1,1),y,y}k_{\phi,(1,2),y}$
$K_{\phi,(1,2)(1,1),y,y}$	$k_{\psi,(1,1),y\to y}F_{(1,2)(1,1),y,y}k_{\phi,(1,2),y}$
$K_{\phi,(1,2)(1,2),x}$	$k_{\phi,(1,2),x}$
Elements of $\boldsymbol{J_\psi}$ Calculated	
Element	Value
N/A	N/A
Elements of $\boldsymbol{K_\psi}$ Calculated	
Element	Value
N/A	N/A

www.manaraa.com

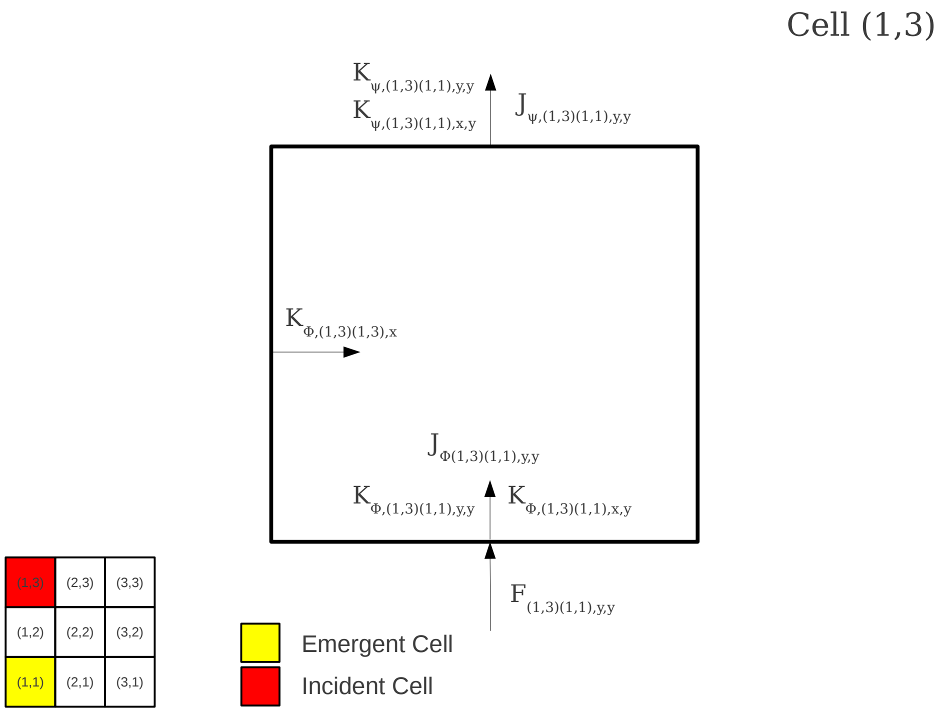


Figure 4.8: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (1,3) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

المنارة للاستشارات
www.manaraa.com

Table 4.4: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J_\phi}$, $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{K_\psi}$ Resulting from an Angular Flux Incident on a Face of Cell (1,3) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
$F_{(1,3)(1,1),y,y}$	$k_{\psi,(1,2),y\to y}$
Elements of $\boldsymbol{J_\phi}$ Calculated	
Element	Value
$J_{\phi,(1,3)(1,1),y,y}$	$j_{\psi,(1,1)}c_{(1,1)}F_{(1,3)(1,1),y,y}k_{\phi,(1,3),y}$
Elements of $\boldsymbol{K_\phi}$ Calculated	
Element	Value
$K_{\phi,(1,3)(1,1),x,y}$	$k_{\psi,(1,1),x\to y}F_{(1,3)(1,1),y,y}k_{\phi,(1,3),y}$
$K_{\phi,(1,3)(1,1),y,y}$	$k_{\psi,(1,1),y\to y}F_{(1,3)(1,1),y,y}k_{\phi,(1,3),y}$
$K_{\phi,(1,3)(1,3),x}$	$k_{\phi,(1,3),x}$
Elements of $\boldsymbol{J_\psi}$ Calculated	
Element	Value
$J_{\psi,(1,3)(1,1),y,y}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(1,3)(1,1),y,y}k_{\psi,(1,3),y\to y}$
Elements of $\boldsymbol{K_\psi}$ Calculated	
Element	Value
$K_{\psi,(1,3)(1,1),x,y}$	$k_{\psi,(1,1),x\to y}F_{(1,3)(1,1),y,y}k_{\psi,(1,3),y\to y}$
$K_{\psi,(1,3)(1,1),y,y}$	$k_{\psi,(1,1),y\to y}F_{(1,3)(1,1),y,y}k_{\psi,(1,3),y\to y}$

المنارة للاستشارات
www.manaraa.com

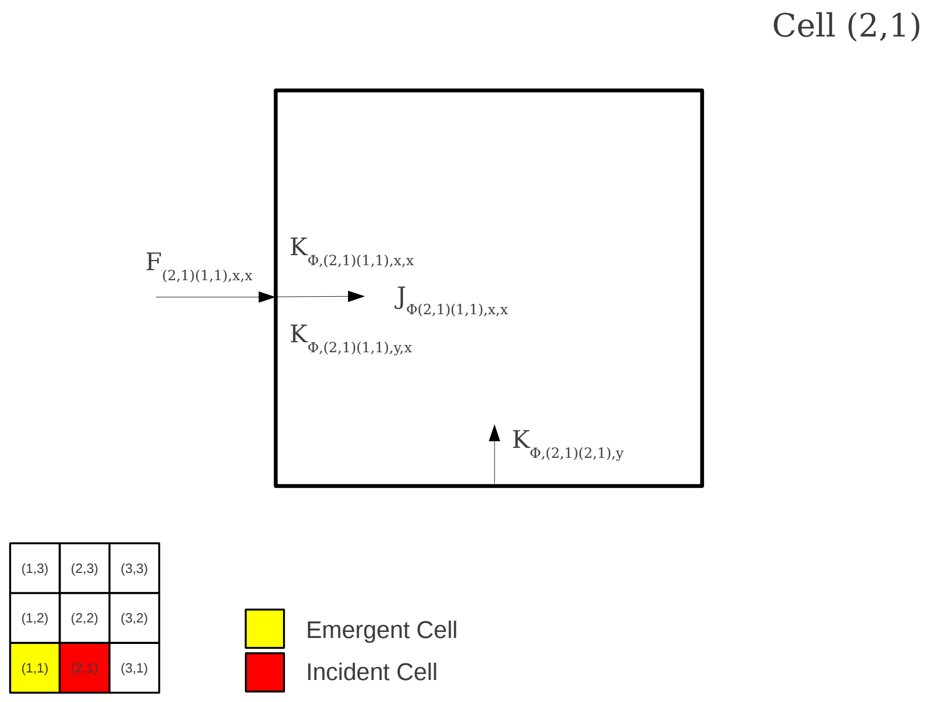


Figure 4.9: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (2,1) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

المنارة للاستشارات
www.manaraa.com

Table 4.5: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J_\phi}$, $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{K_\psi}$ Resulting from an Angular Flux Incident on a Face of Cell (2,1) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
$F_{(2,1)(1,1),x,x}$	1
Elements of $\boldsymbol{J_\phi}$ Calculated	
Element	Value
$J_{\phi,(2,1)(1,1),x,x}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(2,1)(1,1),x,x}k_{\phi,(2,1),x}$
Elements of $\boldsymbol{K_\phi}$ Calculated	
Element	Value
$K_{\phi,(2,1)(1,1),x,x}$	$k_{\psi,(1,1),x\to x}F_{(2,1)(1,1),x,x}k_{\phi,(2,1),x}$
$K_{\phi,(2,1)(1,1),y,x}$	$k_{\psi,(1,1),y\to x}F_{(2,1)(1,1),x,x}k_{\phi,(2,1),x}$
$K_{\phi,(2,1)(2,1),y}$	$k_{\phi,(2,1),y}$
Elements of $\boldsymbol{J_\psi}$ Calculated	
Element	Value
N/A	N/A
Elements of $\boldsymbol{K_\psi}$ Calculated	
Element	Value
N/A	N/A

www.manaraa.com

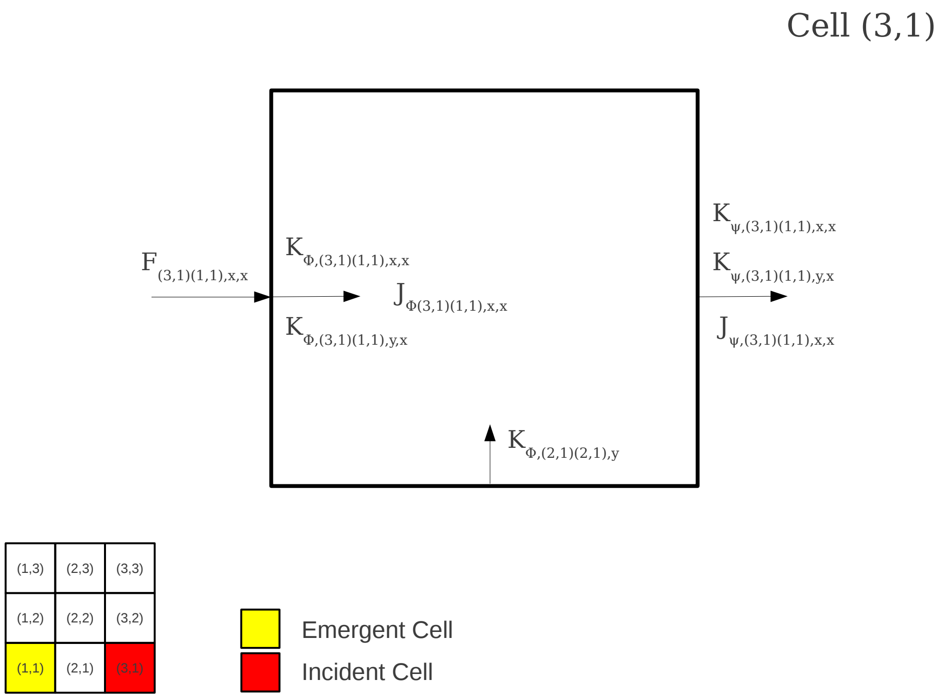


Figure 4.10: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (3,1) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

www.manaraa.com

Table 4.6: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J_\phi}$, $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{K_\psi}$ Resulting from an Angular Flux Incident on a Face of Cell (3,1) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
$F_{(3,1)(1,1),x,x}$	$k_{\psi,(2,1),x\to x}$

Elements of $\boldsymbol{J_\phi}$ Calculated	
Element	Value
$J_{\phi,(3,1)(1,1),x,x}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,1)(1,1),x,x}k_{\phi,(3,1),x}$

Elements of $\boldsymbol{K_\phi}$ Calculated	
Element	Value
$K_{\phi,(3,1)(1,1),x,x}$	$k_{\psi,(1,1),x\to x}F_{(3,1)(1,1),x,x}k_{\phi,(3,1),x}$
$K_{\phi,(3,1)(1,1),y,x}$	$k_{\psi,(1,1),y\to x}F_{(3,1)(1,1),x,x}k_{\phi,(3,1),x}$
$K_{\phi,(3,1)(3,1),y}$	$k_{\phi,(3,1),y}$

Elements of $\boldsymbol{J_\psi}$ Calculated	
Element	Value
$J_{\psi,(3,1)(1,1),x,x}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,1)(1,1),x,x}k_{\psi,(3,1),x\to x}$

Elements of $\boldsymbol{K_\psi}$ Calculated	
Element	Value
$K_{\psi,(3,1)(1,1),x,x}$	$k_{\psi,(1,1),x\to x}F_{(3,1)(1,1),x,x}k_{\psi,(3,1),x\to x}$
$K_{\psi,(3,1)(1,1),y,x}$	$k_{\psi,(1,1),y\to x}F_{(3,1)(1,1),x,x}k_{\psi,(3,1),x\to x}$

المنارة للاستشارات
www.manaraa.com

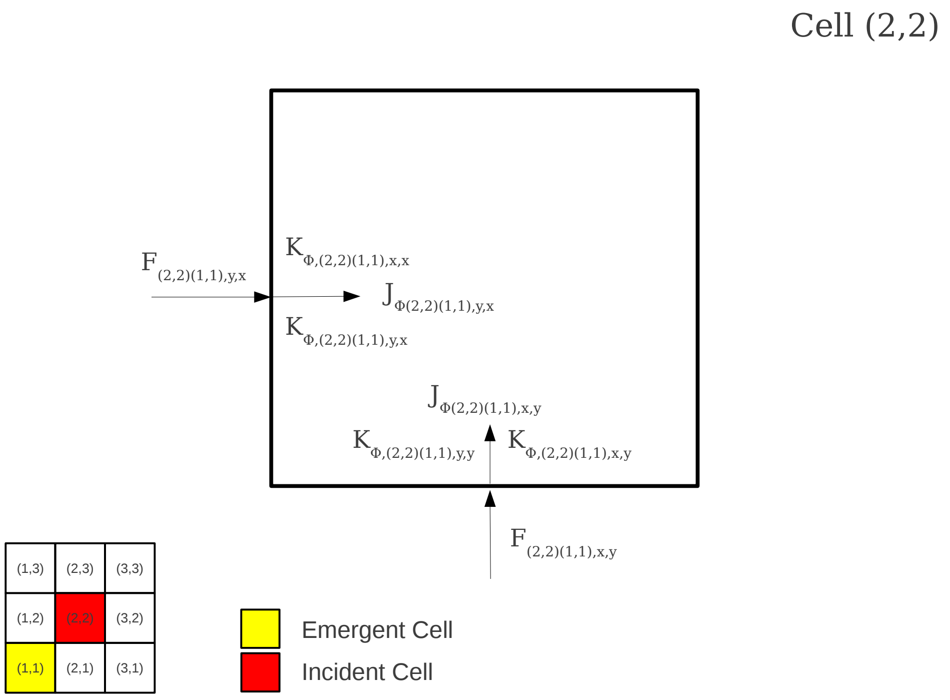


Figure 4.11: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (2,2) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

www.manaraa.com

Table 4.7: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J_\phi}$, $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{K_\psi}$ Resulting from an Angular Flux Incident on a Face of Cell (2,2) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
$F_{(2,2)(1,1),x,y}$	$k_{\psi,(2,1),x\to y}$
$F_{(2,2)(1,1),y,x}$	$k_{\psi,(1,2),y\to x}$
Elements of $\boldsymbol{J_\phi}$ Calculated	
Element	Value
$J_{\phi,(2,2)(1,1),x,y}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(2,2)(1,1),x,y}k_{\phi,(2,2),y}$
$J_{\phi,(2,2)(1,1),y,x}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(2,2)(1,1),y,x}k_{\phi,(2,2),x}$
Elements of $\boldsymbol{K_\phi}$ Calculated	
Element	Value
$K_{\phi,(2,2)(1,1),x,y}$	$k_{\psi,(1,1),x\to x}F_{(2,2)(1,1),x,y}k_{\phi,(2,2),y}$
$K_{\phi,(2,2)(1,1),y,y}$	$k_{\psi,(1,1),y\to x}F_{(2,2)(1,1),x,y}k_{\phi,(2,2),y}$
$K_{\phi,(2,2)(1,1),x,x}$	$k_{\psi,(1,1),x\to y}F_{(2,2)(1,1),y,x}k_{\phi,(2,2),x}$
$K_{\phi,(2,2)(1,1),y,x}$	$k_{\psi,(1,1),y\to y}F_{(2,2)(1,1),y,x}k_{\phi,(2,2),x}$
Elements of $\boldsymbol{J_\psi}$ Calculated	
Element	Value
N/A	N/A
Elements of $\boldsymbol{K_\psi}$ Calculated	
Element	Value
N/A	N/A

المنارة للاستشارات
www.manaraa.com

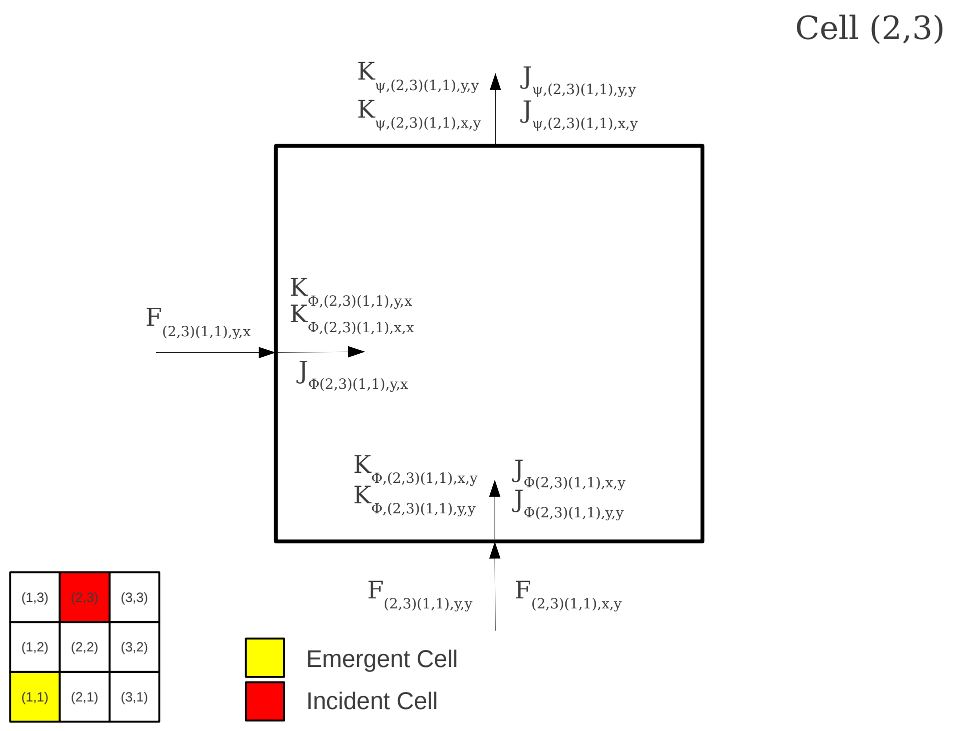


Figure 4.12: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (2,3) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

المنارة للاستشارات
www.manaraa.com

Table 4.8: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J_\phi}$, $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{K_\psi}$ Resulting from an Angular Flux Incident on a Face of Cell (2,3) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
$F_{(2,3)(1,1),x,y}$	$F_{(2,2)(1,1),x,y}k_{\psi,(2,2),y\to y}$
$F_{(2,3)(1,1),y,x}$	$F_{(1,3)(1,1),y,y}k_{\psi,(1,3),y\to x}$
$F_{(2,3)(1,1),y,y}$	$F_{(2,2)(1,1),y,x}k_{\psi,(2,2),x\to y}$
Elements of $\boldsymbol{J_\phi}$ Calculated	
Element	Value
$J_{\phi,(2,3)(1,1),x,y}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(2,3)(1,1),x,y}k_{\phi,(2,3),y}$
$J_{\phi,(2,3)(1,1),y,x}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(2,3)(1,1),y,x}k_{\phi,(2,3),x}$
$J_{\phi,(2,3)(1,1),y,y}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(2,3)(1,1),y,y}k_{\phi,(2,3),y}$
Elements of $\boldsymbol{K_\phi}$ Calculated	
Element	Value
$K_{\phi,(2,3)(1,1),x,y}$	$k_{\psi,(1,1),x\to x}F_{(2,3)(1,1),x,y}k_{\phi,(2,3),y}$
$K_{\phi,(2,3)(1,1),y,y}$	$k_{\psi,(1,1),y\to x}F_{(2,3)(1,1),x,y}k_{\phi,(2,3),y}$
$K_{\phi,(2,3)(1,1),x,x}$	$k_{\psi,(1,1),x\to y}F_{(2,3)(1,1),y,x}k_{\phi,(2,3),x}$
$K_{\phi,(2,3)(1,1),y,x}$	$k_{\psi,(1,1),y\to y}F_{(2,3)(1,1),y,x}k_{\phi,(2,3),x}$
Elements of $\boldsymbol{J_\psi}$ Calculated	
Element	Value
$J_{\psi,(2,3)(1,1),x,y}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(2,3)(1,1),x,y}k_{\psi,(2,3),y\to y}$
$J_{\psi,(2,3)(1,1),y,y}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(2,3)(1,1),y,y}k_{\psi,(2,3),y\to y} + j_{\psi,(1,1),y}c_{(1,1)}F_{(2,3)(1,1),y,x}k_{\psi,(2,3),x\to y}$
Elements of $\boldsymbol{K_\psi}$ Calculated	
Element	Value
$K_{\psi,(2,3)(1,1),x,y}$	$k_{\psi,(1,1),x\to x}F_{(2,3)(1,1),x,y}k_{\psi,(2,3),y\to y} + k_{\psi,(1,1),x\to y}F_{(2,3)(1,1),y,y}k_{\psi,(2,3),y\to y} + k_{\psi,(1,1),x\to y}F_{(2,3)(1,1),y,x}k_{\psi,(2,3),x\to y}$
$K_{\psi,(2,3)(1,1),y,y}$	$k_{\psi,(1,1),y\to y}F_{(2,3)(1,1),y,y}k_{\psi,(2,3),y\to y} + k_{\psi,(1,1),y\to x}F_{(2,3)(1,1),x,y}k_{\psi,(2,3),y\to y} + k_{\psi,(1,1),y\to y}F_{(2,3)(1,1),y,x}k_{\psi,(2,3),x\to y}$

المنارة للاستشارات
www.manaraa.com

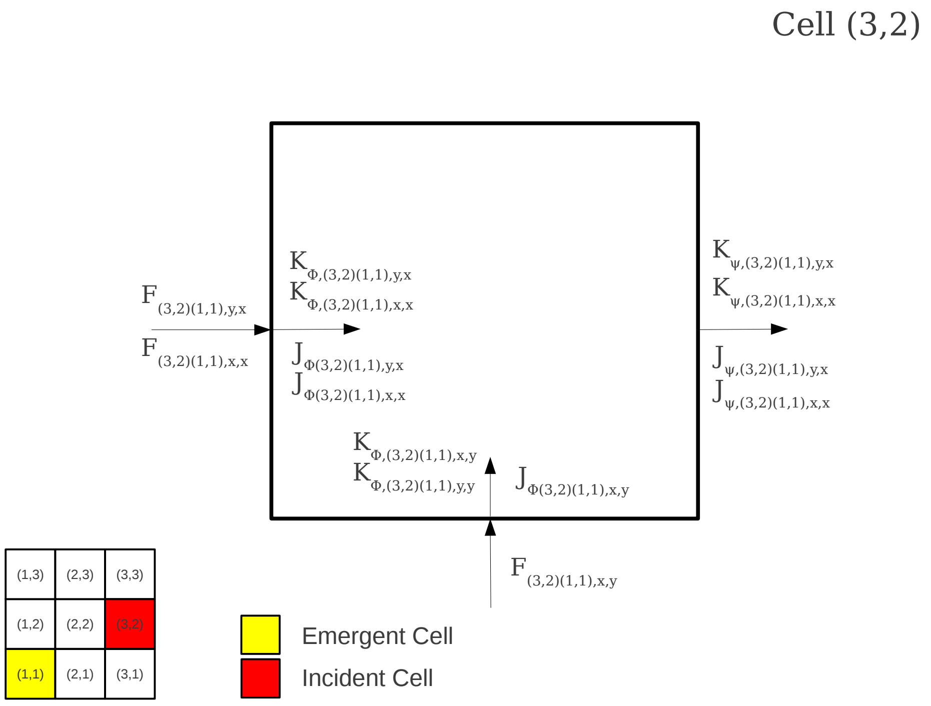


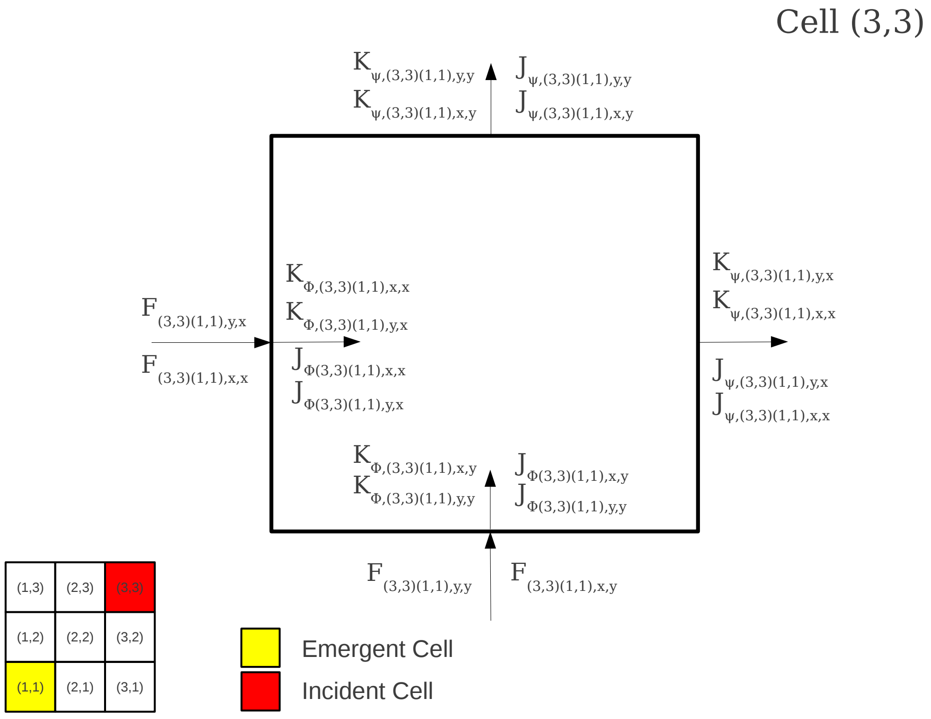
Figure 4.13: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (3,2) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

المنارة للاستشارات
www.manaraa.com

Table 4.9: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J}_{\boldsymbol{\phi}}$, $\boldsymbol{K}_{\boldsymbol{\phi}}$, $\boldsymbol{J}_{\boldsymbol{\psi}}$, and $\boldsymbol{K}_{\boldsymbol{\psi}}$ Resulting from an Angular Flux Incident on a Face of Cell (3,2) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
$F_{(3,2)(1,1),y,x}$	$F_{(2,2)(1,1),y,x}k_{\psi,(2,2),x\to x}$
$F_{(3,2)(1,1),x,y}$	$F_{(3,1)(1,1),x,x}k_{\psi,(3,1),x\to y}$
$F_{(3,2)(1,1),x,x}$	$F_{(2,2)(1,1),x,y}k_{\psi,(2,2),y\to x}$
Elements of $\boldsymbol{J}_{\boldsymbol{\phi}}$ Calculated	
Element	Value
$J_{\phi,(3,2)(1,1),y,x}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(3,2)(1,1),y,x}k_{\phi,(3,2),x}$
$J_{\phi,(3,2)(1,1),x,y}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,2)(1,1),x,y}k_{\phi,(3,2),y}$
$J_{\phi,(3,2)(1,1),x,x}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,2)(1,1),x,x}k_{\phi,(3,2),y}$
Elements of $\boldsymbol{K}_{\boldsymbol{\phi}}$ Calculated	
Element	Value
$K_{\phi,(3,2)(1,1),y,x}$	$k_{\psi,(1,1),y\to y}F_{(3,2)(1,1),y,x}k_{\phi,(3,2),x}$
$K_{\phi,(3,2)(1,1),x,x}$	$k_{\psi,(1,1),x\to y}F_{(3,2)(1,1),y,x}k_{\phi,(3,2),x}$
$K_{\phi,(3,2)(1,1),y,y}$	$k_{\psi,(1,1),y\to x}F_{(3,2)(1,1),x,y}k_{\phi,(3,2),y}$
$K_{\phi,(3,2)(1,1),x,y}$	$k_{\psi,(1,1),x\to x}F_{(3,2)(1,1),x,y}k_{\phi,(3,2),y}$
Elements of $\boldsymbol{J}_{\boldsymbol{\psi}}$ Calculated	
Element	Value
$J_{\psi,(3,2)(1,1),y,x}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(3,2)(1,1),y,x}k_{\psi,(3,2),x\to x}$
$J_{\psi,(3,2)(1,1),x,x}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,2)(1,1),x,x}k_{\psi,(3,2),x\to x} + j_{\psi,(1,1),x}c_{(1,1)}F_{(3,2)(1,1),x,y}k_{\psi,(3,2),y\to x}$
Elements of $\boldsymbol{K}_{\boldsymbol{\psi}}$ Calculated	
Element	Value
$K_{\psi,(3,2)(1,1),y,x}$	$k_{\psi,(1,1),y\to y}F_{(3,2)(1,1),y,x}k_{\psi,(3,2),x\to x} + k_{\psi,(1,1),y\to x}F_{(3,2)(1,1),x,x}k_{\psi,(3,2),x\to x} + k_{\psi,(1,1),y\to x}F_{(3,2)(1,1),x,y}k_{\psi,(3,2),y\to x}$
$K_{\psi,(3,2)(1,1),x,x}$	$k_{\psi,(1,1),x\to x}F_{(3,2)(1,1),x,x}k_{\psi,(3,2),x\to x} + k_{\psi,(1,1),x\to y}F_{(3,2)(1,1),y,x}k_{\psi,(3,2),x\to x} + k_{\psi,(1,1),x\to x}F_{(3,2)(1,1),x,y}k_{\psi,(3,2),y\to x}$

www.manaraa.com



Cell (3,3)
$K_{\psi,(3,3)(1,1),y,y}$
$K_{\psi,(3,3)(1,1),x,y}$
$J_{\psi,(3,3)(1,1),y,y}$
$J_{\psi,(3,3)(1,1),x,y}$
$F_{(3,3)(1,1),y,x}$
$F_{(3,3)(1,1),x,x}$
$K_{\Phi,(3,3)(1,1),x,x}$
$K_{\Phi,(3,3)(1,1),y,x}$
$J_{\Phi(3,3)(1,1),x,x}$
$J_{\Phi(3,3)(1,1),y,x}$
$K_{\psi,(3,3)(1,1),y,x}$
$K_{\psi,(3,3)(1,1),x,x}$
$J_{\psi,(3,3)(1,1),y,x}$
$J_{\psi,(3,3)(1,1),x,x}$
$K_{\Phi,(3,3)(1,1),x,y}$
$K_{\Phi,(3,3)(1,1),y,y}$
$J_{\Phi(3,3)(1,1),x,y}$
$J_{\Phi(3,3)(1,1),y,y}$
$F_{(3,3)(1,1),y,y}$
$F_{(3,3)(1,1),x,y}$
(1,3)
(2,3)
(3,3)
(1,2)
(2,2)
(3,2)
(1,1)
(2,1)
(3,1)
🟨 Emergent Cell
🟥 Incident Cell

Figure 4.14: Construction Basis of All Four ITMM Operators Resulting from an Angular Flux Incident on a Face of Cell (3,3) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

www.manaraa.com

Table 4.10: Values of Nonzero Elements of $\boldsymbol{F}$, $\boldsymbol{J_\phi}$, $\boldsymbol{K_\phi}$, $\boldsymbol{J_\psi}$, and $\boldsymbol{K_\psi}$ Resulting from an Angular Flux Incident on a Face of Cell (3,3) and Emergent from a Face of Cell (1,1) for an Ordinate in the Primary Octant

Elements of $\boldsymbol{F}$ Calculated	
Element	Value
$F_{(3,3)(1,1),y,x}$	$F_{(2,3)(1,1),y,x}k_{\psi,(2,3),x\to x} + F_{(2,3)(1,1),x,y}k_{\psi,(2,3),y\to x}$
$F_{(3,3)(1,1),x,y}$	$F_{(3,2)(1,1),x,x}k_{\psi,(3,2),x\to y} + F_{(3,2)(1,1),x,y}k_{\psi,(3,2),y\to y}$
$F_{(3,3)(1,1),y,y}$	$F_{(3,2)(1,1),y,x}k_{\psi,(3,2),x\to y}$
$F_{(3,3)(1,1),x,x}$	$F_{(2,3)(1,1),x,y}k_{\psi,(2,3),y\to x}$

Elements of $\boldsymbol{J_\phi}$ Calculated	
Element	Value
$J_{\phi,(3,3)(1,1),y,x}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(3,3)(1,1),y,x}k_{\phi,(3,3),x}$
$J_{\phi,(3,3)(1,1),x,y}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,3)(1,1),x,y}k_{\phi,(3,3),y}$
$J_{\phi,(3,3)(1,1),y,y}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(3,3)(1,1),y,y}k_{\phi,(3,3),y}$
$J_{\phi,(3,3)(1,1),x,x}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,3)(1,1),x,x}k_{\phi,(3,3),x}$

Elements of $\boldsymbol{K_\phi}$ Calculated	
Element	Value
$K_{\phi,(3,3)(1,1),y,x}$	$k_{\psi,(1,1),y\to y}F_{(3,3)(1,1),y,x}k_{\phi,(3,3),x} + k_{\psi,(1,1),y\to x}F_{(3,3)(1,1),x,x}k_{\phi,(3,3),x}$
$K_{\phi,(3,3)(1,1),y,y}$	$k_{\psi,(1,1),y\to y}F_{(3,3)(1,1),y,y}k_{\phi,(3,3),y} + k_{\psi,(1,1),y\to x}F_{(3,3)(1,1),x,y}k_{\phi,(3,3),y}$
$K_{\phi,(3,3)(1,1),x,x}$	$k_{\psi,(1,1),x\to y}F_{(3,3)(1,1),y,x}k_{\phi,(3,3),x} + k_{\psi,(1,1),x\to x}F_{(3,3)(1,1),x,x}k_{\phi,(3,3),x}$
$K_{\phi,(3,3)(1,1),x,y}$	$k_{\psi,(1,1),x\to y}F_{(3,3)(1,1),y,y}k_{\phi,(3,3),y} + k_{\psi,(1,1),x\to x}F_{(3,3)(1,1),x,y}k_{\phi,(3,3),y}$

Elements of $\boldsymbol{J_\psi}$ Calculated	
Element	Value
$J_{\psi,(3,3)(1,1),y,x}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(3,3)(1,1),y,x}k_{\psi,(3,3),x\to x} + j_{\psi,(1,1),y}c_{(1,1)}F_{(3,3)(1,1),y,y}k_{\psi,(3,3),y\to x}$
$J_{\psi,(3,3)(1,1),y,y}$	$j_{\psi,(1,1),y}c_{(1,1)}F_{(3,3)(1,1),y,y}k_{\psi,(3,3),y\to y} + j_{\psi,(1,1),y}c_{(1,1)}F_{(3,3)(1,1),y,x}k_{\psi,(3,3),x\to y}$
$J_{\psi,(3,3)(1,1),x,x}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,3)(1,1),x,x}k_{\psi,(3,3),x\to x} + j_{\psi,(1,1),x}c_{(1,1)}F_{(3,3)(1,1),x,y}k_{\psi,(3,3),y\to x}$
$J_{\psi,(3,3)(1,1),x,y}$	$j_{\psi,(1,1),x}c_{(1,1)}F_{(3,3)(1,1),x,y}k_{\psi,(3,3),y\to y} + j_{\psi,(1,1),x}c_{(1,1)}F_{(3,3)(1,1),x,x}k_{\psi,(3,3),x\to y}$

Elements of $\boldsymbol{K_\psi}$ Calculated	
Element	Value
$K_{\psi,(3,3)(1,1),y,x}$	$k_{\psi,(1,1),y\to y}F_{(3,3)(1,1),y,x}k_{\psi,(3,3),x\to x} + k_{\psi,(1,1),y\to y}F_{(3,3)(1,1),y,y}k_{\psi,(3,3),y\to x}+$ $k_{\psi,(1,1),y\to x}F_{(3,3)(1,1),x,x}k_{\psi,(3,3),x\to x} + k_{\psi,(1,1),y\to x}F_{(3,3)(1,1),x,y}k_{\psi,(3,3),y\to x}$
$K_{\psi,(3,3)(1,1),y,y}$	$k_{\psi,(1,1),y\to y}F_{(3,3)(1,1),y,x}k_{\psi,(3,3),x\to y} + k_{\psi,(1,1),y\to y}F_{(3,3)(1,1),y,y}k_{\psi,(3,3),y\to y}+$ $k_{\psi,(1,1),y\to x}F_{(3,3)(1,1),x,x}k_{\psi,(3,3),x\to y} + k_{\psi,(1,1),y\to x}F_{(3,3)(1,1),x,y}k_{\psi,(3,3),y\to y}$
$K_{\psi,(3,3)(1,1),x,x}$	$k_{\psi,(1,1),x\to y}F_{(3,3)(1,1),y,x}k_{\psi,(3,3),x\to x} + k_{\psi,(1,1),x\to y}F_{(3,3)(1,1),y,y}k_{\psi,(3,3),y\to x}+$ $k_{\psi,(1,1),x\to x}F_{(3,3)(1,1),x,x}k_{\psi,(3,3),x\to x} + k_{\psi,(1,1),x\to x}F_{(3,3)(1,1),x,y}k_{\psi,(3,3),y\to x}$
$K_{\psi,(3,3)(1,1),x,y}$	$k_{\psi,(1,1),x\to y}F_{(3,3)(1,1),y,x}k_{\psi,(3,3),x\to y} + k_{\psi,(1,1),x\to y}F_{(3,3)(1,1),y,y}k_{\psi,(3,3),y\to y}+$ $k_{\psi,(1,1),x\to x}F_{(3,3)(1,1),x,x}k_{\psi,(3,3),x\to y} + k_{\psi,(1,1),x\to x}F_{(3,3)(1,1),x,y}k_{\psi,(3,3),y\to y}$

المنارة للاستشارات
www.manaraa.com

The previous figures and tables have demonstrated the two-dimensional form of the FMM algorithm in a step-by-step process. In these tables and figures, it can be seen how elements of $\boldsymbol{F}$ are used in the calculation of other elements of $\boldsymbol{F}$ in addition to the calculation of each element of the ITMM operators. In this manner, $\boldsymbol{F}$ allows for the avoidance of repeat calculations in both the ITMM operators and its own elements.

المنارة للاستشارات
www.manaraa.com

Chapter 5

Performance Model and Timing Results

5.1 Performance Model

It is useful to be able to predict run times as a function of problem parameters, e.g. number of cells in the domain, for scaling purposes. As such, a performance model has been developed to serve this purpose. The performance of the construction algorithm in terms of computational time is dependent upon three variables: The number of cells in the sub-domain, $N = I$ x J x K, the order of anisotropy, L, and the number of angles, D. The performance model was determined based on analysis of the operator construction subroutines, particularly the number of calculations required to build $\boldsymbol{F}$ and each ITMM matrix operator.

The terms of the performance model identified in the analysis of the algorithm are asymptotic in nature and based on the assumption of a cubic mesh (i.e. $I = J = K$). For example, for the total number of cells, N, the number of cells in a single row or column is $N^{1/3}$ and the number of cells in a single plane is $N^{2/3}$.

First, consider the construction of the fundamental matrix $\boldsymbol{F}$. As previously demonstrated, all the ITMM operators are constructed through multiplications to elements of $\boldsymbol{F}$. The performance model of the construction of $\boldsymbol{F}$ involves three terms: The construction of elements of $\boldsymbol{F}$ along a row or a column intersecting the emergent cell ($3a_1DN^{4/3}$), along planes intersecting the emergent cell ($3a_2D(N^{5/3} - N^{4/3})$), and through the entire sub-domain from the emergent cell ($a_3D(N^2 - 3N^{5/3} - 3N^{4/3})$), where a_i ($i = 1, 2, 3, ...$) are measures of the time consumed in completing a single instance of the corresponding instructions listed above.

As an example of the creation of the performance model terms, the creation of each of the terms for the performance model of the construction of $\boldsymbol{F}$ will now be described in detail. The first

www.manaraa.com

term is calculated as the number of possible emergent cells, N, multiplied by the number of possible incident cells in all rows or columns intersecting the emergent cell, $3N^{1/3}$ multiplied by the number of angles, D. The second term is similarly calculated as the number of possible emergent cells, N, multiplied by the number of possible incident cells in all planes intersecting the emergent cell, $3N^{2/3}$, multiplied by the number of angles D. The second term also includes the subtraction of the first term, since those elements of $\boldsymbol{F}$ have already been calculated. The third term is calculated as the number of possible emergent cells, N, multiplied by the number of possible incident cells in the entire sub-domain, N, multiplied by the number of angles, D. The third term also subtracts $3DN^{5/3}$ and $3DN^{4/3}$ because those elements of $\boldsymbol{F}$ have already been calculated. These terms are added together to create the performance model for constructing the fundamental matrix,

$$t_F = 3a_1DN^{4/3} + 3a_2D(N^{5/3} - N^{4/3}) + a_3D(N^2 - 3N^{5/3} - 3N^{4/3}) \ . \tag{5.1}$$

The number of calculations for each of the ITMM matrix operators is a function of the number of cells involved in the operator construction, the number of angles, and the number of angular moments. The number of angular moments, represented here as H, is calculated by $H = (L+1)^2$. While the number of calculations for each operator requires a multiplication by D to account for the number of angles, the use of H differs between the operators. $\boldsymbol{K_\psi}$ is the only operator completely independent of the flux angular moments and, using Table 3.1 as a reference for the number of cells involved in the calculation, the performance model for $\boldsymbol{K_\psi}$ is

$$t_{\boldsymbol{K_\psi}} = a_4DN^{4/3} \ . \tag{5.2}$$

$\boldsymbol{J_\psi}$ and $\boldsymbol{K_\phi}$ both involve the flux angular moments on one end of the calculation. As such, the number of calculations required for these operators is multiplied by H, resulting in

$$t_{\boldsymbol{J_\psi}+\boldsymbol{K_\phi}} = a_5DN^{5/3}H \ . \tag{5.3}$$

$\boldsymbol{J_\phi}$ requires consideration of the flux angular moments in both the emergent cell and the incident cell. Therefore, the number of calculations is repeated H^2 times. In our implementation of the algorithm, several calculations are conducted in a single loop over all angular moments to improve computational efficiency. Hence the need for another term multiplied by only H but the same number of cells, N^2. These terms comprise the performance model for the calculation of $\boldsymbol{J_\phi}$,

$$t_{\boldsymbol{J_\phi}} = a_6DN^2H + a_7DN^2H^2 \ . \tag{5.4}$$

www.manaraa.com

The total construction time for the operators is then

$$t_{total} = t_F + t_{\boldsymbol{K}_\psi} + t_{\boldsymbol{J}_\psi + \boldsymbol{K}_\phi} + t_{\boldsymbol{J}_\phi} \ . \tag{5.5}$$

Substituting the previously defined terms results in

$$\begin{aligned} t_{total} = 3a_1DN^{4/3} + 3a_2D(N^{5/3} - N^{4/3}) + a_3D(N^2 - 3N^{5/3} - 3N^{4/3}) \\ + a_4DN^{4/3} + a_5DN^{5/3}H + a_6DN^2H + a_7DN^2H^2. \end{aligned} \tag{5.6}$$

In order to determine the values of the constants in equation 5.6, timed runs of the code with varying number of cells, number of angles, and order of anisotropy were conducted.

5.2 Timing Results

The following tables (5.1 for L=0, 5.2 for L=1, 5.3 for L=3) show the measured execution times of these trials. Each time shown is the average measured execution time across ten runs of the code. "FMM" designates the Fundamental Matrix Method of constructing the matrix operators, outlined in this work, and "DMS" designates the Differential Mesh Sweep method developed by Zerr [4]. As can be seen in each of these tables, the Fundamental Matrix Method significantly outperforms the Differential Mesh Sweep in all cases. For all timing results shown here, a 4-core, 3.7 GHz processor was used.

Table 5.1: Matrix Operator Construction Time (s) for L=0 (isotropic)

Method	N	S_4	S_8	S_{12}	S_{16}
FMM	64	.0088	.0148	.0268	.0384
DMS	64	.0156	.0356	.0624	.1016
FMM	216	.0416	.0876	.1688	.2768
DMS	216	.0596	.1844	.3820	.6572
FMM	512	.1952	.4764	.9408	1.780
DMS	512	.2468	.8149	1.711	2.925
FMM	1000	.6588	1.507	3.961	6.722
DMS	1000	.8213	2.735	5.651	9.849

www.manaraa.com

Table 5.2: Matrix Operator Construction Time (s) for L=1

Method	N	S_4	S_8	S_{12}	S_{16}
FMM	64	.0140	.0260	.0528	.0868
DMS	64	.0212	.0576	.1116	.1920
FMM	216	.0736	.1892	.4040	.7388
DMS	216	.1296	.4076	.8493	1.467
FMM	512	.4048	1.192	2.871	5.118
DMS	512	.6880	2.273	4.836	8.263
FMM	1000	1.245	4.085	9.172	16.22
DMS	1000	2.242	7.455	15.60	27.04

Table 5.3: Matrix Operator Construction Time (s) for L=3

Method	N	S_4	S_8	S_{12}	S_{16}
FMM	64	.0396	.1148	.2528	.4668
DMS	64	.1036	.3704	.6608	1.110
FMM	216	.3136	1.045	2.193	4.332
DMS	216	.5936	1.965	4.078	6.970
FMM	512	1.944	6.961	13.62	28.63
DMS	512	9.815	30.65	66.90	113.7
FMM	1000	7.394	25.01	45.05	79.77
DMS	1000	12.16	40.46	85.16	147.8

In order to determine the values of the model constants of equation 5.6, a Mathematica fitting function was applied to the timing data for the Fundamental Matrix Method in Tables 5.1 - 5.3. This function uses a least squares algorithm to determine the minimal residual among all the data points by converging on an optimal value of the constants. This procedure results in the values shown in Table 5.4 for the constants, applied to equation 5.6. These constant values are specific to the timing results obtained by the aforementioned processor, and will be different on another system with different capabilities, although the model, i.e. the functional dependence on D, N, and H, will remain the same as it characterizes the algorithm itself [19].

www.manaraa.com

Table 5.4: Least Squares Fit Constant Values for Equation 5.6

Constant	Value (sec)
a_1	3.259 x 10^{-10}
a_2	6.361 x 10^{-9}
a_3	2.693 x 10^{-8}
a_4	7.411 x 10^{-10}
a_5	2.751 x 10^{-9}
a_6	5.024 x 10^{-9}
a_7	6.813 x 10^{-10}

This is now the complete performance model of the FMM algorithm. Figures 5.1 through 5.3 show the performance model and the measured execution times for $L = 0$, 1, and 3, respectively, plotted against the number of cells in the domain, N.

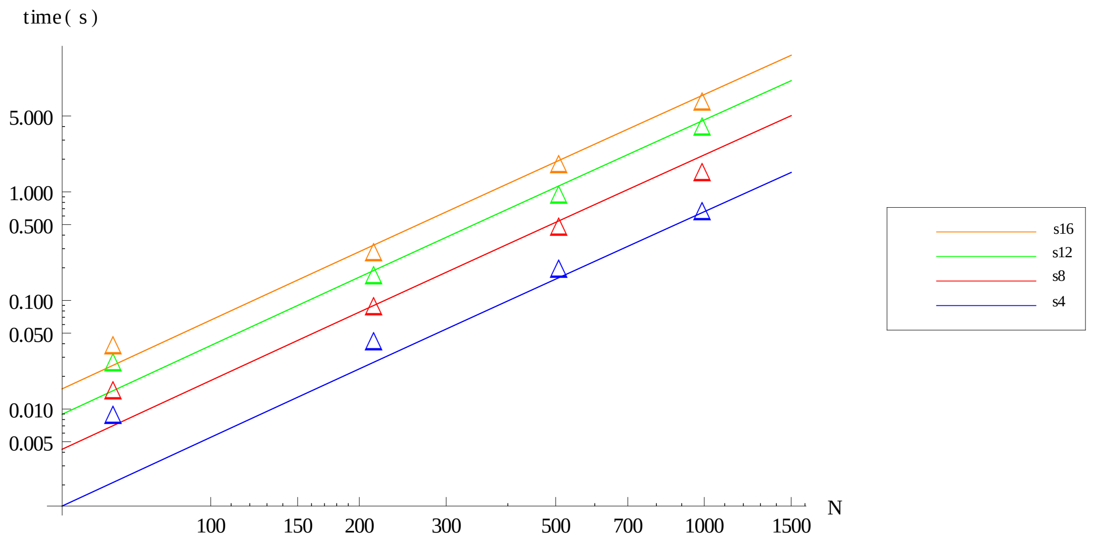


Figure 5.1: Performance Model (lines) and Measured Matrix Operator Construction Times (triangles) for L=0 (isotropic scattering)

www.manaraa.com

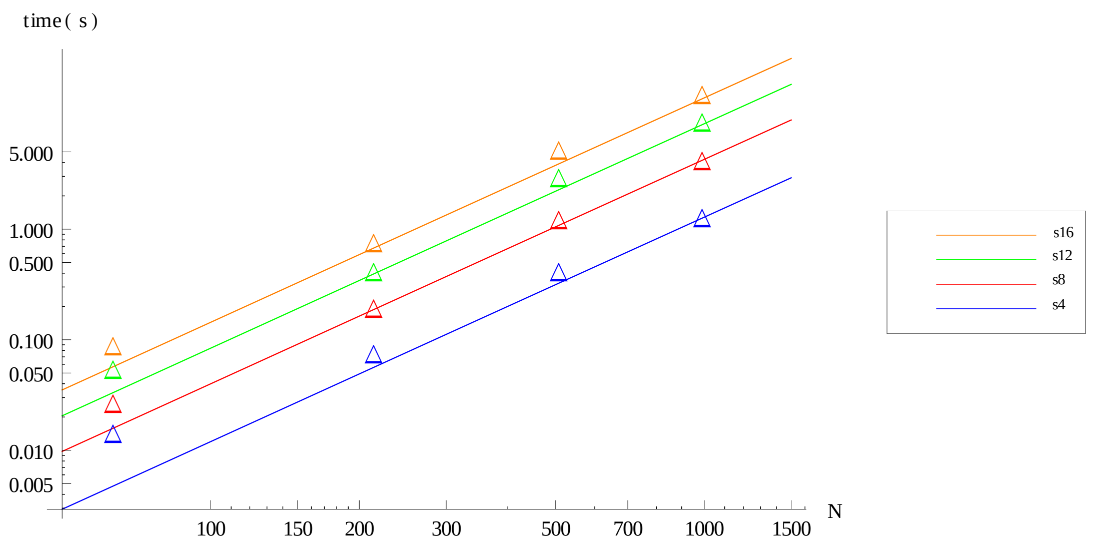


Figure 5.2: Performance Model (lines) and Measured Matrix Operator Construction Times (triangles) for L=1

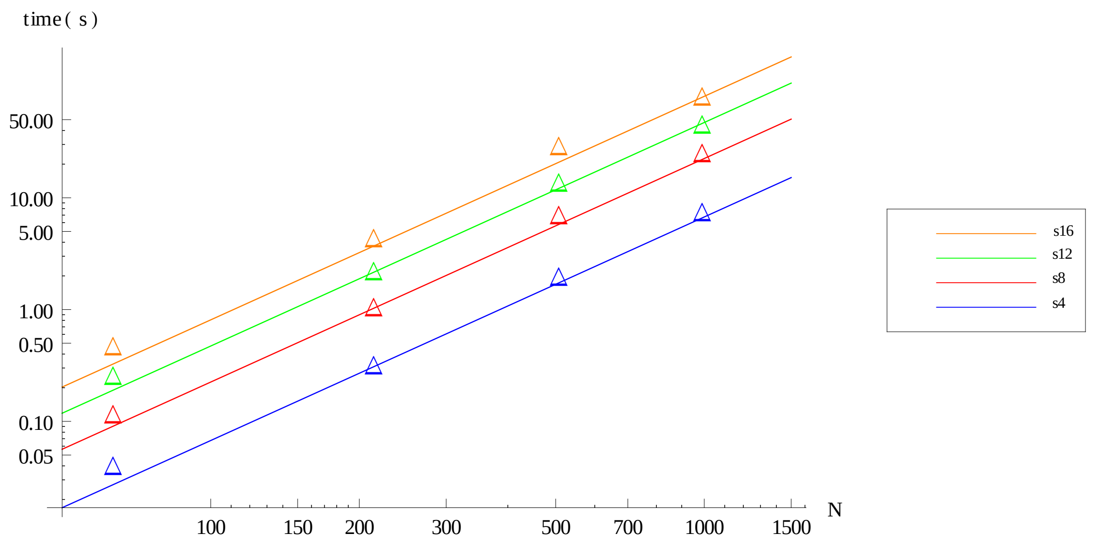


Figure 5.3: Performance Model (lines) and Measured Matrix Operator Construction Times (triangles) for L=3

Discrepancies in the fit of the performance model to the data points can be attributed to several causes. First among these is the fact that we used the entire data set for the determina-

www.manaraa.com

tion of the constants. Any data point that could be considered somewhat aberrant can damage the overall fit and compound errors in conforming to individual data sets. Second would be the effect on execution time from the frequency of memory cache hits and misses. Each miss can increase the overall time while each hit can reduce it. Given the sheer number of memory accesses required for the FMM algorithm, any such problem could cause significant variation in the data points. The last source of fitting error would be background processes executing on the system during the timing measurement runs. While each data point is an average of ten code executions, the averaging does not account for background processes which may slow the performance of the algorithm. Considering these sources of error, the performance model trends, if not exactly predicts, the execution time of the FMM algorithm. The major error visible in the plots above is that the model consistently under-predicts the execution time for small N but becomes highly accurate as N increases. This result is both understandable and expected due to the asymptotic nature of the performance model, meaning the power dependencies on N are only valid in the sense $N \rightarrow \infty$. In scenarios with a small value of N, the asymptotic nature can cause a large discrepancy between the model and the measured execution time. This effect decreases and the model accurately predicts execution times in the asymptotic regime of large N.

5.3 Memory Requirements

The ITMM is a memory-limited method as the size of the necessary matrix operators grows significantly with increased size, number of angles, and anisotropic scattering order. Tables 5.5 through 5.8 demonstrate the reasoning for limiting the timing results in the previous section to the selected values of D, N, and H.

www.manaraa.com

Table 5.5: Matrix Operator Memory Requirements for L=0

Operator	N	S_4	S_8	S_{12}	S_{16}
$\boldsymbol{J}_\phi$	64	33 KB	33 KB	33 KB	33 KB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	64	37 KB	120 KB	260 KB	440 KB
$\boldsymbol{K}_\psi$	64	14 KB	46 KB	97 KB	170 KB
$\boldsymbol{J}_\phi$	216	370 KB	370 KB	370 KB	370 KB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	216	120 KB	410 KB	870 KB	1.4 MB
$\boldsymbol{K}_\psi$	216	47 KB	160 KB	330 KB	560 KB
$\boldsymbol{J}_\phi$	512	2.1 MB	2.1 MB	2.1 MB	2.1 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	512	290 KB	980 KB	2.1 MB	3.6 MB
$\boldsymbol{K}_\psi$	512	110 KB	370 KB	770 KB	1.3 MB
$\boldsymbol{J}_\phi$	1000	8.0 MB	8.0 MB	8.0 MB	8.0 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	1000	580 KB	1.9 MB	4.0 MB	6.9 MB
$\boldsymbol{K}_\psi$	1000	220 KB	720 KB	1.5 MB	2.6 MB

Table 5.6: Matrix Operator Memory Requirements for L=1

Operator	N	S_4	S_8	S_{12}	S_{16}
$\boldsymbol{J}_\phi$	64	520 KB	520 KB	520 KB	520 KB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	64	147 KB	491 KB	1.0 MB	1.8 MB
$\boldsymbol{K}_\psi$	64	14 KB	46 KB	97 KB	170 KB
$\boldsymbol{J}_\phi$	216	6.0 MB	6.0 MB	6.0 MB	6.0 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	216	498 KB	1.7 MB	3.5 MB	6.0 MB
$\boldsymbol{K}_\psi$	216	47 KB	160 KB	330 KB	560 KB
$\boldsymbol{J}_\phi$	512	34 MB	34 MB	34 MB	34 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	512	1.2 MB	3.9 MB	8.3 MB	14 MB
$\boldsymbol{K}_\psi$	512	110 KB	370 KB	770 KB	1.3 MB
$\boldsymbol{J}_\phi$	1000	130 MB	130 MB	130 MB	130 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	1000	2.3 MB	7.7 MB	16 MB	28 MB
$\boldsymbol{K}_\psi$	1000	220 KB	720 KB	1.5 MB	2.6 MB

المنارة للاستشارات
www.manaraa.com

Table 5.7: Matrix Operator Memory Requirements for L=3

Operator	N	S_4	S_8	S_{12}	S_{16}
$\boldsymbol{J}_\phi$	64	8.4 MB	8.4 MB	8.4 MB	8.4 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	64	590 KB	2.0 MB	4.1 MB	7.1 MB
$\boldsymbol{K}_\psi$	64	14 KB	46 KB	97 KB	170 KB
$\boldsymbol{J}_\phi$	216	96 MB	96 MB	96 MB	96 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	216	2.0 MB	6.6 MB	14 MB	24 MB
$\boldsymbol{K}_\psi$	216	47 KB	160 KB	330 KB	560 KB
$\boldsymbol{J}_\phi$	512	540 MB	540 MB	540 MB	540 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	512	4.7 MB	16 MB	33 MB	57 MB
$\boldsymbol{K}_\psi$	512	110 KB	370 KB	770 KB	1.3 MB
$\boldsymbol{J}_\phi$	1000	2.0 GB	2.0 GB	2.0 GB	2.0 GB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	1000	9.2 MB	31 MB	65 MB	110 MB
$\boldsymbol{K}_\psi$	1000	220 KB	720 KB	1.5 MB	2.6 MB

Table 5.8: Matrix Operator Memory Requirements for L=5

Operator	N	S_4	S_8	S_{12}	S_{16}
$\boldsymbol{J}_\phi$	64	42 MB	42 MB	42 MB	42 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	64	1.3 MB	4.4 MB	9.3 MB	16 MB
$\boldsymbol{K}_\psi$	64	14 KB	46 KB	97 KB	170 KB
$\boldsymbol{J}_\phi$	216	480 MB	480 MB	480 MB	480 MB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	216	4.5 MB	15 MB	31 MB	54 MB
$\boldsymbol{K}_\psi$	216	47 KB	160 KB	330 KB	560 KB
$\boldsymbol{J}_\phi$	512	2.7 GB	2.7 GB	2.7 GB	2.7 GB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	512	11 MB	35 MB	74 MB	130 MB
$\boldsymbol{K}_\psi$	512	110 KB	370 KB	770 KB	1.3 MB
$\boldsymbol{J}_\phi$	1000	10 GB	10 GB	10 GB	10 GB
$\boldsymbol{J}_\psi/\boldsymbol{K}_\phi$	1000	21 MB	69 MB	150 MB	250 MB
$\boldsymbol{K}_\psi$	1000	220 KB	720 KB	1.5 MB	2.6 MB

www.manaraa.com

Several important properties of the ITMM matrix operators can be seen in tables 5.5 - 5.8. The most beneficial of these properties to the ITMM memory requirements is that $\boldsymbol{K_\psi}$ is not dependent on the order of anisotropic scattering, only the number of cells and number of angles, and therefore does not have increased memory requirements as the anisotropic scattering order is increased. $\boldsymbol{J_\phi}$, however, is the limiting ITMM matrix operator in terms of memory requirements. $\boldsymbol{J_\phi}$ is dependent upon the number of cells and the anisotropic scattering order in the manner $(N(L+1)^2)^2$. As such, despite the fact that $\boldsymbol{J_\phi}$ is not dependent on the number of angles, as the number of cells and anisotropic scattering order increases, $\boldsymbol{J_\phi}$ quickly dominates the memory requirements of the ITMM and limits the possible size of ITMM sub-domains.

As a result of the properties of $\boldsymbol{J_\phi}$, The timing results shown in this work are limited by the memory requirements of the ITMM. The size of the matrix operators necessarily grows with increased size, number of angles, and anisotropic order. This constraint is significant as memory requirements of the matrix operators quickly exceed system memory limits when any of these variables are augmented beyond the examples shown in this work (e.g., for a 12x12x12 sub-domain, S_{16} quadrature, and $L = 5$, memory requirements exceed 30 GB). Also, previous parallel Gauss-Seidel applications of the ITMM using a Red/Black coloring scheme have found that 4x4x4 sub-domains consume the shortest execution times for numbers of processors exceeding a few hundred [4].

www.manaraa.com

Chapter 6

Conclusions

The ITMM has been shown to be a competitive solution algorithm to the neutron transport equation using spatial domain decomposition on massively parallel computational platforms [4]. In the limit of very large number of processors, the speed of the algorithm, and its suitability for unstructured meshes, i.e. other than an ordered Cartesian grid, is limited by the construction of four matrix operators required for obtaining the solution in each sub-domain. The existing algorithm for construction of the four ITMM matrix operators, the DMS, is computationally expensive and was developed for a structured grid. As such, the motivation for development of a construction algorithm for the ITMM matrix operators that is both less computationally expensive and more geometrically robust is evident.

In this work, a new algorithm that provides for faster construction of the ITMM matrix operators has been described. This algorithm, the FMM, is based on the construction of a single, fundamental matrix, $\boldsymbol{F}$, representing the transport of a particle along every possible path throughout the sub-domain mesh. Each of the operators is constructed by multiplying an element of this fundamental matrix by two values dependent only upon the operator being constructed.

The elements of $\boldsymbol{F}$ are face-based, meaning that they represent the transport of a particle along the aforementioned path from the emergent face of one cell to the incident face of another cell. By being a face-based quantity, the elements of $\boldsymbol{F}$ can then be used to relate one face to another face, a face to a cell, or a cell to a cell, dependent on the single cell coupling factors by which they are multiplied. An example of the FMM algorithm is provided in this work for a two-dimensional, 3 x 3 sub-domain. The actual implementation and timing reported results, however, are for three-dimensional geometry and more numerous cells.

It can be seen from the results presented here that the FMM significantly outperforms the DMS in terms of matrix operator construction time for the ITMM. It should be noted that the time required for the solution algorithm, which was not detailed in this work, is significantly greater

www.manaraa.com

than the operator construction time. The FMM gains significance, however, when considering a calculation requiring multiple energy groups, time steps, or depletion steps, where repeated evaluations of the ITMM operators must be performed. In this situation, the time savings gained when using the FMM versus the DMS is multiplied by the number of time steps and/or the number of groups. As such, the FMM algorithm becomes more critical to the computational cost of the ITMM as the complexity of the target problem grows [19].

Although the FMM as shown here was developed for a structured Cartesian grid, an advantage of the FMM is its applicability to an unstructured mesh. The geometric simplicity of the FMM algorithm lies in that it does not require an orderly mesh sweep, but rather can be applied to any grouping of cells as long as their spatial relation to each other is known. Although the challenge of an unstructured tetrahedral mesh is to manage the bookkeeping of how the cells (and in the planned parallel environment, the sub-domains) fit together, this algorithm has demonstrated its potential to overcome this challenge of spatial domain decomposition, thus paving the way for an effective massively parallel method of solving the neutron transport equation on an unstructured tetrahedral mesh.

المنارة للاستشارات
www.manaraa.com

REFERENCES

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

[2] B. G. Carlson and K. D. Lathrop, Transport Theory - The Method of Discrete Ordinates, in *Computing Methods in Reactor Physics*, H. Greenspan, C.N. Kelber and D. Okrent (eds), Gordon and Breach, New York, (1968).

[3] E. W. Larsen and J. E. Morel, Advances in Discrete-Ordinates Methodology, in *Nuclear Computational Science*, Y.Y. Azmy and E. Sartori (eds), Springer, New York, (2010).

[4] R. J. Zerr, Solution of the Within Group Multidimensional Discrete Ordinates Transport Equations on Massively Parallel Architectures, PhD Thesis, The Pennsylvania State University (2011).

[5] Wienke B. R. and Hiromoto R. E., Parallel S_n Iteration Schemes, *Nuclear Science and Engineering*, **90**, 116 (1985).

[6] Wienke B. R. and Hiromoto R. E., Parallel S_n Transport Algorithms, *Transport Theory and Statistical Physics*, **15**, 49 (1986).

[7] Wienke B. R., Hiromoto R. E., and Brickner R. G., Parallel Discrete Ordinates Algorithms on Distributed and Common Memory Systems, *Transactions of the American Nuclear Society*, **55**, 321 (1987)

[8] J. W. Fischer and Y. Y. Azmy, Comparison via parallel performance models of angular and spatial domain decompositions for solving neutral particle transport problems, *Progress in Nuclear Energy*, **49**, 37 (2007).

[9] Y. Y. Azmy, Multiprocessing for neutron diffusion and deterministic transport methods, *Progress in Nuclear Energy*, **31** (3), 317 (1997).

[10] M. Yavuz and E. W. Larsen, ”Spatial Domain Decomposition for Neutron Transport Problems,” *Transport Theory and Statistical Physics*, **18** (2), 205 (1989).

للاستشارات المنارة

www.manaraa.com

[11] M. Yavuz and E. W. Larsen, ”Iterative Methods for Solving x-y Geometry S_N Problems on Parallel Architecture Computers,” *Nuclear Science and Engineering*, **112**, 32 (1992).

[12] K. R. Koch, R. S. Baker, and R. E. Alcouffe, Solution of the First-Order Form of Three-Dimensional Discrete Ordinates Equations on a Massively Parallel Machine, *Transactions of the American Nuclear Society*, **65**, 198 (1992).

[13] R. S. Baker and K. R. Koch, An S_n Algorithm for the Massively Parallel CM-200 Computer, *Nuclear Science and Engineering*, **128**, 312 (1998).

[14] S. O. Lindahl and Z. Weiss, The Response Matrix Method, *Advances in Nuclear Science and Technology*, **13**, 73 (1981).

[15] U. R. Hanebutte and E. E. Lewis, A Massively Parallel Discrete Ordinates Response Matrix Method for Neutron Transport, *Nuclear Science and Engineering*, **111**, 46 (1992).

[16] Y. Y. Azmy, A New Algorithm for Generating Highly Accurate Benchmark Solutions to Transport Test Problems, *Proceedings of the XI ENFIR/IV ENAN Joint Nuclear Conferences, Pocos de Caldas Springs*, MG, Brazil, August 1822, 1997 (1997).

[17] Y. Y. Azmy, Iterative Convergence Acceleration of Neutral Particle Transport Methods via Adjacent-Cell Preconditioners, *Journal of Computational Physics*, **152**, 359 (1999).

[18] M. Rosa, Y. Y. Azmy, and J. E. Morel, Properties of the S_NEquivalent Integral Transport Operator in Slab Geometry and the Iterative Acceleration of Neutral Particle Transport Methods, *Nuclear Science and Engineering*, **162** (3), 234 (2009).

[19] R. J. Zerr, Personal Communication, January 10, 2012 (2012).

المنارة للاستشارات

www.manaraa.com